Proceedings of the 6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2021)

Frederic Font, Annamaria Mesaros, Daniel P.W. Ellis, Eduardo Fonseca, Magdalena Fuentes, and Benjamin Elizalde (eds.)

November 15-19, 2021

Citation:

Frederic Font, Annamaria Mesaros, Daniel P.W. Ellis, Eduardo Fonseca, Magdalena Fuentes, and Benjamin Elizalde (eds.), Proceedings of the 6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2021), Nov. 2021.

DOI: 10.5281/zenodo.5770113

ISBN: 978-84-09-36072-7

Table of contents

Preface

This volume is a collection of the papers presented at the Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE 2021) in Barcelona (online), on November 15-19, 2021.

The DCASE 2021 Workshop is the sixth workshop on Detection and Classification of Acoustic Scenes and Events, organized again in conjunction with the DCASE Challenge. The aim of the workshop was to bring together researchers from many different universities and companies with interest in the topic, and provide the opportunity for scientific exchange of ideas and opinions.

The DCASE 2021 Workshop was jointly organized by researchers at Universitat Pompeu Fabra, Tampere University, Google Inc., New York University, Microsoft, Universidad de la República, Apple Inc., Adobe Research, Voicemod and Bose Corporation. The associated DCASE 2021 Challenge tasks were organized by researchers at Tampere University (Task 1: Acoustic scene classification, Task 3: Sound Event Localization and Detection with Directional Interference, Task 6: Automated Audio Captioning); Hitachi Ltd. (Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring under Domain Shifted Conditions); Google Inc. (Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring under Domain Shifted Conditions, Task 4: Sound Event Detection and Separation in Domestic Environments); NTT Corporation (Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring under Domain Shifted Conditions); Inria Nancy Grand-Est (Task 3: Sound Event Localization and Detection with Directional Interference, Task 4: Sound Event Detection and Separation in Domestic Environments); University of Lorraine (Task 4: Sound Event Detection and Separation in Domestic Environments); Adobe Research (Task 4: Sound Event Detection and Separation in Domestic Environments); Universitat Pompeu Fabra (Task 4: Sound Event Detection and Separation in Domestic Environments); Northwestern University (Task 4: Sound Event Detection and Separation in Domestic Environments); Università Politecnica delle Marche (Task 4: Sound Event Detection and Separation in Domestic Environments); Queen Mary University of London (Task 5: Few-shot Bioacoustic Event Detection); Tilburg University (Task 5: Few-shot Bioacoustic Event Detection); University of Konstanz (Task 5: Few-shot Bioacoustic Event Detection); BIOTOPIA Naturkundemuseum Bayern (Task 5: Few-shot Bioacoustic Event Detection); and AGH University of Science and Technology (Task 5: Few-shot Bioacoustic Event Detection).

For this edition of the DCASE 2021 Workshop, 67 full papers were submitted, each reviewed by at least three members of our Technical Program Committee. From these, 47 papers were accepted.

The Organizing Committee was also pleased to invite leading experts for keynote addresses: Laurie Heller (Professor and director of the Auditory Lab at Carnegie Mellon University), and Kristen Grauman (Professor at Department of Computer Science University of Texas at Austin and Research Director at Facebook AI Research).

The success of the DCASE 2021 Workshop was the result of the hard work of many people whom we wish to warmly thank here, including all the authors and keynote speakers, as well as all the members of the Technical Program Committee, without whom this edition of the DCASE 2021 Workshop would not exist. We also wish to thank the organizers and participants of the DCASE Challenge tasks.

This edition of the workshop was supported by sponsorship from Facebook, Apple Inc., Bose Corporation, Hitachi Ltd., Line, Audio Analytic, Cochlear.ai, Dolby, Google Inc., Mitsubishi Electric, and RION. We wish to thank them warmly for their valuable support to this workshop and the expanding topic area.

Frederic Font, Annamaria Mesaros, Daniel P.W. Ellis,
Eduardo Fonseca, Magdalena Fuentes, and Benjamin Elizalde

DCASE 2021 Sponsors

TOYADMOS2: ANOTHER DATASET OF MINIATURE-MACHINE OPERATING SOUNDS FOR ANOMALOUS SOUND DETECTION UNDER DOMAIN SHIFT CONDITIONS

Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi
Masahiro Yasuda and Shoichiro Saito

NTT Corporation, Japan

ABSTRACT

This paper proposes a new large-scale dataset called "ToyADMOS" for anomaly detection in machine operating sounds (ADMOS). As with our previous ToyADMOS dataset, we collected a large number of operating sounds of miniature machines (toys) under normal and anomaly conditions by deliberately damaging them, but extended them in this case by providing a controlled depth of damages in the anomaly samples. Since typical application scenarios of ADMOS require robust performance under domain-shift conditions, the ToyADMOS2 dataset is designed for evaluating systems under such conditions. The released dataset consists of two sub-datasets for machine-condition inspection: fault diagnosis of machines with geometrically fixed tasks and fault diagnosis of machines with moving tasks. Domain shifts are represented by introducing several differences in operating conditions, such as the use of the same machine type but with different models and parts configurations, operating speeds, microphone arrangements, etc. Each subdataset contains over 27 k samples of normal machine-operating sounds and over 8 k samples of anomalous sounds recorded with five to eight microphones. The dataset is freely available for download at https://github.com/nttcslab/ToyADMOS2-dataset and https://doi.org/10.5281/zenodo.4580270.

***Index Terms*—** Anomaly detection in sounds, machine operating sounds, product inspection, dataset, domain shift conditions

1. INTRODUCTION

Extensive research efforts have recently been focused on anomaly detection. An anomaly detection task is designed to detect anomaly states by learning only normal condition data. Microphones have been used as sensors to detect anomalies, referred to as anomaly sound detection (ASD) or acoustic condition monitoring [1–9]. This task setting is different from other sound event detection tasks such as gunshot detection [10].

In general, it is very difficult or almost impossible to collect massive anomaly data. The ToyADMOS [11] and MIMII [12] datasets were the first to be utilized for evaluating anomaly detection systems using sound. These datasets enable us to compare the performance of different systems. In 2020, a number of systems from various research organizations in academia and industry were submitted to DCASE 2020 Challenge Task 2: Unsupervised detection of anomalous sounds for machine condition monitoring [13]. The submitted systems performed quite well on the task, thus demonstrating the great potential of applying deep learning-based systems for unsupervised anomaly detection tasks [14–19].

However, the ASD scenario given in the DCASE 2020 Challenge was not realistic compared to actual use cases. The task setting was too basic, and the task requirements were much easier than they would be for typical ones in practical applications, where (for example) the same machine type but different models are used at different operating speeds, and the conditions are not given as training data. Several independent research groups have tackled tasks related to domain shift or domain adaptation [20–25], but few open datasets that could serve this need have been made available. While the previous ToyADMOS dataset has some data variations that can be used for testing domain-shift conditions, we would like to have more variations on the test configuration.

When evaluating the performance of ASD systems, the statistical characteristics of anomalous sound samples should be different from those of normal samples. However, if the difference is too significant, the anomaly detection task might not be difficult enough to properly evaluate the system performance. One way of controlling the difficultly of the test configuration is to adjust the signal-to-noise ratio (SNR) of the added noise level, but noise reduction techniques, such as the ones in [15, 26], can be used to mitigate the difficulty of the task. Therefore, there should be a way to control the difficulty, without relying on the SNR.

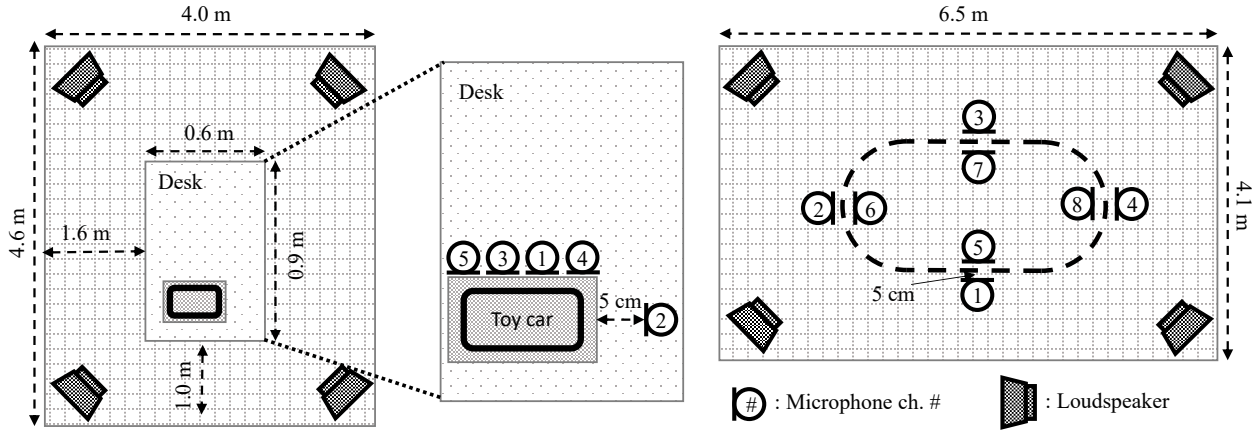
The difficulty of the task under domain shifts can be controlled by changing the statistical difference among normal samples across domains and/or the statistical difference between normal and anomaly sound samples within a domain. In designing challenging test conditions, it is preferred to strike an appropriate balance between these two approaches.

To address the application scenarios discussed above, we provide a new ADMOS dataset called ToyADMOS2. The ToyADMOS2 dataset adds more variations on condition arrangements dedicated to domain shifts. As we did for the previous ToyADMOS dataset, we collected normal and anomalous operating sounds of miniature machines by deliberately damaging their components. The ToyADMOS2 dataset has the following characteristics:

- Designed for two ADMOS tasks: product inspection (toy car) and fault diagnosis for a moving machine (toy train).
- Provides controlled domain-shift conditions on machine models, parts configurations, operating speeds, microphone models and arrangements, and environmental noise.
- Enables control of depth of damage in anomaly samples to provide choices on significance levels of statistical differences between normal and anomalous samples.

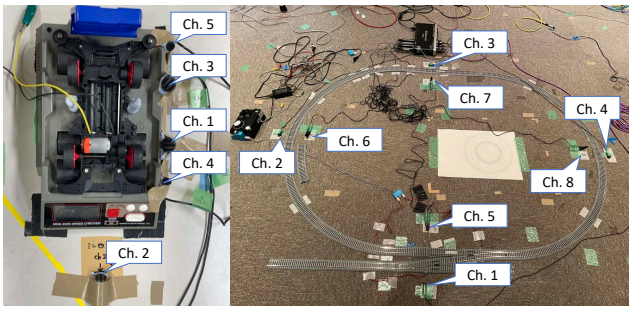
Note that the ToyADMOS2 dataset can be used alongside the previous ToyADMOS dataset to provide a larger variety of test conditions.

The proposed ToyADMOS2 dataset is freely available for download at https://github.com/nttcslab/ToyADMOS2-dataset and https://doi.org/10.5281/zenodo.4580270. The license and explanation of some of its uses are also available at those links.



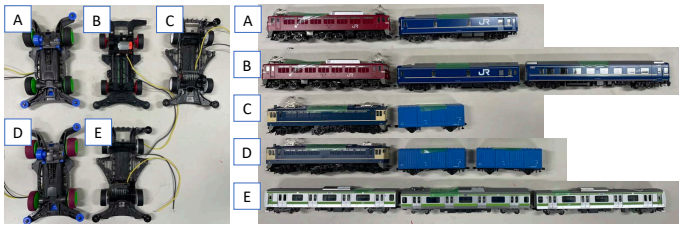
(a) (b)

Figure 1: Recording-room layouts and microphone arrangements.



(a) (b)

Figure 2: Images of microphone arrangements.



(a) (b)

Figure 3: Images of toy-model configurations.

2. DATASET OVERVIEW

The ToyADMOS2 dataset consists of two sub-datasets for two types of ADMOS task. Each sub-dataset has three domain shift conditions. A different type of toy, namely a toy car and a toy train, is used for each task.

Toy car: Designed for the product-inspection task. The toy car runs on an inspection device. Sound data were collected with five microphones arranged close to the inspection device, as shown in Figs. 1(a) and 2(a). The setting is similar to the toy car sub-dataset in ToyADMOS, but the machine models, parts configurations, operating speed settings, and microphone arrangements are different. The toy-car models are shown in Fig. 3(a).

Toy train: Designed for fault diagnosis of a moving machine. The toy train runs on a railway track. Sound data were collected with eight microphones surrounding the track, as shown in Figs. 1(b) and 2(b). The setting is similar to that of the toy train sub-dataset of the ToyADMOS dataset, but the machine models, cargo arrangements, operating speed settings, and microphone arrangements are different. The toy-train models are shown in Fig. 3(b).

To collect various normal and anomalous sounds depending on individual differences, operating sounds were recorded using several models with different parts configurations.

Normal sound: Operating sound when the target machine operates normally in accordance with its specifications.

Anomalous sound: Operating sound when the target machine was forced to operate anomalously by deliberately damaging its components or adding extraneous objects. The anomaly level was controlled by changing the depth of the damage deliberately made on the parts.

Environmental noise: Environmental noise for simulating a factory or other locations such as a roadside. Noise samples were collected at several locations with multiple microphones. These sounds were emitted from four loudspeakers positioned at the corners of each recording room and recorded with the same microphone configurations used for recording the normal and anomalous sound samples. In addition to newly recorded noise samples, noises from the ToyADMOS dataset were also used.

We utilized omnidirectional dynamic (SURE SM11-CN) and condenser (TOMOCA EM-700) microphones to collect these sounds. All sounds were stored as multiple WAV files.

In ToyADMOS2, several application scenarios representing domain-shift conditions were designed. Examples of possible domain-shift scenarios are as follows:

Model and parts configuration shift: The normal and anomaly data were recorded using several different machine models and parts configurations.

Operating speed shift: Operating speed was varied. Normal and anomaly data were recorded under several different speed levels.

Microphone and environmental noise shift: Different environmental noise was recorded at different positions using different types of microphone.

3. DETAILS OF SUB-DATASETS

3.1. Toy-car sub-dataset

We assumed a product inspection task and took up the task of detecting anomalous sounds from the running sound of a toy car on an inspection device, as shown in Fig. 2(a). The miniature car machine was a toy car called "mini 4WD", which has four wheels and is driven by a small motor through gears and a shaft. A stabilized power supply was connected to the motor, and running sounds on an inspection device were recorded with five microphones. There were five machine configurations: A, B, C, D, and E, as shown in Fig. 3(a). In each configuration, different machine chassis models and parts were used.

To control the operating speed, five voltage levels, 2.8, 3.1, 3.4, 3.7, and 4.0 V, were provided through the stabilized power supply. Each recorded WAV file contains a normal or anomalous sound of a toy car. The duration of the normal and anomalous sound samples is 12 s. This results in over 35 k samples times five channels; in total over 177 k sound samples. Anomalous sounds were generated by deliberately damaging the shaft, gears, or tires. Three damage depth levels were provided for each part, as listed in Table 1. In total, over 8 k samples of anomalous sounds were recorded with various combinations (300 patterns) including three different depths of damage.

For recording the environmental noise, four types of noise files were played through the four loudspeakers and recorded with the same microphone setting shown in Figs. 1(a) and 2(a), and normal and anomalous sounds were recorded. Three dynamic and two condenser microphones were used. Some of the noise files were newly recorded; others were taken from ToyADMOS.

3.2. Toy-train sub-dataset

We assumed the use of fault diagnosis of a moving machine task of anomalous sounds from the running sound of a toy train. We used an HO-scale model train which is a precisely detailed miniature, as shown in Fig. 3(b). Sound data were collected with eight microphones positioned inside and outside the railway track perimeter (four microphones each). The microphones and loudspeakers in the recording room were arranged as shown in Figs. 1(b) and 2(b). Dynamic microphones were used as the four outer microphones (channels 1 to 4), and condenser microphones were used as the four inner microphones (channels 5 to 8). Domain shift conditions in each task configuration are listed in Table 1. There are five machine configurations (A, B, C, D, and E).

To control the operating speed, five levels of speed control (5, 6, 7, 8, and 9) were used. Each recorded WAV file contains normal or anomalous sounds of a toy train. The duration of the normal and anomalous sound samples are 12 s each. This results in over 35 k samples times two mic types; in total, over 71 k sound samples. Anomalous sounds were generated by deliberately damaging the carriage and railway track. Three depth levels of the damage were provided to each part, as listed in Table 1. In total, over 8 k samples of anomalous sounds were recorded with their various combinations (300 patterns).

For recording the environmental noise, four types of noise files were played through the four loudspeakers and recorded with the same microphone setting (shown in Figs. 1(b) and 2(b)) that the normal and anomalous sounds were recorded with. Four dynamic microphones (channels 1 to 4) and four condenser microphones (channels 5 to 8) were used. Some of the noise files were newly recorded; others were taken from ToyADMOS.

Table 1: Anomaly conditions of sub-datasets.

Toy car		**Toy train**	
Part	Condition	Part	Condition
Shaft	- Bent	Carriage	- Flat tire
Gears	- Deformed		- Broken shaft
	- Melted	Railway track	- Disjointed
Wheels	- Damaged	(straight, curve)	- Obstructing stone

Table 2: Variation settings of sub-datasets.

	Toy car	**Toy train**
Model variations	Five	Five
Speed levels	Five	Five
Mic. type and channel config.	Dynamic: 1–3 Condenser: 4, 5	Dynamic: 1–4 Condenser: 5–8
Noise type	Four recordings	
Normal samples (Total hours)	1,094 samples×5 models×5 speed levels (91 hours × 5 ch) (91 hours×2 ch-sets)	
Anomaly samples (Total hours)	324 samples×5 models×5 speed levels (27 hours×5 ch) (27 hours×2 ch-sets)	
Noise samples	24 hours per a channel	

Table 3: Example domain-shift task configurations.

Domain	**Configuration**		**Training data**
Source domain	Car/Train model	B	1,500 normal samples
	Speed level	1, 3	
	Mic. type	Dynamic	
	Noise	N1	
Target domain 1: Model and parts shift	**Car/Train model**	**D**	Four normal samples
	Speed level	1, 3	
	Mic. type	Dynamic	
	Noise	N1	
Target domain 2: Operating speed shift	Car/Train model	B	Four normal samples
	Speed level	**2, 5**	
	Mic. type	Dynamic	
	Noise	N1	
Target domain 3: Mic. and noise shift	Car/Train model	B	Four normal samples
	Speed level	1, 3	
	Mic. type	**Condenser**	
	Noise	**N2**	
Target domain 4: All of above	**Car/Train model**	**D**	Four normal samples
	Speed level	**2, 5**	
	Mic. type	**Condenser**	
	Noise	**N2**	

Test data for each domain consists of 100 normal and 100 anomaly samples.

Further details of the ToyADMOS2 dataset are available at https://github.com/nttcslab/ToyADMOS2-dataset.

4. SAMPLE DOMAIN-SHIFT TASK SETTINGS AND BENCHMARK

To demonstrate how to use the ToyADMOS2 dataset, we designed examples of task configurations for testing ASD systems under various domain-shift conditions. The domain-shift configurations are listed in Table 3.

We assume a typical application scenario with a task setting similar to that in [21]. A relatively large number of normal samples

Table 4: Benchmark AUC results of DCASE 2020 Challenge Task 2 baseline under domain-shift conditions.

		Toy car				Toy train			
	Damage level	**Clean**	**6 dB**	**0 dB**	**–6 dB**	**Clean**	**6 dB**	**0 dB**	**–6 dB**
Source domain	high	**0.99**	0.99	0.96	0.85	**1.00**	0.86	0.66	0.61
	mid	**0.99**	0.95	0.94	0.80	**0.98**	0.66	0.57	0.56
	low	**1.00**	0.96	0.95	0.80	**0.83**	0.57	0.59	0.49
	avg.	**0.99**	0.97	0.95	0.82	**0.94**	0.70	0.61	0.55
Target domain 1:	high	0.37	0.29	0.34	0.38	0.82	0.75	0.71	0.62
Model and parts shift	mid	0.92	0.55	0.58	0.59	0.71	0.61	0.61	0.53
	low	0.80	0.69	0.68	0.69	0.57	0.56	0.58	0.58
	avg.	0.70	0.51	0.53	0.55	0.70	0.64	0.63	0.57
Target domain 2:	high	0.77	0.78	0.83	0.76	0.71	0.65	0.62	0.57
Operating speed shift	mid	0.61	0.63	0.65	0.66	0.65	0.61	0.51	0.52
	low	0.64	0.72	0.67	0.69	0.52	0.52	0.48	0.44
	avg.	0.68	0.71	0.71	0.70	0.63	0.59	0.54	0.51
Target domain 3:	high	0.55	0.53	0.45	0.52	0.72	0.62	0.57	0.57
Mic. type and noise shift	mid	0.39	0.55	0.47	0.48	0.66	0.55	0.54	0.52
	low	0.45	0.57	0.56	0.52	0.65	0.52	0.52	0.52
	avg.	0.46	0.55	0.49	0.50	0.68	0.56	0.54	0.54
Target domain 4:	high	0.55	0.71	0.59	0.65	0.60	0.57	0.53	0.49
All above shifts	mid	0.44	0.64	0.54	0.56	0.51	0.45	0.44	0.45
	low	0.52	0.71	0.55	0.65	0.51	0.45	0.42	0.44
	avg.	0.50	0.69	0.56	0.62	0.54	0.49	0.46	0.46

recorded in a source domain (e.g., 1,500 samples) are available as a training dataset; however, a very limited number of normal samples recorded in the target domains (four samples each) are given for training. As shown in Table 3, there are four example domain-shift configurations: **Target domain 1**, Model and parts shift; **Target domain 2**, Operating speed shift; **Target domain 3**, Mic. type and noise shift; and **Target domain 4**, A mixture of shifts 1 to 3.

For the source domain training dataset, 1,500 normal samples of Toy-car model B and Toy-train model B were used. For Toy car, normal sample data recorded with one dynamic microphone (ch 1) were used. For Toy train, normal sample data recorded by dynamic microphones (ch 1 to 4) were used. In these normal samples, operating speeds were set to levels 1 and 3. For target domains 1 to 4, only four normal samples were given as training data for each of the target domains. Environmental noises N1 and N2 recorded using microphones at the same positions as for the normal samples were used. SNRs calculated by

$$SNR = 20\log_{10}\left\{\frac{1}{J}\sum rms(S_j)\right\}/rms(N_k), \qquad (1)$$

where S_j is a recorded machine-operating sound sample and N_k is a recorded environmental noise sample with the same duration as S_j, were set to $+\infty$ dB (clean), 6 dB, 0 dB, and –6 dB. After being mixed with the noise, all sound samples were down-sampled at a sampling rate of 16 kHz. Detailed conditions of each target domain are shown in Table 3.

To simplify the evaluation system example, a baseline system of DCASE 2020 Challenge Task 2 [13]—a simple unsupervised-ASD one with an auto encoder as a normal model—was tested under the task setting described above. All the training data—1,500 normal samples of the source domain, and other normal samples from the target domains (four samples each)— were merged to formulate a training dataset that contains 1,516 normal samples. The baseline system was trained for each SNR condition of Toy car and Toy train.

The trained baseline systems were tested using 200 unknown test samples that were not used for training. We calculated anomaly scores on each time frame for all the test WAV files as described in [13]. Calculated area under curve (AUC) results are shown in Table 4. The results in bold indicate that when the damage level was higher, anomaly detection was easier for the source domain/clean for Toy train. For Toy car, the AUC results were around 0.99 for all the levels. Distribution of the anomaly scores for the source domain/clean samples are shown in Fig. 4. The target domain results show that the DCASE baseline failed to distinguish normal and anomaly data under domain-shifted conditions even with a high damage level.

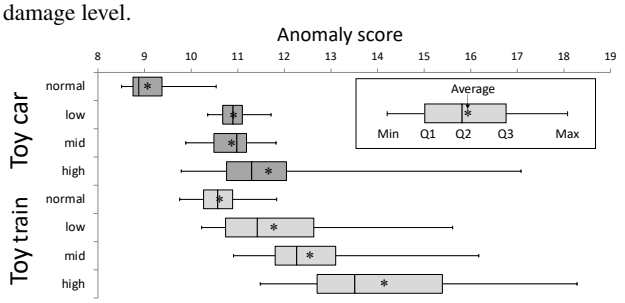


Figure 4: Anomaly scores of source domain/clean samples.

5. CONCLUSION

This paper proposed a new large-scale dataset called "ToyADMOS2" for ADMOS. The ToyADMOS2 dataset is designed for evaluating systems under domain-shift conditions and is available for free download. It consists of two sub-datasets for machine-condition inspection: fault diagnosis of machines with geometrically fixed tasks and fault diagnosis of machines with moving tasks. Domain shifts are represented by introducing several differences in operating conditions, such as the use of the same machine type but with different machine models and parts configurations, operating speeds, microphone arrangements, etc. Each sub-dataset contains over 27 k samples of normal machine-operating sounds and over 8 k samples of anomalous sounds recorded at a 48-kHz sampling rate. A subset of the ToyADMOS2 dataset was used in the DCASE 2021 Challenge Task 2: Unsupervised anomalous sound detection for machine condition monitoring under domain-shifted conditions.

6. REFERENCES

[1] Y. Koizumi, S. Saito, H. Uematsu, Y. Kawachi, and N. Harada, "Unsupervised detection of anomalous sound based on deep learning and the neyman–pearson lemma," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 212–224, 2019.

[2] E. Rushe and B. M. Namee, "Anomaly detection in raw audio using deep autoregressive networks," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3597–3601.

[3] Y. Koizumi, S. Saito, H. Uematsu, and N. Harada, "Optimizing acoustic feature extractor for anomalous sound detection based on neyman-pearson lemma," in *25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 698–702.

[4] Y. Kawachi, Y. Koizumi, and N. Harada, "Complementary set variational autoencoder for supervised anomaly detection," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2366–2370.

[5] Y. Koizumi, S. Murata, N. Harada, S. Saito, and H. Uematsu, "Sniper: Few-shot learning for anomaly detection to minimize false-negative rate with ensured true-positive rate," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 915–919.

[6] Y. Kawachi, Y. Koizumi, S. Murata, and N. Harada, "A two-class hyper-spherical autoencoder for supervised anomaly detection," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3047–3051.

[7] Y. Koizumi, S. Saito, M. Yamaguchi, S. Murata, and N. Harada, "Batch uniformization for minimizing maximum anomaly score of dnn-based anomaly detection in sounds," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019, pp. 6–10.

[8] Y. Koizumi, M. Yasuda, S. Murata, S. Saito, H. Uematsu, and N. Harada, "Spidernet: Attention network for one-shot anomaly detection in sounds," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 281–285.

[9] S. Heincke, A. K. Kalan, O. J. Wagner, R. Mundry, H. Lukashevich, and H. S. Kühl, "Assessing the performance of a semi-automated acoustic monitoring system for primates," *Methods in Ecology and Evolution*, pp. 753–763, 2015.

[10] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, 2007, pp. 21–26.

[11] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, November 2019, pp. 308–312.

[12] H. Purohit, R. Tanabe, T. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, "MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, November 2019, pp. 209–213.

[13] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on dcase2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, pp. 81–85.

[14] R. Giri, S. V. Tenneti, K. Helwani, F. Cheng, U. Isik, and A. Krishnaswamy, "Unsupervised anomalous sound detection using self-supervised classification and group masked autoencoder for density estimation," DCASE2020 Challenge, Tech. Rep., July 2020.

[15] P. Daniluk, M. Gozdziewski, S. Kapka, and M. Kosmider, "Ensemble of auto-encoder based systems for anomaly detection," DCASE2020 Challenge, Tech. Rep., July 2020.

[16] P. Primus, "Reframing unsupervised machine condition monitoring as a supervised classification task with outlier-exposed classifiers," DCASE2020 Challenge, Tech. Rep., July 2020.

[17] P. Vinayavekhin, T. Inoue, S. Morikuni, S. Wang, T. Hoang Trong, D. Wood, M. Tatsubori, and R. Tachibana, "Detection of anomalous sounds for machine condition monitoring using classification confidence," DCASE2020 Challenge, Tech. Rep., July 2020.

[18] T. Hayashi, T. Yoshimura, and Y. Adachi, "Conformer-based id-aware autoencoder for unsupervised anomalous sound detection," DCASE2020 Challenge, Tech. Rep., July 2020.

[19] Q. Zhou, "Arcface based sound mobilenets for dcase 2020 task 2," DCASE2020 Challenge, Tech. Rep., July 2020.

[20] M. Yamaguchi, Y. Koizumi, and N. Harada, "Adaflow: Domain-adaptive density estimator with application to anomaly detection and unpaired cross-domain translation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2019*, 2019, pp. 3647–3651.

[21] Q. Wang, G. Michau, and O. Fink, "Domain adaptive transfer learning for fault diagnosis," in *2019 Prognostics and System Health Management Conference (PHM-Paris)*, 2019, pp. 279–285.

[22] G. Michau and O. Fink, "Domain adaptation for one-class classification: Monitoring the health of critical systems under limited information," *International journal of prognostics and health management.*, vol. 10, 2019.

[23] A. Kumagai, T. Iwata, and Y. Fujiwara, "Transfer anomaly detection by inferring latent domain representations," in *NeurIPS*, 2019.

[24] Z. Yang, I. S. Bozchalooi, and E. F. Darve, "Anomaly detection with domain adaptation," *ArXiv*, vol. abs/2006.03689, 2020.

[25] V. Vincent, M. Wannes, and D. Jesse, "Transfer learning for anomaly detection through localized and unsupervised instance selection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 6054–6061, Apr. 2020.

[26] Y. Kawaguchi, R. Tanabe, T. Endo, K. Ichige, and K. Hamada, "Anomaly detection based on an ensemble of dereverberation and anomalous sound extraction," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 865–869.

AUTOMATED AUDIO CAPTIONING WITH WEAKLY SUPERVISED PRE-TRAINING AND WORD SELECTION METHODS

Qichen Han, Weiqiang Yuan*, Dong Liu, Xiang Li, Zhen Yang*

NetEase (Hangzhou) Network Co., Ltd., China,
{hanqichen, yuanweiqiang, hzliudong, hzlixiang, yangzhen1}@corp.netease.com

ABSTRACT

Audio captioning is a multi-modal task, focusing on generating a natural sentence to describe the content in an audio clip. This paper proposes a solution of automated audio captioning based on weakly supervised pre-training and word selection methods. Our solution focuses on solving two problems in automated audio captioning: data insufficiency and word selection indeterminacy. As the amount of training data is limited, we collect large-scale weakly labeled dataset from Web with heuristic methods. Then we pre-train the encoder-decoder models with this dataset followed by fine-tuning on the Clotho dataset. To solve the word selection indeterminacy problem, we use keywords extracted from captions of similar audios and audio tags produced by pre-trained audio tagging models to guide caption generation. The proposed system achieves the best SPIDEr score of 0.310 in the DCASE 2021 Challenge Task 6.

Index Terms— Audio captioning, encoder-decoder modeling, weakly supervised pre-training, audio similarity, audio tag

1. INTRODUCTION

The automated audio captioning (AAC) problem is defined as an intermodal translation task of automatically generating a textual description for an input audio signal [1]. This task needs information including identification of sound events, acoustic scenes, spatiotemporal relationships of sources, foreground versus background discrimination, concepts, and physical properties of objects and environment [2]. Audio captioning needs to extract the feature representation of audio space and map it to natural language space. Therefore, most of the previous works adopt encoder-decoder framework [3-6]. Our solution focuses on solving two problems in automated audio captioning: data insufficiency and word selection indeterminacy.

As the amount of training data in the audio captioning task is limited, training a well generalized end-to-end model is difficult. It is well-established that pre-training on large datasets followed by fine-tuning on target datasets boosts performance [7]. We use heuristic methods to collect a weakly labeled dataset for pretraining, which contains 65667 audios and corresponding captions. In addition, our system uses PANN's [8] architecture as an encoder, which is trained on the large-scale AudioSet [9] dataset.

In AAC task, one acoustic event/scene in an audio can be described with different words, leading to a combinatorial explosion of possible captions [3]. This word selection indeterminacy problem may lead to difficulty in training. We try two methods to tackle this problem. Firstly, considering that similar audios may have similar captions, we train a model to calculate the similarity between audios and use keywords extracted from the captions of similar audios to assist decoding. Secondly, we try to use audio tag information to assist decoding.

The contributions of this work are in the following aspects. Firstly, we propose a method to use pretrained PANN models as encoder and to pretrain the whole model on a large weakly labeled dataset. Secondly, to relieve the word selection indeterminacy, we introduce audio tags and caption keywords in the decoding stage. Thirdly, ablation studies are conducted to confirm the effectiveness of different strategies in the proposed approach.

The paper is organized as follows: Section 2 describes the proposed method for DCASE 2021 audio captioning challenge. Section 3 introduces the ablation study experiment setup. The experimental results are presented in Section 4. Section 5 concludes this work.

2. SYSTEM DESCRIPTION

This section describes our methods. Please refer to our technical report for more details [10].

2.1. Data augmentation

Perturb audio data In the Clotho dataset, each audio has five captions. Using audio augmentation methods such as speed perturbation [11] and reverberation [12], we perform a 5-fold augmentation of the Clotho dataset.

* These authors contributed equally to this work.

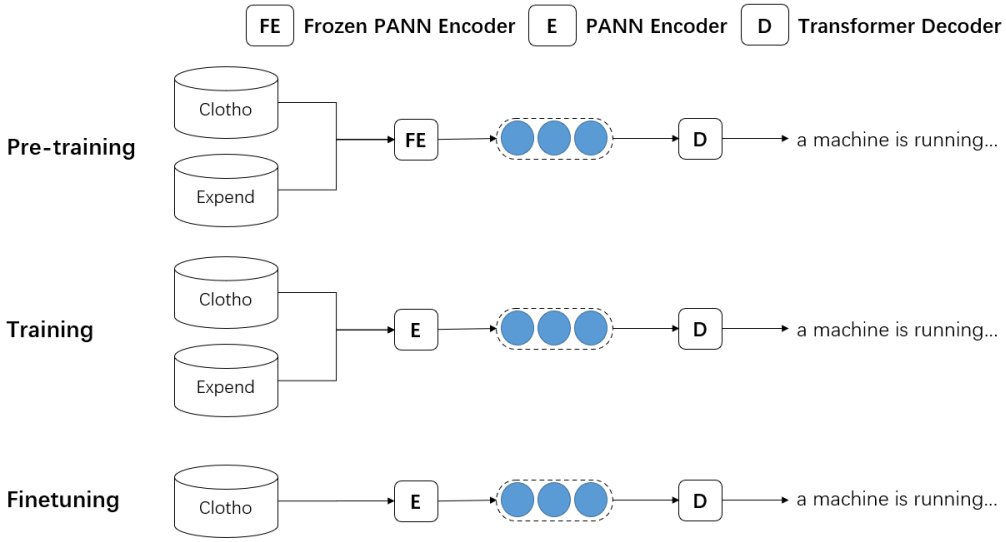


Figure 1: The overview diagram of the proposed method. The Expend data includes AudioCaps and weakly labeled dataset.

AudioCaps dataset We also add AudioCaps [13] for training, which is a large-scale dataset of about 51K audio clips to human-written text pairs collected via crowdsourcing on the AudioSet dataset.

Weakly labeled dataset We collect audios and corresponding descriptions from the Freesound[1], Zapsplat[2], Soundbible[3] and SoundJay[4] website. Audios that are shorter than 5 seconds are removed. And for those longer than 30 seconds, we randomly select 15 to 30 seconds of clips from the audio. We use heuristic rules to filter and clean captions [10]. As a result, we collected 65667 audios with captions from the four websites.

2.2. Pretrain encoder

PANNs are models pre-trained on raw AudioSet recordings with different structures. Several PANN systems outperform previous state-of-the-art audio tagging systems and can be transferred to other audio comprehension tasks. Three different networks in PANNs are selected as encoders, namely CNN14, Resnet38 and Wavegram-Logmel CNN[5].

2.3. Similar audio searching

As mentioned above, similar audios may have similar descriptions. For an audio without captions, we can get relevant keywords from the captions of its similar audios, which can help to generate better caption.

Inspired by text similarity calculation methods such as ESIM [14], we design a model to calculate the similarity between audios. We use CNN14[6] as audio encoder and get the 2048-dimension feature sequence. Then, we treat each audio feature sequence as the word embedding sequence and similarity between two audios is calculated with the ESIM network.

For training, we use SPIDEr score between captions of two audios as their ground truth similarity. We train this model with triplet dynamic margin loss. Given an anchor audio a, its similar audio p, and unsimilar audio n, the loss calculation is defined as follows:

$$Loss(a,p,n) = \max(0, m(a,p,n) + s(a,p) - s(a,n)) \quad (1)$$

where $s(.)$ is the similarity as mentioned above, and $m(.)$ is the margin function. The margins for each pair of (a, p, n) are different. We calculate the margins by the SPIDEr scores between captions of audios:

$$m(a,p,n) = \max(0.4, SPIDEr(a,p) - SPIDEr(a,n)) \quad (2)$$

2.4. Decoder

As in [4], we use transformer networks as decoder.

Tag enhanced decoder In order to reduce the search space, we utilize audio tag information by adding it to the beginning of the output sequence. The tags we use is based on AudioSet Ontology[7]. To avoid the problem of sparse data, we merge fine-grained tags with the help of the structural features of the ontology and get 13 tags, named Self-Tag-13. CNN14 is used to get the Self-Tag-13 tags of audios in training set and test set. During the training process, the decoder needs to predict the tag of audio before generating caption. In the test phase, the decoder generates caption given the corresponding tag of test audio.

Keyword enhanced decoder We extract keywords from captions of similar audios. Specifically, we select 50 captions of the top 10 most similar audios for the target audio, use NLTK[8] to perform stemming, and extract the top 10 keyword stems according to the TF-IDF weight. In the decoding stage, a fixed boost score is added to log likelihood for all word forms of the keywords. The boost score is set to 0.5.

[1] https://freesound.org
[2] https://www.zapsplat.com/
[3] https://soundbible.com/
[4] https://www.soundjay.com/

[5] https://zenodo.org/record/3987831#.YMhofqgzaUk
[6] https://github.com/qiuqiangkong/audioset_tagging_cnn
[7] http://research.google.com/audioset/ontology/index.html
[8] https://www.nltk.org/

Table 1: Ablation study. PE: Pre-trained encoder. KD: Keyword enhanced decoder. TD: Tag enhanced decoder. PD: Perturbed audio data. AD: AudioCaps dataset. WD: Weak label dataset.

Model	$BLUE_1$	$BLUE_2$	$BLUE_3$	$BLUE_4$	METEOR	ROUGE-L	CIDEr	SPICE	SPIDEr
Baseline	0.521	0.328	0.216	0.139	0.153	0.353	0.326	0.102	0.2142
PE	0.541	0.348	0.228	0.149	0.162	0.362	0.386	0.112	0.2490
PE+KD	0.552	0.360	0.240	0.156	0.167	0.372	0.409	0.119	0.2641
PE+TD	0.537	0.341	0.225	0.148	0.163	0.359	0.371	0.114	0.2427
PE+PD	0.550	0.353	0.232	0.149	0.164	0.366	0.385	0.118	0.2514
PE+PD+AD	0.554	0.356	0.235	0.153	0.167	0.364	0.405	0.117	0.2609
PE+PD+AD+WD	0.578	0.381	0.258	0.171	0.176	0.384	0.444	0.123	0.2837
PE+PD+AD+WD+KD	**0.583**	**0.391**	**0.267**	**0.177**	**0.179**	**0.388**	**0.456**	**0.128**	**0.2920**

Table 2: Experimental results on the evaluation split of Clotho dataset.

Model	$BLUE_1$	$BLUE_2$	$BLUE_3$	$BLUE_4$	METEOR	ROUGE-L	CIDEr	SPICE	SPIDEr
Model 1: CNN14	0.583	0.388	0.265	0.178	0.179	0.385	0.473	0.128	0.300
Model 2: Resnet38	0.593	0.400	0.274	0.184	0.183	0.392	0.482	0.133	0.308
Model 3: Resnet38 + TD	0.581	0.386	0.261	0.173	0.178	0.384	0.456	0.131	0.294
Model 4: Wavegram-Logmel CNN	0.585	0.392	0.269	0.182	0.177	0.389	0.474	0.130	0.302
Ensemble 1 2 4	0.600	0.409	0.283	0.192	0.184	0.398	0.497	0.135	0.316
Ensemble 1 2 3 4	**0.603**	**0.414**	**0.286**	**0.195**	**0.186**	**0.400**	**0.499**	**0.137**	**0.318**

3. EXPERIMENTS

3.1. Dataset

The Clotho [2] v2 dataset consists of audio clips from the Freesound platform [15] with captions annotated via crowdsourcing [16]. The Clotho v2 dataset is divided into a development split of 3839 audio clips, a validation split of 1045 audio clips, an evaluation split of 1045 audio clips, and a test split of 1043 audio clips.

We used the development split of Clotho, AudioCaps and weakly labeled dataset for training, the evaluation split for testing. The validation split is selected as the validation data.

3.2. Data pre-processing

All audio clips are down-sampled to 32kHz. The configuration of audio feature extraction is the same as that of PANNs[8]. We use words in the development-training split of Clotho as vocabulary. Words out of vocabulary are represented by <UNK>. <SOS> and <EOS> are also employed as the start-of-sequence and end-of-sequence tokens, respectively.

3.3. Training detail

Training method As is shown in Figure 1, the whole training process is divided into three stages. In the pre-training stage, the parameters of encoder are frozen and only the decoder is trained. In the training stage, the encoder parameters are unfrozen and trained together with the decoder. In the experiment where AudioCaps and weakly labeled dataset are included, the finetuning stage is used to finetune the model only with the Clotho dataset.

Model settings We use CNN14 as an encoder for ablation study, which consists of 6 convolutional blocks and each convolutional block consists of 2 convolutional layers with a kernel size of 3×3. In addition to CNN14, Resnet38 and Wavegram-Logmel CNN are used as encoder in our submissions. Resnet38 consists of 16 basic blocks in the Resnet [17], where each block consists of two convolutional layers with a kernel size of 3×3, and a shortcut connection between input and output. Wavegram-Logmel-CNN uses CNN14 as a backbone and uses a trainable 1D-Conv based frontend to extract features from time-domain waveforms. We use a 2-layer Transformer [18] with a hidden dimension of 256 and 4 heads as decoder.

To improve performance and avoid over-fitting, Label smoothing [19] and SpecAugment [20] are applied during training. The configuration of SpecAugment is the same as that of PANN. The learning rate is 3e-4, 1e-4 and 5e-5 for the three training stages separately. In the inference stage, a beam search with beam size 3 is implemented to achieve better decoding performance.

3.4. Evaluation metrics

A total of eight objective metrics are utilized to evaluate our model generated captions. Among the metrics used, BLEU@1-4 [21] measures a modified n-gram precision. METEOR [22] measures a harmonic mean of precision and recall of segments of the captions between the predicted and the target. ROUGEL [23] measures F-score based on the longest common subsequence. CIDEr [24] measures a weighted cosine similarity of n-grams. SPICE [25] compares semantic propositions extracted from caption and reference. SPIDEr [26] is the arithmetic mean between the SPICE score and the CIDEr score.

4. RESULTS

4.1. Ablation study

To verify the effectiveness of the tricks and components in the proposed model, several ablation experiments are conducted.

The experiment results are shown in Tab.1. Baseline is the model training from scratch with the Clotho dataset.

As shown in Table 1, most of the tricks and components can improve the final SPIDEr score. First of all, the benefits of data augmentation are significant. Both the AudioCaps dataset and the weakly labeled dataset collected from the Internet can improve the final performance. And the collected weakly labeled dataset can bring more benefit than the AudioCaps dataset. The reason may be that the weakly labeled dataset is collected from websites similar to Freesound, which matches the data distribution of Clotho dataset more closely.

Secondly, adding a pre-trained encoder can significantly improve SPIDEr scores, which shows that the pre-trained PANN can extract more effective features from the audio. Thirdly, perturbing audio data slightly improves the spider score.

Finally, the keyword enhanced decoder can assist the generation of captions. Compared to experiments without keyword enhanced decoder, both experiments with this component get great improvement on all evaluation metrics. This indicates that similar audio captions contain valuable information for AAC task.

Note that compared to using PE only, by adding the tag enhanced decoder, the SPIDEr score drops a little bit. We thought this decline was due to the tag prediction errors produced by pre-trained PANN model.

4.2. Submitted systems

In Table 2, we present the relevant results of our submission 2, which is our best submission in the DCASE Challenge. To tackle the problem of insufficient data, validation data is added to the train dataset.

Three different model architectures of PANN are trained using the best strategy combination (PE+PD+AD+WD) obtained from ablation research. At the same time, we also try to add tag enhanced decoder to the Resnet architecture for training. Finally, model ensemble is performed by adding the predicted scores of multiple models. We first ensemble three different snapshots under the same model architecture and then we try two ensemble strategies, ensemble of three different architecture models and ensemble of all four architecture models.

Experiments show that Resnet achieves the highest spider score, and Resnet38 with tag enhanced decoder has the lowest score. Model ensemble can significantly improve SPIDEr scores. Although the SPIDEr score of the model with tag enhanced decoder declines, it improves the final result after the model ensemble.

4.3. Case study

We choose two cases from evaluation split of Clotho to show the impact of keyword enhanced decoder and tag enhanced decoder on captions generation. For each case, 2 representative reference captions are listed.

Table 3 shows the impact of keyword enhanced decoding. We can see that keyword enhanced decoding can make the caption more specific and closer to the caption written manually. The different forms of the ten keyword stems extracted from similar audio captions in the vocabulary is shown in the bottom of Table 3. Most of the keywords are in line with the content of the audio, which can help to generate captions more precisely.

Table 3: The case for Chopping pieces of mushrooms vigorously.wav.

Name	Caption
Ref 1	Vegetables are cut and chopped on a cutting board by someone.
Ref 2	A person cutting and chopping vegetables on a cutting board.
w/o KD	chopping vegetables with a knife.
KD	chopping vegetables **on a cutting board** with a knife.
keyword	knives/knife chopping/chopped/chop/chops vegetable/vegetables woods/wood **cutting**/cuts/cut saw/saws/sawed/sawing/ boards/**board** wooden food slices/sliced/slicing

Table 4: The case for SamyeLing_Pheasant121102.wav.

Name	Caption
Ref 1	A bird caws at regular intervals while smaller birds chirp in the background.
Ref 2	A bird making a call and another bird that is chirping.
w/o TD	a person uses a tool to each other.
TD	a bird is chirping and then another bird is chirping in the background.
Tag	TGA_animal

Table 4 shows an example of the advantage of the tag enhanced decoder. Guided by the tag, i.e., TGA_animal, captions about birds can be generated. Without this strategy, the generated captions are far from the reference captions. This case indicates that when the tag is accurate, the tag enhanced decoder can keep the generated caption within a reasonable space.

5. CONCLUSIONS

In this paper, we present a solution of automated audio captioning based on weakly supervised pre-training and word selection methods, and conducted a detailed ablation study to clarify which element is effective. From the results, pretrained encoder, keyword enhanced decoder and data augmentation are effective in improving the accuracy of AAC task. In particular, we propose a set of heuristic methods for collecting weakly-labeled data sets. This method can effectively alleviate the problem of insufficient data. We also verified that the captions of similar audio are valuable for the AAC task. In future work, we will explore the promotion of larger-scale data pre-training for AAC tasks, and try other effective methods to integrate similar audio captions information into AAC tasks.

6. REFERENCES

[1] K. Drossos, S. Adavanne, and T. Virtanen, “Automated audio captioning with recurrent neural networks,” in IEEE Workshop Appl. Signal Process, Audio Acoust, (WASPAA), 2017, pp. 374–378.

[2] K. Drossos, S. Lipping, and T. Virtanen, “Clotho: An audio captioning dataset,” in 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 736–740.

[3] Takeuchi, Daiki, et al. “Effects of word-frequency based pre- and post-processings for audio captioning,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2020, pp. 190-195.
[4] Chen, Kun, et al. “Audio Captioning Based On Transformer And Pre-trained CNN,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020), 2020, pp. 21-26.
[5] Perez-Castanos, Sergi, et al., “Listen carefully and tell: an audio captioning system based on residual learning and gammatone audio representation,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020), 2020, pp. 150-154.
[6] Xu, Xuenan, et al., “A crnn-gru based reinforcement learning approach to audio captioning,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2020, pp. 225-230.
[7] Gururangan S, Marasović A, Swayamdipta S, et al., “Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks,” in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), 2020, pp. 8342-8360.
[8] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang and M. D. Plumbley, “PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition,” in IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2020, vol. 28, pp. 2880-2894.
[9] Gemmeke, Jort F., et al., “Audio set: An ontology and human-labeled dataset for audio events,” in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 776-780.
[10] Yuan, Weiqiang, Han, Qichen, et al., “The DCASE 2021 Challenge Task 6 System: Automated Audio Captioning with Weakly Supervised Pre-training and Word Selection Methods,” DCASE2021 Challenge, Tech. Rep., 2021.
[11] Ko, Tom, et al. “Audio augmentation for speech recognition,” in Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH), 2015, pp. 3586-3589.
[12] Ko, Tom, et al., “A study on data augmentation of reverberant speech for robust speech recognition,” in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 5220-5224.
[13] C. D. Kim, B. Kim, H. Lee, and G. Kim, “Audiocaps: Generating captions for audios in the wild,” in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 119–132.
[14] Chen, Q., Zhu, X., Ling, Z. H., Wei, S., Jiang, H., & Inkpen, D, “Enhanced LSTM for Natural Language Inference,” in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2015, pp. 1657-1668.
[15] F. Font, G. Roma, and X. Serra, “Freesound technical demo,” in Int. Conf. Multimedia (MM’13), 2013, pp. 411-412.
[16] S. Lipping, K. Drossos, and T. Virtanen, “Crowdsourcing a dataset of audio captions,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2019, pp. 139–143.
[17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,”in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in neural information processing systems, 2017, pp. 5998–6008.
[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
[20] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “Specaugment: A simple data augmentation method for automatic speech recognition,” in Proc. Interspeech 2019, pp. 2613–2617, 2019.
[21] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002, pp. 311–318.
[22] S. Banerjee and A. Lavie, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” in Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, 2005, pp. 65–72.
[23] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in Text Summarization Branches Out. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81.
[24] R. Vedantam, C. L. Zitnick, and D. Parikh, “Cider: Consensus-based image description evaluation,” in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4566–4575.
[25] P. Anderson, B. Fernando, M. Johnson, and S. Gould, “Spice: Semantic propositional image caption evaluation,” in European Conference on Computer Vision. Springer, 2016, pp. 382–398.
[26] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, “Improved image captioning via policy gradient optimization of spider,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 873–881.

ENSEMBLE OF COMPLEMENTARY ANOMALY DETECTORS UNDER DOMAIN SHIFTED CONDITIONS

Jose A. Lopez, Georg Stemmer, Paulo Lopez-Meyer, Pradyumna S. Singh
Juan A. del Hoyo Ontiveros, Hector A. Courdourier

Intel Corporation
{jose.a.lopez, georg.stemmer, paulo.lopez.meyer, pradyumna.s.singh}@intel.com
{juan.del.hoyo.ontiveros, hector.a.cordourier.maruri}@intel.com

ABSTRACT

We present our submission to the DCASE2021 Challenge Task 2, which aims to promote research in anomalous sound detection. We found that blending the predictions of various anomaly detectors, rather than relying on well-known domain adaptation techniques alone, gave us the best performance under domain shifted conditions. Our submission is composed of two self-supervised classifier models, a probabilistic model we call NF-CDEE, and an ensemble of the three – the latter obtained the top rank in the DCASE2021 Challenge Task 2.

Index Terms— DCASE, anomaly detection, domain shift, machine condition monitoring, machine health monitoring.

1. INTRODUCTION

The DCASE2021 Challenge Task 2 is concerned with identifying anomalous behavior from a target machine using sound recordings [1]. A major difference between this task and other DCASE tasks is that it is not supervised. Accordingly, the available training data only contains samples from the normal-state distributions. A further complication added to this challenge is that the acoustic characteristics of the training data and of the test data are different – this condition is known as domain shift and there are some known results for reducing the performance gap between the training and test data [2, 3, 4, 5, 6, 7, 8]. In our experiments, while we recognize the potential of these techniques, we did not generally gain much from using these methods alone.

In our submission, we used two self-supervised classifiers that classified the section IDs similar to the approach several teams followed in DCASE2020 [9, 10, 11, 12, 13]. For a third model, we introduce a model that relies on several normalizing flows to estimate the conditional density of input Mel spectrogram sections and use their combined outputs to produce an anomaly score [14, 15, 16, 17, 18, 19, 20, 21, 22].

In the sequel we describe each model, how it was trained, its hyperparameters, and their respective results. In order to put the results into perspective, we include the scores for the baseline autoencoder and MobileNetV2 models on Tables 1 and 2, respectively. The data used in this challenge is 16 KHz, single-channel, audio. For more details, please see [1, 23, 24].

	ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
h-mean AUC	0.6249	0.6171	0.6324	0.6597	0.6192	0.6674	0.5341
h-mean pAUC	0.5236	0.5381	0.5338	0.5276	0.5441	0.5594	0.5054

Table 1: Baseline Autoencoder Scores

	ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
h-mean AUC	0.5604	0.5746	0.6156	0.6670	0.6189	0.5926	0.5651
h-mean pAUC	0.5637	0.5161	0.6302	0.5916	0.5737	0.5600	0.5264

Table 2: Baseline MobileNetV2 Scores

2. ARCHITECTURES

The first model described below builds on the work from [9]. In particular, the encoder network has been updated to use 1D convolutions rather than 2D. The input to this model is a spectrogram with or without a Mel transformation. The second model builds on the well-known WaveNet architecture [25] by adding an x-vector [26] classification head after the dilated convolutions – in a sense, the WaveNet functions as a time-series encoder for the x-vector component. Both models mentioned above are trained to reduce the cross entropy loss between predictions and the section IDs. The third model differs from the first two models in that it is completely unsupervised and attempts to learn several distributions of some Mel spectrogram bins conditioned on the remaining bins. We also describe a fourth model, a 1D convolutional autoencoder, which we did not include in our submission but we believe may be of interest to the community. We call these approaches complementary because of the different input modalities and learning approaches. The last system described is an ensemble of the first three models described above.

All our development was done using PyTorch [27] and spectrograms were computed using nnAudio [28]. The third model additionally used the Pyro [29] probabilistic programming library.

2.1. XVector1D

A high-level view of the architecture of the first model is shown in Figure 1. We denote additive margin softmax as AMS [30].

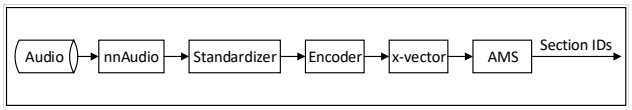


Figure 1: XVector1D High-level Architectures

In Figure 1, we use the term “standardizer” as a preprocessing step done before passing data to the rest of the network. In the simplest case, it is a batch-norm layer with the learnable parameters disabled. In this way, this batch-norm will perform the usual

ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
STFT	MEL	STFT	MEL	STFT	STFT	STFT
AutoDIAL	batch-norm	AutoDIAL	AutoDIAL	AutoDIAL	AutoDIAL	AutoDIAL
C(128,192)	C(128,192)	C(128,192)	C(128,192)	C(128,192)	C(128,192)	C(128,192)
5 x C(192,192)	5 x C(192,192)	5 x C(192,192)	4 x C(192,192)	5 x C(192,192)	5 x C(192,192)	5 x C(192,192)
			C(192,90)			

Table 3: Input And Encoder Parameters

frequency-wise normalization once the running statistics have converged. However, in other cases, "standardizer" can mean an AutoDIAL layer which mixes the statistics from the source and target domains for normalization [4]. In early experiments we evaluated the more general domain adaptation technique from [6]; however, we found the performance similar to AutoDIAL but, in our implementation, much more computationally expensive.

The encoder used in this model includes 1D convolutions with kernel size 3 and leaky-relu activations. The number of layers varied with machine type as shown on Table 3 – in this table and going forward, we use "C" to mean a 1D convolution.

The x-vector component used here remains largely the same as in [9] except the interface to the encoder was adapted (as expected) to accept the 1D encoder output.

2.1.1. Preprocessing

This model did not use any special preprocessing or augmentation. The logarithm was taken for both the STFT and the Mel spectrograms. All spectrograms were computed with frequency min and max values set to 100 and 8000 Hertz, respectively. Mel spectrograms were computed using 128 bins.

2.1.2. Training & Results

The model was trained to predict the section ID meta-data parameter using the cross entropy loss function. We found that the spectrogram parameters had a big effect on the performance. Parameters like the number of input samples, the number of points used for the FFT, the hop length can have a significant effect. We generally used the AdamW optimizer with the default learning rate of 1×10^{-3} and weight decay set to 1×10^{-4}. However, we used ASGD with the default learning rate (and no weight decay) for gearbox. Generally, the training losses converge more slowly using ASGD but sometimes the slower trajectory spends more epochs close to an optimal region with respect to AUC and this can yield better results[1]. During training, random contiguous audio clips were sampled and the spectrograms were computed on the fly using nnAudio [28]. The training was usually run for 300 epochs, using all the training data from the development and evaluation datasets. Lastly, we computed the average embedding, during training, using the embedding from the layer prior to the final AMS classification layer. At test time, the average embedding was used to compute the cosine and Mahalanobis distances to the test embedding which served as additional options for anomaly scores. Table 4 shows the results, and, Table 5 shows the effect on performance when using AutoDIAL.

2.2. WaveNet-XVector

We explored the use of a WaveNet model processing the audio samples directly. For details on the architecture we refer the reader to the original publication [25]. In the original paper the authors explain that the model can be readily adapted to classification tasks and in their classification experiment they add a mean pooling layer after the dilated convolutions followed by "a few non-causal convolutions". The training proceeds with two loss terms: one for predicting the next sample and the other is the classification loss. We follow this procedure in that we use a mean pooling layer (with kernel size 10) and train with the two loss functions but instead of using a few convolutions, we use an x-vector component, with AMS top layer, as with the XVector1D model. In this way, one can consider this model a variant of the XVector1D model which uses an audio-only encoder. For the WaveNet encoder, we used a single block with 14 layers. Gearbox and pump used 64 channels for the dilation, residual, and skip channels. The other machines used 32 channels. For valve we used an AutoDIAL standarizer, and the rest used a batch-norm.

[1]We used the harmonic mean of AUC and pAUC harmonic means to assign a single score to a model configuration. For gearbox, the experiment using ASGD had an 8.89% greater score.

	ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
batch size	128	64	128	64	128	128	64
input samples	16384	16384	16384	98000	16384	16384	98000
no. Mels	2048	128	2048	128	2048	2048	2048
no. FFT	4096	1024	4096	1024	4096	4096	4096
hop	80	512	512	80	512	512	512
scoring	cosine	mahalanobis	softmax	mahalanobis	softmax	softmax	softmax
h-mean AUC	0.6702	0.7193	0.7171	0.8342	0.7799	0.7871	0.9032
h-mean pAUC	0.6233	0.6772	0.7295	0.7443	0.6684	0.6728	0.7724

Table 4: XVector1D Scoring Results

	ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
AutoDIAL	2.79%	-12.90%	4.49%	23.41%	2.24%	1.31%	0.46%

Table 5: Relative Change In XVector1D Score Using AutoDIAL

2.2.1. Preprocessing

For valve and ToyTrain we used the Teager-Kaiser energy operator (TKEO) to preprocess the audio [31, 32, 33, 34]. The motivation was that, because the valve noises are sparse and impulsive events, the noise suppression provided by the Teager-Kaiser operator would improve the signal-to-noise (SNR) ratio in the valve recordings. Despite improving the results for valve and ToyTrain, the improvement was modest.

2.2.2. Training & Results

To train this model, we used the Adamax optimizer with the default learning rate for 200 epochs in the same manner as XVector1D, with 16384 input samples. Table 6 shows the performance of this model using softmax scoring. Table 7 shows the effects on performance due to AutoDIAL and TKEO independently.

	ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
batch size	128	128	128	64	64	128	128
h-mean AUC	0.5843	0.6641	0.8122	0.7156	0.7543	0.7184	0.7297
h-mean pAUC	0.5629	0.5696	0.8025	0.5964	0.6506	0.6239	0.6206

Table 6: WaveNet-XVector Scoring Results

2.3. NF-CDEE

For our third system, we began by attempting to model the probability density function of the Mel spectrograms of the machine sounds, for a single machine, using normalizing flows. We used the Pyro [29] probabilistic programming library to develop this model. We

	ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
AutoDIAL	-0.54%	-0.11%	-3.13%	-2.62%	-1.71%	-2.22%	4.97%
TKEO	-3.15%	7.69%	-27.04%	-5.98%	-9.70%	-14.08%	3.30%

Table 7: Relative Change On WaveNet-XVector Score Using AutoDIAL & TKEO

found that training a model to fit a distribution with the same dimensions as Mel bins to be somewhat unstable. In order to improve the stability we instead estimated several conditional densities and trained them in a single model, minimizing the sum of their negative log-likelihoods. We consider this model an ensemble of conditional density anomaly detectors. Hence, we call this model NF-CDEE, because it uses normalizing flows and it is a conditional density estimator ensemble. Each conditional density estimator fits the distribution of a n-bin segment of input spectrograms conditioned on the remaining bins. This reduces the instability due to dimensionality. The parameter n and the amount of overlap are tunable by the user. For this work, we chose $n = 32$ with no overlap. Each normalizing flow uses a single conditional spline with 16 count-bins and the default hidden layer dimensions – these are also tunable but in our experiments they did not significantly affect the performance.

To summarize, each estimator outputs the probability $p(s_A|s_{A^c})$ where s is a vector of dimension equal to the number of Mel bins m that is indexed by the set $\mathcal{I} = \{1, \ldots, m\}$. A is an n-element subset of $\mathcal{I}$, and A^c is its complement $\mathcal{I} - A$. We define the likelihood of the normal state as:

$$p(\text{normal}) = \prod_i p(s_{A_i}|s_{A_i^c}) \tag{1}$$

where $i \in [1, \ldots, k]$ and k is a positive integer provided by the user – it is the number of estimators in the ensemble. Here, we used $A_1 = \{1, \ldots, 32\}$, $A_2 = \{33, \ldots, 64\}$, and so forth. To train the model, we minimize the negative logarithm of $p(\text{normal})$. Therefore, the output of NF-CDEE is the sum of the individual negative log-likelihoods.

2.3.1. Training & Results

To train this model we converted the input audio to 256-bin Mel spectrograms, computed using 8192-point FFTs with hop-length 512, and applied frequency-wise normalization before passing to the conditional density estimators. Unlike the self-supervised models, the spectrograms were pre-computed and spectrogram windows were passed to the network in the same manner as [35]. Each model was trained with all the sections of the development (or evaluation) training data, per machine type – except for fan for which we trained a model for each section. To further reduce training instability, caused by the normalizing flow determinant computation, we take the mean across the time dimension. This last step was important for stabilizing the training of the ensemble. As previously stated the loss function used was the sum of the negative log-likelihoods and this also served as the anomaly score. Figure 2 shows the inference process.

For the optimizer, we used the same optimizer as the XVector1D, with gradient clipping. In our experiments this model generally needs to train for about 50 epochs. Table 8 shows the results, sampling 192 spectrogram frames in batches of 32.

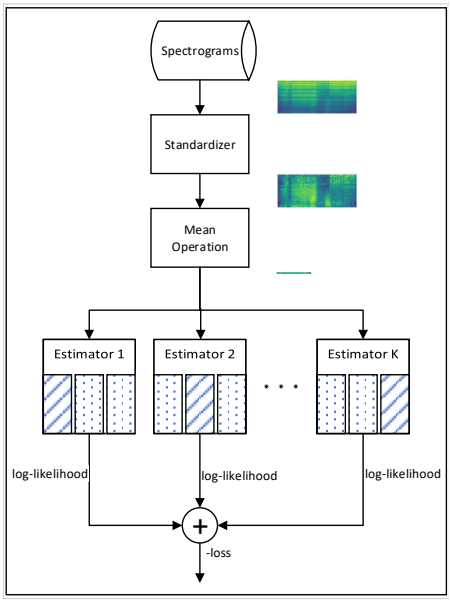


Figure 2: NF-CDEE Inference

	ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
h-mean AUC	0.8657	0.7797	0.7866	0.8081	0.6993	0.7483	0.6130
h-mean pAUC	0.7831	0.6031	0.6024	0.6513	0.5655	0.6054	0.5275

Table 8: NF-CDEE Scoring Results

2.4. 1D CNN Autoencoder

The model described here is a 1D convolutional autoencoder that reconstructs (Mel) spectrograms. We excluded this model for several reasons including that, like NF-CDEE, its performance was strongest for ToyCar but NF-CDEE was also strong for other machines. Additionally, WaveNet-XVector offered stronger fan performance than either XVector1D or NF-CDEE and resulted in a better ensemble when including three models.

The architecture of this model is shown in Table 9. The bottleneck for this autoencoder was inspired by [36] in that the time dimension was mostly preserved[2]. In our post-DCASE2020 analyses, we found that preserving the time dimension to be a key factor for the success of the autoencoder in [36]. Additionally, we found the scoring methods in [36] to be effective at improving AUC performance in (spectrogram) autoencoders. For example, the scoring methods in [36] can improve the results from [35]. Table 9 shows the architecture of this model, which uses leaky-relu activations, kernel size 3, and a batch-norm standardizer.

[2]In [36] the time dimension was not reduced at all because causal convolutions were used.

ToyCar	ToyTrain	fan	gearbox	pump	slider	valve
C(128,256)	C(128,192)	C(128,256)	C(128,256)	C(128,192)	C(128,192)	C(128,192)
3 x C(256,256)	4 x C(192,192)	2 x C(256,256)	5 x C(256,256)	3 x C(192,192)	5 x C(192,192)	3 x C(192,192)
C(256,20)	C(192,30)	C(256,10)	C(256,30)	C(192,90)	C(192,90)	C(192,90)
C(20,256)	C(30,192)	C(10,256)	C(30,256)	C(90,192)	C(90,192)	C(90,192)
3 x C(256,256)	4 x C(192,192)	2 x C(256,256)	5 x C(256,256)	3 x C(192,192)	5 x C(192,192)	3 x C(192,192)
C(256, 128)	C(192, 128)	C(256, 128)	C(256, 128))	C(192, 128)	C(192, 128)	C(192, 128)

Table 9: 1D CNN Autoencoder

2.4.1. Training & Results

The training for this model followed the same approach as NF-CDEE, using spectrogram windows as in [35]. The loss function used was mean absolute error (MAE). However, for gearbox, fan, and ToyCar we attached an x-vector classifier head to the bottleneck layer and additionally included a cross-entropy loss term for the meta-data. Moreover, for gearbox, fan, and ToyCar we employed a special weighting scheme during training and at test time.

The intuition behind the weighting is that if one knew the SNR of a frequency bin, one could weigh the reconstruction loss using this information, giving greater importance to bins with greater SNR. Estimating the SNR is not straightforward, so we used the frequency bin variance in its place. The weight vector w is precomputed with elements given by $w_f = \frac{1}{\sigma_f^2}$, where σ_f^2 denotes the variance of the fth Mel bin[3], using the training data for each domain and each section.

Table 10 shows the results sampling 192 frames, from 128-bin Mel spectrograms, in batches of 64. The spectrogram hop length was set to 512. The E1 and E2 scoring methods referenced in Table 10 come from [36] and are repeated here for the reader's convenience.

$$E1(X, \hat{X}) = \frac{1}{FT} \sum_{f=1}^{F} \left[\sum_{t=1}^{T} \left(X_{f,t} - \hat{X}_{f,t} \right) \right]^2 \tag{2}$$

and

$$E2(X, \hat{X}) = \frac{1}{FT} \sum_{f=1}^{F} \left| \sum_{t=1}^{T} \left(X_{f,t} - \hat{X}_{f,t} \right) \right| \tag{3}$$

where $X \in \mathbb{R}^{F \times T}$ and $\hat{X} \in \mathbb{R}^{F \times T}$ are the true and reconstructed spectrograms. F and T are natural numbers that denote the frequency and time dimensions, respectively.

	ToyCar	**ToyTrain**	**fan**	**gearbox**	**pump**	**slider**	**valve**
scoring	E2	MAE	E1	E1	E1	E2	E2
no. FFT	8192	2048	8192	4096	2048	8192	4096
h-mean AUC	0.8663	0.7180	0.7265	0.7287	0.6981	0.7022	0.6117
h-mean pAUC	0.7502	0.6023	0.5738	0.5992	0.5951	0.5782	0.5230

Table 10: 1D CNN Autoencoder Scoring Results

2.5. Ensemble

For the last system we combined the first three models (described in Sections 2.1, 2.2, and 2.3) by first standardizing the training data scores and then searching over a grid of convex combinations, similar to [37].

We could have included the autoencoder, for example, by ensembling separately with each system but we did not have time to explore this or other ensembling alternatives. As it stands, this model influenced the development of XVector1D, particularly its encoder, and the selection of hyperparameters for NF-CDEE. Table 11 shows the results of the ensemble of the first three models.

[3]The weight vector was also scaled to have a max element of 1.

	ToyCar	**ToyTrain**	**fan**	**gearbox**	**pump**	**slider**	**valve**
WaveNet weight	0.03	0.03	1	0.04	0.32	0.02	0
XVector1D weight	0.06	0.55	0	0.61	0.68	0.52	1
NF-CDEE weight	0.91	0.42	0	0.35	0	0.46	0
h-mean AUC	0.8745	0.7756	0.8122	0.8613	0.7958	0.8287	0.9032
h-mean pAUC	0.7837	0.7048	0.8025	0.7635	0.6790	0.6925	0.7724

Table 11: Ensemble Scoring Results

3. CONCLUSIONS

We have outlined our submission to the DCASE2021 Challenge Task 2, which featured a domain shift between the training and test distributions. We found it concerning that domain adaptation methods that seem to do well for other modalities, especially vision, do not seem to work as well for audio (at least in our implementations). This discrepancy gives the DCASE2021 Challenge a greater relevance, because it highlights the need for the audio community to generate more effective domain adaptation methods for audio.

As the XVector1D, WaveNet-XVector, and NF-CDEE models (respectively) ranked 11, 52, and 31, it is clear that the three were indeed complementary and that ensembling is a good option for improving results under domain shifted testing conditions. We do not find the lower individual ranks too concerning because the scores are for single models, as opposed to ensembles, and because the rankings do not fully reflect the performance on individual machine categories.

Of the models we investigated, we find NF-CDEE to be particularly promising because it performed well and is unsupervised. In real-world settings it is not always practical to leverage meta-data, even when it is possible to do so. Moreover, we expect the ensembling nature of the model to perform better under domain shifted conditions. We plan to develop this model further going forward.

4. REFERENCES

[1] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and discussion on dcase 2021 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring under domain shifted conditions," *arXiv preprint arXiv:2106.04492*, 2021.

[2] G. Wilson and D. J. Cook, "A survey of unsupervised deep domain adaptation," 2020.

[3] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, 2018.

[4] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulò, "Autodial: Automatic domain alignment layers," 2017.

[5] ——, "Just dial: Domain alignment layers for unsupervised domain adaptation," 2017.

[6] M. Mancini, L. Porzi, S. R. Bulò, B. Caputo, and E. Ricci, "Boosting domain adaptation by discovering latent domains," 2018.

[7] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein distance guided representation learning for domain adaptation," 2018.

[8] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," 2015.

[9] J. A. Lopez, H. Lu, P. Lopez-Meyer, L. Nachman, G. Stemmer, and J. Huang, "A speaker recognition approach to anomaly detection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 96–99.

[10] R. Giri, S. V. Tenneti, F. Cheng, K. Helwani, U. Isik, and A. Krishnaswamy, "Self-supervised classification for detecting anomalous sounds," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 46–50.

[11] T. Inoue, P. Vinayavekhin, S. Morikuni, S. Wang, T. Hoang Trong, D. Wood, M. Tatsubori, and R. Tachibana, "Detection of anomalous sounds for machine condition monitoring using classification confidence," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 66–70.

[12] P. Primus, V. Haunschmid, P. Praher, and G. Widmer, "Anomalous sound detection as a simple binary classification problem with careful selection of proxy outlier examples," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 170–174.

[13] Q. Zhou, "Arcface based sound mobilenets for dcase 2020 task 2," DCASE2020 Challenge, Tech. Rep., July 2020.

[14] E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164.

[15] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," 2016.

[16] I. Kobyzev, S. Prince, and M. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2020.

[17] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," 2021.

[18] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows," 2019.

[19] H. M. Dolatabadi, S. Erfani, and C. Leckie, "Invertible generative modeling using linear rational splines," 2020.

[20] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," 2015.

[21] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," 2017.

[22] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," 2016.

[23] R. Tanabe, H. Purohit, K. Dohi, T. Endo, Y. Nikaido, T. Nakamura, and Y. Kawaguchi, "MIMII DUE: Sound dataset for malfunctioning industrial machine investigation and inspection with domain shifts due to changes in operational and environmental conditions," *In arXiv e-prints: 2006.05822, 1-4*, 2021.

[24] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, "ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions," *arXiv preprint arXiv:2106.02369*, 2021.

[25] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," 2016.

[26] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[28] K. W. Cheuk, H. Anderson, K. Agres, and D. Herremans, "nnaudio: An on-the-fly gpu audio to spectrogram conversion toolbox using 1d convolution neural networks," 2020.

[29] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, pp. 28:1–28:6, 2019.

[30] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, p. 926–930, Jul 2018.

[31] P. Maragos, J. Kaiser, and T. Quatieri, "Energy separation in signal modulations with application to speech analysis," *IEEE Transactions on Signal Processing*, vol. 41, no. 10, pp. 3024–3051, 1993.

[32] P. Maragos, J. F. Kaiser, and T. F. Quatieri, "On amplitude and frequency demodulation using energy operators," *IEEE Transactions on Signal Processing*, vol. 41, no. 4, pp. 1532–1550, Apr. 1993.

[33] A. Georgogiannis and V. Digalakis, "Speech emotion recognition using non-linear teager energy based features in noisy environments," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, 2012, pp. 2045–2049.

[34] H. Li, H. Zheng, and L. Tang, "Gear fault detection based on teager-huang transform," *International Journal of Rotating Machinery*, vol. 2010, pp. 1–9, 2010.

[35] A. Ribeiro, L. Matos, P. Pereira, E. Nunes, A. Ferreira, P. Cortez, and A. Pilastri, "Deep dense and convolutional autoencoders for unsupervised anomaly detection in machine condition sounds," DCASE2020 Challenge, Tech. Rep., July 2020.

[36] V. K. Agrawal and S. S. Maurya, "Unsupervised detection of anomalous sounds for machine condition monitoring," DCASE2020 Challenge, Tech. Rep., July 2020.

[37] P. Daniluk, M. Gozdziewski, S. Kapka, and M. Kosmider, "Ensemble of auto-encoder based systems for anomaly detection," DCASE2020 Challenge, Tech. Rep., July 2020.

SQUEEZE-EXCITATION CONVOLUTIONAL RECURRENT NEURAL NETWORKS FOR AUDIO-VISUAL SCENE CLASSIFICATION

Javier Naranjo-Alcazar[1,2]*, Sergi Perez-Castanos*[2]*, Aaron Lopez-Garcia*[2]*, Pedro Zuccarello*[1]*, Maximo Cobos*[2]*, Francesc J. Ferri*[2]

[1] Instituto Tecnológico de Informática, València, Spain {jnaranjo, pzuccarello}@iti.es
[2] Universitat de València, Burjassot, Spain, {pecaser, aaron.lopez, maximo.cobos, francesc.ferri}@uv.es

ABSTRACT

The use of multiple and semantically correlated sources can provide complementary information to each other that may not be evident when working with individual modalities on their own. In this context, multi-modal models can help to produce more accurate and robust predictions in machine learning tasks where audio-visual data is available. This paper presents a multi-modal model for automatic scene classification that exploits simultaneously auditory and visual information. The proposed approach makes use of two separate networks which are respectively trained in isolation on audio and visual data, so that each network specializes in a given modality. The visual subnetwork is a pre-trained VGG16 model followed by a bidiretional recurrent layer, whilst the residual audio subnetwork is based on stacked squeeze-excitation convolutional blocks trained from scratch. After training each subnetwork, the fusion of information from the audio and visual streams is performed at two different stages. The early fusion stage combines features resulting from the last convolutional block of the respective subnetworks at different time steps to feed a bidirectional recurrent structure. The late fusion stage combines the output of the early fusion stage with the independent predictions provided by the two subnetworks, resulting in the final prediction. We evaluate the method using the recently published TAU Audio-Visual Urban Scenes 2021, which contains synchronized audio and video recordings from 12 European cities in 10 different scene classes. The proposed model has been shown to provide an excellent trade-off between prediction performance (86.5%) and system complexity (15M parameters) in the evaluation results of the DCASE 2021 Challenge.

Index Terms— Deep Learning, Multi-modal, Convolutional Neural Networks, Scene Classification, Squeeze-Excitation, Gammatone, DCASE 2021

1. INTRODUCTION

The world as it is perceived by humans involves multiple modalities. In general, a sensory modality is understood as a primary channel of communication and sensation, such as vision, hearing or touch. In this context, multi-modal machine learning aims at exploiting datasets including multiple such modalities, building models that can process and relate information among them [1]. The explosion of deep learning and its use in vision, natural language processing and acoustic analysis, makes of multi-modal machine learning a multi-disciplinary area with increasing potential. Obviously, the most abundant multi-modal datasets are those made up of audio-visual data, where the included examples come in the form of videos that include both sound and images.

One important application scenario of multi-modal machine learning is automatic scene classification on videos, which refers to the task of classifying audiovisual data to one of the predefined scene categories (such as airport, park or shopping mall) [2], based on the ambient content provided by the information contributed by both modalities. Note, however, that scene classification has also been a topic of intensive research in the last decade by considering different modalitites on their own [3, 4, 5].

This paper presents a multi-modal approach for scene classification consisting of two specialized components or modules (an audio module and a visual module) that are further trained together to achieve a more robust solution by incorporating two fusion strategies simultaneously. The visual module is based on a VGG16 [6] convolutional neural network (CNN) pre-trained on the *places365* dataset [7, 8]. The training procedure of this component is based on a transfer learning scheme with fine tuning. On the other hand, the audio module is based on a fully convolutional neural network with convolutional blocks implementing residual and squeeze-excitation techniques [9] and gammatone filterbank audio representations as input. Finally, the audio and video modules with frozen weights are combined into a multimodal recurrent structure that performs information fusion both at early and late stages. The early fusion stage combines features resulting from the last convolutional block of the audio and visual subnetworks at different time steps, while the late fusion stage provides a final prediction by combining the output of the early fusion stage with the independent predictions provided by the visual and audio modules.

The model is evaluated by considering the TAU Audio-Visual Urban Scenes 2021 dataset [10], which contains synchronized audio and video recordings from 12 European cities in 10 different scenes classes. For a complete assessment of the model, different input pre-processing alternatives and architecture choices are considered and discussed.

The rest of this paper is structured as follows. Section 2 describes in detail the architecture of the different sections making up the whole learning system. Section 3 describes the experimental set-up and evaluates the proposed system considering some variants with respect to the input and the system architecture. Section 4 compares our system with other Challenge submissions. Finally, Section 5 concludes this work.

2. SYSTEM DESCRIPTION

This section describes the full architecture of the system, providing details on the different modules making up the whole multi-modal network. An schematic view of the full model is depicted in Fig. 1, where the visual and audio flows are represented with different col-

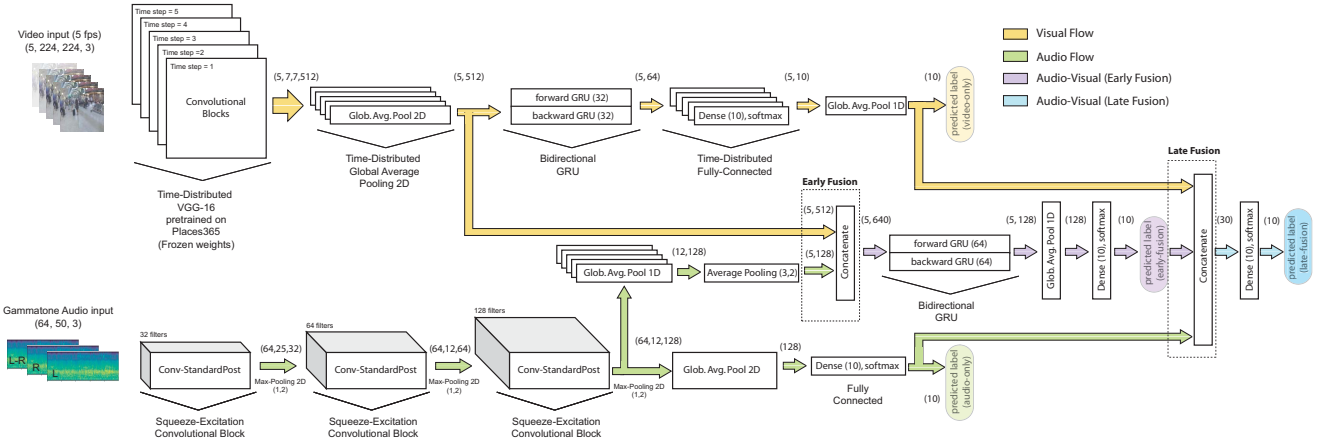


Figure 1: Proposed network achitecture for audiovisual scene classification.

ors.

2.1. Audio Module

2.1.1. Audio Input Representation

Based on previous works by the authors [11, 12], the input to the network consists is a multi-channel 3D audio representation compiling information from the left and right input audio channels as well as their difference. Each channel is converted into a time-frequency domain representation, provided by a Gammatone or a Mel-scale filter bank. Both alternatives have been widely adopted by the machine listening community [13, 14, 15, 16] and we evaluate both options in the experimental section of this paper.

All the considered representations are computed using 64 frequency bands, with a window size of 40 ms and 50% overlap. The audio was resampled from 48 kHz to 44.1 kHz. Gammatone representations were computed by using the Auditory Toolbox presented in [17] with Python implementation and Mel-spectrograms were obtained by using the LibRosa library [18]. Taking the above details into account, one second of audio results in a tensor input of size (64, 50, 3), where the third axis corresponds to the left-right-difference channels.

2.1.2. Audio Subnetwork

The audio module is based on a fully convolutional neural network combining residual connections with squeeze-excitation. More specifically, the convolutional blocks follow the structure of those denoted as *Conv-StandardPOST* in [9] (see Fig. 2), which showed very good performance for acoustic scene classification tasks. The aim of these blocks is to achieve improved accuracy by recalibrating the internal feature maps using residual [19] and squeeze-excitation techniques [20, 21]. An important feature is the use of the scSE (spatial and channel Squeeze and Excitation) module, which performs a spatial and channel-wise recalibration of the block feature maps. The interested reader is referred to [9] for a full description and evaluation of such blocks. All use a 3×3 kernel size, while the number of filters in each block are specified in Fig. 1. In between convolutional blocks, Max Pooling layers are used to halve the resolution of the resulting feature maps along the time axis. Additionally, Dropout [22] with a rate of 0.3 is also included after

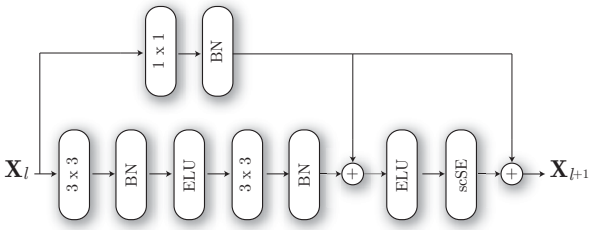


Figure 2: Structure of the *Conv-StandardPOST* block [9]. BN and ELU denote batch normalization and exponential linear unit activation, respectively. The $N \times N$ notation denotes a convolutional layer with the corresponding kernel size. The input to the l-th block is denoted as $\mathbf{X}_l$.

the pooling layers to prevent overfitting. The output feature maps from the last convolutional block are summarized with global average pooling into a 128-dimensional feature vector, which is fed to a fully-connected layer with softmax activation for classification. This subnetwork is trained from scratch using only audio data on the whole dataset by minimizing the cross-entropy loss.

2.2. Visual Module

2.2.1. Visual Input Representation

The visual input is adapted to match the pre-trained VGG16 architecture [6], which accepts color images of size 224×224 pixels. Moreover, as visual scene recognition does not require a very high frame rate (images do not change that much from frame to frame), the videos from the dataset are subsampled for obtaining a frame rate of 5 frames per second (fps). Therefore, a one-second video clip results in a tensor shape of (5, 224, 224, 3).

2.2.2. Visual Subnetwork

The visual module is based on the VGG16 CNN architecture [6] pretrained on the *places365* dataset [7]. With the aim of processing temporal information extracted from multiple frames, a time-distributed structure with frozen weights is considered. It must be

emphasized that the pre-trained model is used as a feature extractor, so that the top fully-connected layers of the network are omitted. The outputs from each time step (5 temporal steps) are globally averaged channel-wise, resulting in a sequence of 512 output features. This sequence is fed to a bidirectional 64-neuron Gated Recurrent Unit (GRU) layer and the returned sequences are processed by a time-distributed fully-connected layer with softmax activation, resulting in a predicted label for each time step. The final label is taken as the temporal average of the predictions. This subnetwork is trained on the visual data only, with trainable weights only on the recurrent and final dense layer.

2.3. Full Audio-Visual Network

The complete audio and visual modules described above are then merged into a full audio-visual framework that combines information from both modalities at two different levels. On an early fusion stage, the output of the last convolutional block of the audio and visual modules are concatenated into a sequence of 640 features. To achieve this, the feature maps of the audio module are turned into a temporal sequence matching the temporal resolution of the visual data (i.e. 5 fps) using global and average pooling operators. A bidirectional GRU processes the sequence and a new prediction is created by stacking a global average pooling and a dense layer. A late fusion stage receives the predictions from the independent modalities as well as the one resulting from their combination and produces the final prediction with a dense layer with softmax activation.

Note that, as observed in Fig. 1, the full network can be used to extract both predictions from the independent modalitites, i.e. labels from the visual (yellow) and audio (green) information flows, and from the fusion flows, i.e. early fusion (purple) and late fusion (blue).

2.4. Dataset

The system is trained on the recently published TAU Urban Audio-Visual Scenes 2021 [10]. This dataset contains fragments of recordings obtained in 12 large European cities corresponding to 10 scene classes: airport, shopping mall (indoor), metro station (underground), pedestrian street, public square, street (traffic), traveling by tram, bus and metro (underground), and urban park. The data was gathered with four devices recording simultaneously. The data examples are provided as segments with a length of 10 seconds, annotated by the corresponding scene class, city and recording location identifier. The dataset contains 34 hours of recordings and it specifies training/test partitions to facilitate comparisons. The training set contains approximately 70% of the data, while the validation set contains the remaining 30%.

2.5. Training Details

The whole network was trained in three steps using the default training and validation partitions provided by the TAU Audio-Visual Urban Scenes 2021 dataset. The first step corresponds to the training of the audio module from scratch using audio data only. The second step trains the recurrent and classification parts of the visual module (the convolutional blocks use frozen weights from the pre-trained network). In the last step, the whole audio-visual network is trained using frozen weights from the audio and visual modules. A fine-tuning strategy is finally followed, unfreezing all the weights and using a very small learning rate. The loss function used at each training step was categorical cross-entropy. The optimizer used was Adam [23] with default parameters. The models were trained with a maximum of 200 epochs. Batch size was set to 32 for training the independent subnetworks and 16 for the complete audio-visual network due to memory constraints. The 10 second examples provided in the dataset were randomly trimmed into 1 second segments in each epoch. The learning rate started with a maximum value of 0.001 decreasing with a factor of 0.5 in case of no improvement in validation accuracy after 20 epochs. In the last fine-tuning with all trainable weights, the starting learning rate was 10^{-5}. The training is considered as early finished in case of no improvement in validation accuracy after 50 epochs. Mixup data-augmentation [24] with $\alpha = 0.4$ was used. All the models were implemented using Keras with Tensorflow backend and trained using NVidia Titan RTX GPU.

2.6. Model Complexity

Assuming that all the weights of the different subsystems are trainable, the number of parameters corresponding to the different modules of the network are as follows: audio module (323k trainable weights), visual module (14M parameters, only 105k trainable) and full audio-visual system (15M parameters, only 272k trainable). Note that, although the number of parameters used by the visual module is considerably higher than that of the audio module, the visual one uses frozen weights in all its convolutional blocks. Thus, all subsystems can be trained considerably fast, as only 272k weights are trainable. Additionally, the final fine-tuning step in which all the weights are unfrozen only requires 2 epochs and, therefore, it does not require too much extra training time.

3. EXPERIMENTS

This section evaluates the proposed multi-modal framework over the default validation partition of the TAU Audio-Visual Urban Scenes 2021 dataset. For those systems submitted to the DCASE 2021 Challenge, we also provide the performance reported by the organizers over the evaluation/test dataset. Note that no custom test partition was created in order to facilitate comparisons with other competing systems using the same dataset. The performance of the proposed system is analyzed considering three different aspects:

- Audio input representation: log-Mel spectrogram and gammatone filterbank.
- Independent modalities: audio-only and visual-only.
- Multi-modal fusion: early fusion and late fusion.

Table 1 shows the accuracy results obtained for the different subsystems involved in the audio-visual framework on the default validation set. Similarly, Table 2 shows the results obtained for the evaluation partition used in DCASE 2021 Task1b. Comparisons to other competing approaches can be directly accessed via the DCASE 2021 Challenge website[1].

In general, the results clearly highlight that the visual-only modality is much more accurate than the audio-only one (e.g. 87.0% vs 69.0% in the development validation partition). Although it is true that the visual network departs from previous knowledge provided by a pre-trained model, this confirms that, as of today, acoustic scene recognition is a more challenging problem than visual scene recognition. Nonetheless, the audio module used in this

[1] http://dcase.community/challenge2021/task-acoustic-scene-classification-results-b

	Modality			
	Audio-Only	**Visual-Only**	**Multi-Modal (Early Fusion)**	**Multi-Modal (Late Fusion)**
log-Mel	68.4	87.0	88.5	88.7
Gammatone	69.0	87.0	89.2	90.0

Table 1: Accuracy results on the TAU Audio-Visual Urban Scenes 2021 validation partition.

Modality		
Audio-Only (Gammatone)	**Visual-Only**	**Multi-Modal (Late Fusion)**
66.8	83.2	86.5

Table 2: Accuracy results on the DCASE 2021 Task1b evaluation set.

work was ranked as the best performing model from all the submissions considering only the audio modality in terms of log-loss performance. In addition, it is observed that although both audio input representations, log-Mel and Gammatone spectrograms, led to very similar performances, the results were slightly better with the use of Gammatone filterbanks (69.0% vs 68.4% in the validation partition).

Despite the fact that the multi-modal models provide the best performance, their accuracy is only slightly better than the best of the individual modalities, which is particularly the visual one. In any case, for our specific case and despite the significant performance gap between the audio and video modalities, the multi-modal approaches achieved a performance gain of approximately 3 percentage points. In this context, although both the early and late fusion stages were able to exploit the information from the audio and visual data, the late fusion stage performed consistently (slightly) better in all our experiments.

In order to provide further insight, Fig. 3 shows the performance achieved by each modality and their combination in the DCASE 2021 Task1b evaluation set on each class. Note that the audio module only outperforms the visual one for the tram class and it can be clearly observed for this case that multi-modality allows to exploit significantly both types of information. Interestingly, although the multi-modal system usually outperforms the individual modalities alone, there are some classes at which this does not happen, as in public square or metro.

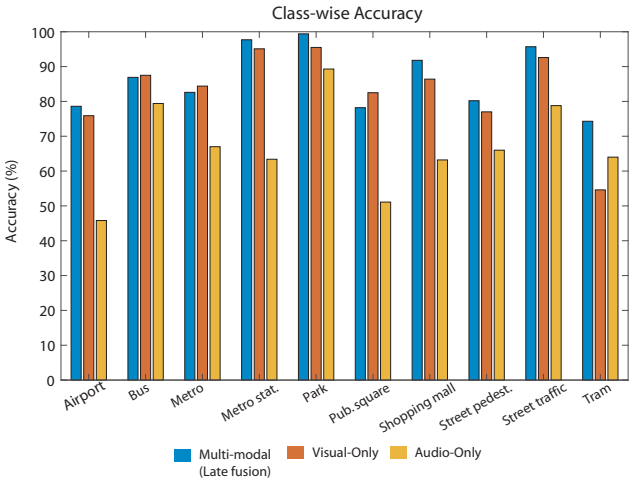


Figure 3: Class-wise performance in DCASE 2021 Task1b evaluation set.

4. CHALLENGE COMPARISON

The presented multi-modal framework ranked 7th in Task1b of the DCASE 2021 Challenge. It is important to remark that only 7 teams exceeded 85% accuracy. The team ranking 8th achieved an accuracy of 74% [25], 12 percentage points lower than our system. Moreover, our model presents a great performance-complexity balance. For example, the system ranking 6th [26] (with 88.4% accuracy) has 140M parameters, while ours has only 15M. All systems above 90% accuracy have more than 40M parameters, with the most complex one having 1B parameters [27]. Thus, the model presented in this paper allows very good accuracy with moderate complexity.

5. CONCLUSION

This paper presented a multi-modal system for audio-visual scene classification based on convolutional recurrent neural networks. The full system is based on two individual modules that are trained in isolation on the audio and visual modalities, respectively. The visual module is based on a time-distributed VGG16 model pre-trained on the places365 dataset, followed by a bidirectional recurrent layer. The audio module is a convolutional neural network that incorportes residual and squeeze-excitation techniques, working over Gammatone input representations. After training both modules, both are incorporated into a full audio-visual architecture that performs information fusion at early and late stages. Early fusion combines features extracted from both modalities at each time step into a bidirectional recurrent layer. Late fusion decides a final label after receiving the predictions obtained from each independent modality and that resulting from early fusion. The results show that the proposed framework is able to exploit successfully information from both modalities, even though the visual modality is considerably more accurate than the audio one. The results obtained in the DCASE 2021 Challenge confirm that the proposed system provides an excellent trade-off between prediction performance and system complexity.

6. ACKNOWLEDGEMENTS

This work is partially supported by ERDF and the Spanish Ministry of Science, Innovation and Universities under Grant RTI2018-097045-B-C21, as well as grants AICO/2020/154 and AEST/2020/012 from Generalitat Valenciana.

7. REFERENCES

[1] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 423–443, 2018.

[2] D. Zeng, M. Liao, M. Tavakolian, Y. Guo, B. Zhou, D. Hu, M. Pietikäinen, and L. Liu, "Deep Learning for Scene Classification: A Survey," *arXiv preprint arXiv:2101.10531*, 2021.

[3] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2015.

[4] J. Abeßer, "A review of Deep Learning based methods for Acoustic Scene Classification," *Applied Sciences*, vol. 10, no. 6, 2020.

[5] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: Benchmark and state of the art," *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1865–1883, 2017.

[6] K. Simonyan and A. Zisserman, "Very deep Convolutional Networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[8] G. Kalliatakis, "Keras-vgg16-places365," https://github.com/GKalliatakis/Keras-VGG16-places365, 2017.

[9] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, and M. Cobos, "Acoustic Scene Classification with squeeze-excitation residual networks," *IEEE Access*, vol. 8, pp. 112 287–112 296, 2020.

[10] S. Wang, A. Mesaros, T. Heittola, and T. Virtanen, "A curated dataset of urban scenes for audio-visual scene analysis," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, accepted. [Online]. Available: https://arxiv.org/abs/2011.00030

[11] S. Perez-Castanos, J. Naranjo-Alcazar, P. Zuccarello, M. Cobos, and F. J. Ferri, "CNN depth analysis with different channel inputs for Acoustic Scene Classification," *arXiv preprint arXiv:1906.04591*, 2019.

[12] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, and M. Cobos, "Task 1 DCASE 2020: ASC with mismatch devices and reduced size model using residual squeeze-excitation CNNs," DCASE2020 Challenge, Tech. Rep, Tech. Rep., 2020.

[13] S. Tabibi, A. Kegel, W. K. Lai, and N. Dillier, "Investigating the use of a Gammatone filterbank for a cochlear implant coding strategy," *Journal of Neuroscience Methods*, vol. 277, pp. 63–74, 2017.

[14] Z. Zhang, S. Xu, S. Cao, and S. Zhang, "Deep Convolutional Neural Network with mixup for environmental sound classification," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2018, pp. 356–367.

[15] S. Perez-Castanos, J. Naranjo-Alcazar, P. Zuccarello, and M. Cobos, "Anomalous Sound Detection using Unsupervised and Semi-Supervised Autoencoders and Gammatone Audio Representation," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 145–149.

[16] ——, "Listen Carefully and Tell: An Audio Captioning System Based on Residual Learning and Gammatone Audio Representation," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 150–154.

[17] M. Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, no. 1998, 1998.

[18] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in Python," in *Proceedings of the 14th Python in Science Conference*, vol. 8, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[20] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation Networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00745

[21] A. G. Roy, N. Navab, and C. Wachinger, "Concurrent spatial and channel 'squeeze & excitation'in fully convolutional networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 421–429.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[25] W. Boes and H. Van hamme, "Multi-source transformer architectures for audiovisual scene classification," DCASE2021 Challenge, Tech. Rep., June 2021.

[26] L. Pham, A. Schindler, M. Schutz, J. Lampert, and R. King, "DCASE 2021 task 1B: Technique report," DCASE2021 Challenge, Tech. Rep., June 2021.

[27] Q. Wang, S. Zheng, Y. Li, Y. Wang, Y. Wu, H. Hu, C.-H. H. Yang, S. M. Siniscalchi, Y. Wang, J. Du, and C.-H. Lee, "A model ensemble approach for audio-visual scene classification," DCASE2021 Challenge, Tech. Rep., June 2021.

DOMAIN GENERALIZATION ON EFFICIENT ACOUSTIC SCENE CLASSIFICATION USING RESIDUAL NORMALIZATION

Byeonggeun Kim[1], *Seunghan Yang*[1], *Jangho Kim*[1,2,*], *Simyung Chang*[1]

[1]Qualcomm AI Research†, Qualcomm Korea YH, Seoul, Republic of Korea
[2]Seoul National University, Seoul, Republic of Korea
{kbungkun, seunghan, jangkim, simychan}@qti.qualcomm.com

ABSTRACT

It is a practical research topic how to deal with multi-device audio inputs by a single acoustic scene classification system with efficient design. In this work, we propose Residual Normalization, a novel feature normalization method that uses frequency-wise normalization with a shortcut path to discard unnecessary device-specific information without losing useful information for classification. Moreover, we introduce an efficient architecture, BC-ResNet-ASC, a modified version of the baseline architecture with a limited receptive field. BC-ResNet-ASC outperforms the baseline architecture even though it contains the small number of parameters. Through three model compression schemes: pruning, quantization, and knowledge distillation, we can reduce model complexity further while mitigating the performance degradation. The proposed system achieves an average test accuracy of 76.3% in TAU Urban Acoustic Scenes 2020 Mobile, development dataset with 315k parameters, and average test accuracy of 75.3% after compression to 61.0KB of non-zero parameters. The proposed method won the 1st place in DCASE 2021 challenge, TASK1A.

Index Terms— acoustic scene classification, efficient neural network, domain imbalance, residual normalization, model compression

1. INTRODUCTION

Acoustic scene classification (ASC) is the task of classifying sound scenes such as "airport", "train station", and "urban park" to which a user belongs. ASC is an important research field that plays a key role in various applications such as context-awareness and surveillance [1, 2, 3]. Detection and Classification of Acoustic Scenes and Events (DCASE) [4] is an annual challenge, attracting attention to the field. There are various interesting tasks in the DCASE2021 challenge, and we aim for TASK1A: Low-Complexity Acoustic Scene Classification with Multiple Devices [5, 6].

TASK1A classifies ten different audio scenes from 12 European cities using four real and 11 simulated devices. In this year, the task becomes more challenging as an ASC model needs to solve two problems simultaneously which practically exist in real applications; First, data is collected from multiple devices, and the number of samples per device is unbalanced. Therefore, the proposed system needs to solve the domain imbalance problem while generalizing to different devices. Second, TASK1A restricts the model size and therefore requires an efficient network design.

† Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. * Author completed the research in part during an internship at Qualcomm Technologies, Inc.

In recent years, a number of researches have been proposed for more efficient and high-performance ASC. Most of them are based on convolutional neural network (CNN) using residual network and ensemble [7, 8, 9, 10]. The top-performing models in the previous TASK1A utilize multiple CNNs in a single model with parallel connections [7, 9]. For the generalization of the model, [8, 11] show that there is a regularization effect by adjusting the receptive field size in CNN-based design. However, these works also use models of several MB, and it is still challenging to satisfy the low model complexity of TASK1A of this year. In addition, when using the previous methods, we found an accuracy drop of up to 20% on the unseen devices compared to the device with sufficient training data. In this work, we propose methods to leverage the generalization capabilities of unseen devices while maintaining the model's performance in lightweight models. First, we introduce a network architecture for ASC that utilizes broadcasted residual learning [12]. Based on this architecture, we can achieve higher accuracy while reducing the size by a third of the baseline [8]. Next, we propose a novel normalization method, Residual Normalization (ResNorm), which can leverage the generalization performance for unseen devices. ResNorm allows maintaining classification accuracy while minimizing the influence on different frequency responses of devices by performing normalization of frequency bands in the residual path. Finally, we describe model compression combined with pruning and quantization to satisfy the model complexity of the task while maintaining performance using knowledge distillation.

This work is an expanded version from the challenge technical report submissions [13]. The rest of the paper is organized as follows. Section 2 describes the network architecture, Residual Normalization, and model compression methods. Section 3 shows the experimental results and analysis. Finally, we conclude the work in Section 4.

2. PROPOSED METHOD

This session introduces an efficient model design for device-imbalanced acoustic scene classification. First, we present a modified version of Broadcasting-residual network [12] for the acoustic scene domain. Following, we propose Residual Normalization for generalization in a device-imbalanced dataset. Finally, we describe how to get a compressed version of the proposed system.

2.1. Network Architecture

To design a low-complexity network in terms of the number of parameters, we use a Broadcasting-residual network (BC-ResNet) [12] which uses 1D and 2D CNN features together for better effi-

Table 1: **BC-ResNet-ASC.** Each row is a sequence of one or more identical modules repeated n times with input shape of frequency by time by channel and total time step T.

Input	Operator	n	Channels
$256 \times T \times 1$	conv2d 5x5, stride 2	-	2c
$128 \times T/2 \times 2c$	stage1: BC-ResBlock	2	c
$128 \times T/2 \times c$	max-pool 2x2	-	-
$64 \times T/4 \times c$	stage2: BC-ResBlock	2	1.5c
$64 \times T/4 \times 1.5c$	max-pool 2x2	-	-
$32 \times T/8 \times 1.5c$	stage3: BC-ResBlock	2	2c
$32 \times T/8 \times 2c$	stage4: BC-ResBlock	3	2.5c
$32 \times T/8 \times 2.5c$	conv2d 1x1	-	num class
$32 \times T/8\times$ num class	avgpool	-	-
$1 \times 1\times$ num class	-	-	-

ciency. While the BC-ResNet targets human voice, we aim to classify the audio scenes. To adapt to the differences in input domains, we make two modifications to the network, *i.e.*, limit the receptive field and use max-pool instead of dilation.

The proposed architecture, BC-ResNet-ASC, is shown in Table 1. The model has 5x5 convolution on the front with a (2, 2) stride for downsampling followed by BC-ResBlocks [12]. In [8], they show that the size of the receptive field can regularize CNN-based ASC models. We change the depth of the network and use max-pool to control the size of the receptive field. With a total of 9 BC-ResBlocks and two max-pool layers, the receptive field size is 109x109. We also do the last 1x1 convolution before global average pooling that the model classifies each receptive field separately and ensembles them by averaging. Original BC-ResNets use dilation in temporal dimension to obtain a larger receptive field while maintaining temporal resolution across the network. We observe that time resolution does not need to be fully kept in the audio scene domain, and instead of dilation, we insert max-pool layers in the middle of the network.

In this work, we use *BC-ResNet-ASC-1* and *BC-ResNet-ASC-8* whose base numbers of channels c are 10 and 80, respectively, in Table 1. Table 2 compares our BC-ResNet-ASC-8 with two baselines: CP-ResNet [8] which is a residual network-based ASC model with limited receptive field size; and original BC-ResNet-8 with the number of Subspectral Normalization [14] groups of 4. As shown in Table 2, BC-ResNet-ASC-8 records Top-1 test accuracy 69.5% with only one-third number of parameters compared to CP-ResNet showing 67.8% accuracy. Moreover, BC-ResNet-ASC-8 outperforms the original BC-ResNet-8 by a 1% margin with the modifications.

2.2. Residual Normalization

Instance normalization (IN) [15] is a representative approach to reducing unnecessary domain gaps for better domain generalization [16] or domain style transfer [17, 18] in the image domain. While domain difference can be captured by channel mean and variance in the image domain, we observe that differences between audio devices are revealed along frequency dimension as shown in Figure 1. To get audio device generalized features, we use instance normalization by frequency (FreqIN) as below.

$$FreqIN(x) = \frac{x - \mu_{nf}}{\sqrt{\sigma_{nf}^2 + \epsilon}}, \tag{1}$$

Table 2: **Network Architectures.** Compare Top-1 test accuracy (%) on TAU Urban AcousticScenes 2020 Mobile, development dataset.

Network Architecture	#Param	Top-1 Acc. (%)
CP-ResNet, c=64	899k	67.8
BC-ResNet-8, num SSN group $= 4$	317k	68.6 ± 0.4
BC-ResNet-ASC-8	315k	69.5 ± 0.3

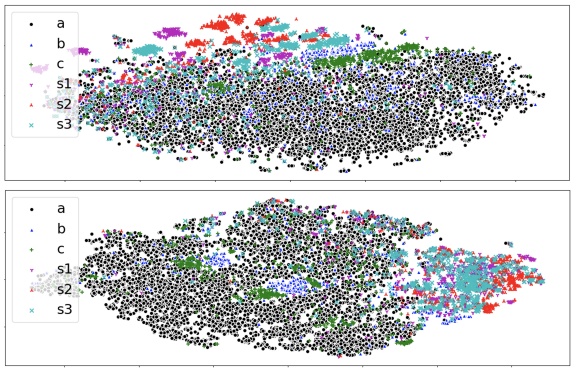


Figure 1: **2D t-SNE [19] visualization** of feature maps of BC-ResNet-ASC-1 stage2 (without ResNorm). **Top:** Concatenation of frequency-wise mean and standard deviations. **Bottom:** Concatenations of channel mean and standard deviations. The training samples are separated better by device ID (A to S3) with frequency-wise statistics.

where,

$$\mu_{nf} = \frac{1}{CT}\sum_{c=1}^{C}\sum_{t=1}^{T} x_{ncft},$$
$$\sigma_{nf}^2 = \frac{1}{CT}\sum_{c=1}^{C}\sum_{t=1}^{T} (x_{ncft} - \mu_{nf})^2. \tag{2}$$

Here, μ_{nf}, $\sigma_{nf} \in \mathbb{R}^{N\times F}$ are mean and standard deviation of the input feature $x \in \mathbb{R}^{N\times C\times F\times T}$, where N, C, F, T denote batch size, number of channel, frequency dimension, and time dimension respectively. ϵ is a small number added to avoid division by zero.

Direct use of IN can result in loss of useful information for classification contained in domain information. To compensate for information loss due to FreqIN, we add an identity shortcut path multiplied by a hyperparameter λ. We suggest a normalization method, named Residual Normalization (ResNorm) which is

$$ResNorm(x) = \lambda \cdot x + FreqIN(x). \tag{3}$$

We apply ResNorm for input features and after the end of every stage in Table 1. There are a total of five ResNorm modules in the network.

2.3. Model Compression

To compress the proposed model, we utilize three model compression schemes: pruning, quantization, and knowledge distillation.
Pruning. The pruning method prunes unimportant weights or channels based on many criteria. In this work, we choose a magnitude-based one-shot unstructured pruning scheme used in [20]. After

Table 3: **Residual Normalization.** We demonstrate how residual normalization affects BC-ResNet-ASC on TAU Urban AcousticScenes 2020 Mobile, development dataset. We show mean and standard deviation of Top-1 test accuracy (%) (averaged over 3 seeds, * averaged over 6 seeds).

Method	#Param	A	B	C	S1	S2	S3	S4	S5	S6	Overall
BC-ResNet-ASC-1 (Baseline)	8.1k	73.1	61.2	65.3	58.2	57.3	66.2	51.5	51.5	46.3	58.9 ± 0.8
BC-ResNet-ASC-1 + Global FreqNorm	8.1k	73.9	60.9	65.5	60.2	57.9	67.9	50.2	54.3	49.4	60.0 ± 0.9
BC-ResNet-ASC-1 + Fixed PCEN	8.1k	68.0	60.4	57.2	64.0	63.0	66.2	62.3	61.8	56.5	62.2 ± 0.8
BC-ResNet-ASC-1 + ResNorm	8.1k	**76.4**	65.1	**68.3**	**66.0**	62.2	**69.7**	63.0	63.0	58.3	$*65.8 \pm 0.7$
w/o ResNorm in Network	8.1k	75.1	**68.9**	67.0	**66.0**	63.9	69.3	63.4	**66.9**	**63.6**	$\mathbf{67.1 \pm 0.8}$
w/o Shortcut	8.1k	68.2	62.1	58.6	64.2	**65.3**	66.3	**65.1**	63.8	61.3	63.9 ± 0.7
BC-ResNet-ASC-8 + ResNorm	315k	**81.3**	**74.4**	**74.2**	**75.6**	**73.1**	**78.6**	73.0	**74.0**	**72.7**	$\mathbf{*75.2 \pm 0.4}$
w/o ResNorm in Network	315k	80.8	73.7	73.0	74.0	72.9	77.8	**73.3**	72.1	71.0	74.3 ± 0.3
w/o Shortcut	315k	78.3	73.5	69.1	73.8	72.9	75.6	72.2	72.5	71.0	73.2 ± 0.3

training, we conduct unstructured pruning on all convolution layers and do additional training to enhance the pruned model's performance.

Quantization. Quantization is the method to map continuous infinite values to a smaller set of discrete finite values. We quantize all of our models with quantization-aware training (QAT) with symmetric quantization [20]. We combine the pruning and quantization methods. It means that we quantize the important weights which are not pruned after the pruning process in the additional training phase. We quantize all convolution layers as an 8-bit while utilize the half-precision representation for other weights.

Knowledge Distillation. Knowledge Distillation (KD) trains the lightweight model using the outputs of a pre-trained teacher network. In general, previous model compression schemes such as pruning and quantization decrease the performance by reducing the model complexity. To enhance the performance of the compressed model, we use a KD loss [21] using the pre-trained model as a teacher network.

3. EXPERIMENTS

3.1. Experimental Setup

Datasets. We evaluate the proposed method on the TAU Urban Acoustic Scenes 2020 Mobile, development dataset [6]. The dataset consists of a total of 23,040 audio segment recordings from 12 European cities in 10 different acoustic scenes using 3 real devices (A, B, and C) and 6 simulated devices (S1-S6). The 10 acoustic scenes contain "airport", "shopping mall", "metro station", "pedestrian street", "public square", "street with traffic", "park", and travelling by "tram", "bus", and "metro". Audio segments from B and C are recorded simultaneously with device A, but not perfectly synchronized. Simulated devices S1-S6 generate data using randomly selected audio segments from real device A. Each utterance is 10-sec-long and the sampling rate is 48kHz. [6] divides the dataset into training and test of 13,962 and 2,970 segments, respectively. In the training data, device A has 10,215 samples while B, C, and S1-S3 have 750 samples each, which means the data is device-imbalanced. Devices S4-S6 remain unseen in training. In test data, all devices from A to S6 have 330 segments each.

Implementation Details. We do downsampling by 16kHz and use input features of 256-dimensional log Mel spectrograms with a window length of 130ms and a frameshift of 30ms. During training, we augment data to get a more generalized model. In the time dimension, we randomly roll each input feature in the range of -1.5 to 1.5 sec, and the out-of-range part is added to the opposite side. We also use Mixup [22] with $\alpha = 0.3$ and Specaugment [23] with two frequency masks and two temporal masks with mask parameters of 40 and 80, respectively, except time warping. We use Specaugment only for the large model, BC-ResNet-ASC-8. In BC-ResNet-ASC, we use Subspectral Normalization [14] as indicated in [12] with 4 sub-bands and use dropout rate of 0.1. We train the models for 100 epoch using stochastic gradient descent (SGD) optimizer with momentum to 0.9, weight decay to 0.001, mini-batch size to 64, and learning rate linearly increasing from 0 to 0.06 over the first five epochs as a warmup [24] before decaying to zero with cosine annealing [25] for the rest of the training. We use fixed $\lambda = 0.1$ for ResNorm in experiments. Due to the absence of validation split in the development dataset, we report the numbers of early stopping.

Baselines. We compare our method with other methods and do some ablation studies: 1) Global FreqNorm, which normalizes data by global mean and variance of each frequency bin; 2) Fixed per-channel energy normalization (PCEN) [26], which is an automatic gain control based dynamic compression and is used instead of log Mel spectrogram in our experiment; 3) *w/o ResNorm in Network*, which uses ResNorm module only at input not in the middle of the network. 4) *w/o shortcut*, which is a special case of ResNorm when $\lambda = 0$ in Equation 3 and uses FreqIN.

3.2. Residual Normalization

We do the experiments using BC-ResNet-ASC-1 and BC-ResNet-ASC-8, and the overall results are on Table 3. The task has multi-device inputs which are imbalanced with dominant device A. As a result, the baseline, BC-ResNet-ASC-1, shows that the accuracy of the device A is relatively higher than other seen devices, B, C, S1, S2, and S3. Furthermore, the accuracy on unseen devices, S4, S5, and S6 are even lower, and these results imply that the model is not generalized well to multiple devices, especially for unseen devices. When we use global normalization by frequency dimension, the result shows 60.0% accuracy which is 1% improvements compared to the baseline, but still we can observe poor domain generalization. We also try PCEN, a normalized feature instead of log Mel spectrogram. PCEN shows improvements for unseen devices, but we also observe that the performance of device A degrades due to its normalization. The proposed ResNorm uses FreqIN to get domain

Table 4: **Model compression** Compare bitwidth, top-1 test accuracy (%) on Tau Urban AcousticScenes 2020 Mobile, development dataset, and pruning ratio of the models (Average over 6 seeds).

BC-ResNet-ASC-8 + ResNorm, 300 epochs, KD				
Method	Bitwidth	KD	Pruning	Accuracy
Vanilla model	32	-	-	76.3 ± 0.8
Compressed model	8, 16		0.89	75.1 ± 0.9
Compressed model	8, 16	✓	0.89	75.3 ± 0.8

invariant features while not loosing the useful class-discriminative information through identity shortcut connection. The 'BC-ResNet-ASC-1 + ResNorm' shows a large improvement, 6% compared to baseline and records 65.8% test accuracy. The ResNorm shows performance improvements not just for unseen devices but also for all seen devices.

We do some ablation studies for the component of ResNorm. First, we use the ResNorm module as the preprocessing module, and do not use the module in the middle of the network; 'w/o ResNorm in Network'. For the small model, BC-ResNet-ASC-1, 'w/o ResNorm in Network' shows better performance, 67.1%, and for the larger model, BC-ResNet-ASC-8, it shows a performance degradation of 1%. Due to ResNorm's regularization effect, it was expected that this module could degrade the performance of a small network. We expect that the module can control the normalization power by the hyperparameter λ in Equation 3 to adapt to various size of networks. In this work, we use fixed $\lambda = 0.1$, and leave the automatic update of the λ as a future work. Second, 'w/o shortcut' shows the result when $\lambda = 0$ in ResNorm which equals to FreqIN in Equation 1. Our design motivation is that the shortcut path will keep the useful information for classification. The results show that FreqIN records relatively lower accuracy for seen devices compared to ResNorm. Especially, the margins on device A are 8.2% and 3.0% on BC-ResNet-ASC-1 and BC-ResNet-ASC-8, respectively.

3.3. Model Compression

Simultaneously, we distill the knowledge of the pre-trained teacher network ('Vanilla' model) into the compressed model for enhancing the performance and achieve the 0.2% improvement in test accuracy. In detail, we prune the convolution layers of the model with 89% pruning ratios compared to vanilla and quantize all convolution layers in a compressed model as an 8-bit. Other layers are quantized as a 16-bit. The resulting 'Compressed' model has 33K 8-bit nonzero for convolution layers and 15K 16-bit parameters for normalization, resulting in 61.5kB and shows 75.3% test accuracy which is 1% lower than Vanilla model. We use the ensemble of two compressed model in the DCASE 2021 challenge, task 1A.

4. CONCLUSIONS

In this work, we design a system to achieve two goals; 1) efficient design in terms of the number of parameters and 2) adapting to device imbalanced dataset. To design an efficient acoustic scene classification model, we suggest a modified version of Broadcasting residual network [12] by limiting receptive field and using max-pool. We compress the model further by utilizing three model compression schemes, pruning, quantization, and knowledge distillation. Moreover, we propose a frequency-wise normalization method, named Residual Normalization which uses instance normalization by frequency and shortcut connection to be generalized to multiple devices while not losing discriminative information. Our system achieves 76.3% test accuracy on TAU Urban Acoustic Scenes 2020 Mobile, development dataset with 315k number of parameters and the compressed version achieves 75.3 % test accuracy with 89% pruning, 8-bit quantization, and knowledge distillation. Residual normalization has a hyperparameter λ which can control the regularization power of the module. We leave the automatic update of the hyperparameter as future work.

5. REFERENCES

[1] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," in *Proc. Workshop Detection Classif. Acoust. Scenes Events*, 2016, pp. 95–99.

[2] R. Radhakrishnan, A. Divakaran, and A. Smaragdis, "Audio analysis for surveillance applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE, 2005, pp. 158–161.

[3] S. Chu, S. Narayanan, and C.-C. J. Kuo, "Environmental sound recognition with time–frequency audio features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1142–1158, 2009.

[4] http://dcase.community/challenge2021/.

[5] I. Martín-Morató, T. Heittola, A. Mesaros, and T. Virtanen, "Low-complexity acoustic scene classification for multi-device audio: analysis of dcase 2021 challenge systems," 2021.

[6] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, submitted. [Online]. Available: https://arxiv.org/abs/2005.14623

[7] S. Suh, S. Park, Y. Jeong, and T. Lee, "Designing acoustic scene classification models with CNN variants," DCASE2020 Challenge, Tech. Rep., June 2020.

[8] K. Koutini, H. Eghbal-zadeh, M. Dorfer, and G. Widmer, "The receptive field as a regularizer in deep convolutional neural networks for acoustic scene classification," in *EUSIPCO*. IEEE, 2019, pp. 1–5.

[9] H. Hu, C.-H. H. Yang, X. Xia, X. Bai, X. Tang, Y. Wang, S. Niu, L. Chai, J. Li, H. Zhu, F. Bao, Y. Zhao, S. M. Siniscalchi, Y. Wang, J. Du, and C.-H. Lee, "Device-robust acoustic scene classification based on two-stage categorization and data augmentation," DCASE2020 Challenge, Tech. Rep., June 2020.

[10] H. Chen, Z. Liu, Z. Liu, P. Zhang, and Y. Yan, "Integrating the data augmentation scheme with various classifiers for acoustic scene modeling," DCASE2019 Challenge, Tech. Rep., June 2019.

[11] S. S. R. Phaye, E. Benetos, and Y. Wang, "Subspectralnet–using sub-spectrogram based convolutional neural networks for acoustic scene classification," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 825–829.

[12] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," *CoRR*, vol. abs/2106.04140, 2021.

[13] B. Kim, S. Yang, J. Kim, and S. Chang, "QTI submission to DCASE 2021: Residual normalization for device-imbalanced acoustic scene classification with efficient design," DCASE2021 Challenge, Tech. Rep., June 2021.

[14] S. Chang, H. Park, J. Cho, H. Park, S. Yun, and K. Hwang, "Subspectral normalization for neural audio data processing," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 850–854.

[15] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016.

[16] H. Nam and H. Kim, "Batch-instance normalization for adaptively style-invariant neural networks," in *NeurIPS*, 2018, pp. 2563–2572.

[17] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*. IEEE Computer Society, 2017, pp. 1510–1519.

[18] D. Jung, S. Yang, J. Choi, and C. Kim, "Arbitrary style transfer using graph instance normalization," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 1596–1600.

[19] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[20] J. Kim, K. Yoo, and N. Kwak, "Position-based scaled gradient for model quantization and pruning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 20 415–20 426.

[21] J. Kim, M. Hyun, I. Chung, and N. Kwak, "Feature fusion for online mutual knowledge distillation," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 4619–4625.

[22] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR (Poster)*. OpenReview.net, 2018.

[23] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *INTERSPEECH*. ISCA, 2019, pp. 2613–2617.

[24] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[25] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with warm restarts," in *ICLR (Poster)*. OpenReview.net, 2017.

[26] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, "Trainable frontend for robust and far-field keyword spotting," in *ICASSP*. IEEE, 2017, pp. 5670–5674.

DETECTING PRESENCE OF SPEECH IN ACOUSTIC DATA OBTAINED FROM BEEHIVES

Pascal Janetzky[1], *Padraig Davidson*[1], *Michael Steininger*[1], *Anna Krause*[1], *Andreas Hotho*[1]

[1] Computer Science Dept., University of Würzburg, Würzburg, Germany
{janetzky,davidson,steininger,anna.krause,hotho}@informatik.uni-wuerzburg.de

ABSTRACT

Sound recorded from beehives is important to understand a colony's state. This fact is used in the *we4bee*[1] project, where beehives are equipped with sensors (among them microphones), distributed to educational institutions and set up to record colony characteristics at the communication level. Due to data protection laws, we have to ensure that no human is recorded besides the bees' sound. However, detecting the presence of speech is challenging since the frequencies of human speech and the humming of bees largely overlap. Despite having access to only a limited amount of labeled data, in this initial study we show how to solve this problem using Siamese networks. We find that using common convolutional neural networks in a Siamese setting can strongly improve the ability to detect human speech in recordings obtained from beehives. By adding train-time augmentation techniques, we are able to reach a recall of up to 100 %, resulting in a reliable technique adhering to privacy regulations. Our results are useful for research projects that require written permits for acquiring data, which impedes the collection of samples. Further, all steps, including pre-processing, are calculated on the GPU, and can be used in an end-to-end pipeline, which allows for quick prototyping.

Index Terms— Audio classification, Siamese networks, Speech detection, Deep learning

1. INTRODUCTION

With the introduction of the General Data Protection Regulation (GDPR) [1] in the European Union, publicly recording sound at any time, even for scientific purposes, requires written agreements and the immanent possibility to stop recording from the user's side. Smart home devices ensure this by only recording data after a signal word. Uploading recorded data is allowed, if no speaker can be recognized individually from the recording (i.e., distortion) or the recording device ensures no privacy concerned data is contained in the file (i.e., speech detection). Despite these challenges, it is desirable to use this data since sound is rich in information and enables more fundamental understandings of communicating organisms, such as bee colonies.

In this paper we focus on the task of detecting the presence of speech in audio signals obtained from beehives of our *we4bee* project. The we4bee project distributed over 100 *Top Bar Hives* (TBH), mainly to educational institutions in Germany. Within these "smart" beehives (see Figure 1), we can continuously monitor the state of the colony, and even analyse the auditory communication of the bees. Bee monitoring systems are an important tool for apiarists [2], and especially sound recordings are inevitable to fully understand the beehive's state [3]. Especially in swarming prediction, sound recordings have shown great success, in contrast to simpler monitoring, such as observing the temperature [4], which we used in prior work [5]. Our contributions are as follows: 1) An algorithm which achieves high recall, allowing allows us to detect human speech in a challenging environment, 2) a privacy-regulation conform recording approach without distorting the signals, and 3) to the best of our knowledge, the first study on human speech detection for smart apiculture.

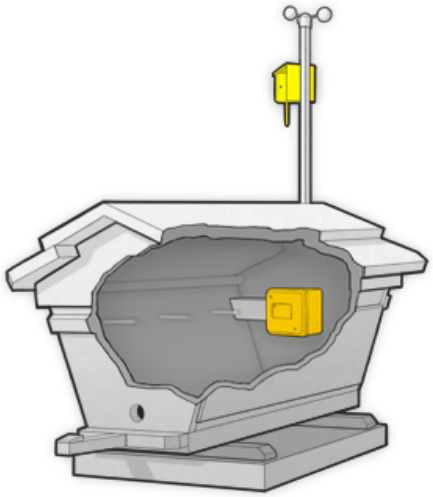


Figure 1: Sensor placement of the TBH. This cutaway view of the beehive shows the sensor placement of all sensors. The placement of the two microphones (one inside, one outside), is highlighted in yellow. Both mono sources are merged as a stereo-signal and uploaded as such. Image taken from we4bee[1]

2. RELATED WORK

As one of the first, [6] uses a convolutional neural network (CNN) to classify sound. This model is trained on the Environmental Sound Classification dataset (ESC-50) [7], consisting of 50 classes with 40 samples each. All samples are 5 s long and split into overlapping spectrogram segments. The length of the extracts is 950 ms (short variant) and 2.3 s (long variant), Lastly, the probabilities of all segment-level predictions are taken into account to obtain a final prediction. The authors find that using the longer samples improves the classification accuracy, reaching a score of 64.5 %.

In 2017, Stowell et al. hosted the Bird Audio Detection challenge (BADc) [8]. For this challenge, the task is to detect the presence

[1] https://we4bee.org

Table 1: Overview of the dataset.

Dataset	negative samples	positive samples
Train	119	16
Validation	27	3
Test	25	10
Overall	171	29

of any bird sound in short (mostly 10 s) audio recordings. The development data and test data come from different sources, which requires methods that generalize to unseen recording conditions. The highest scoring approach uses CNNs on spectrogram inputs and cyclical time shifting to classify the data [9]. With their submission named `bulbul`, which works on spectrograms calculated over 14 s, they reach an area under ROC score (AUROC) of 88.7 %.

Building on this architecture [10] classify sound excerpts of audio data recorded from beehives. The excerpts are labeled as containing bee-related sound or containing external sounds. Mel spectrograms are then calculated and used as input features to the network, with random pitch and time shifting augmenting the training data. Using a wide receptive field of 30 s, the classifier network reaches an AUROC score of 90.1 %.

In [11], Manocha et al. use a Siamese network [12] to compare the similarity of input pairs. On several audio datasets, among them ESC-50, they study the problem of retrieving semantically similar audio clips. In their setup, log-scaled spectrograms are calculated for the data, and feed to Siamese networks to obtain dense embedding vectors. Using a k nearest neighbor search on the embeddings, the authors achieve a mean precision of up to 78.4 %, indicating that the learned representations successfully capture similarity.

In this work, we use the mentioned CNNs in a Siamese setting to detect speech presence in audio recordings obtained from beehives.

3. DATASET

In order to enable continuous sound monitoring with we4bee, we obtained permission to record audio at one location. Since the data for this feasibility study currently originates from a single beehive only, we plan to add more recording stations in the future. Recording started in May 2020, and is still running. For the purpose of this paper, we used one day each in August and September for training and validation data, and a separate day in October for final testing.

Each audio sample is recorded at 44.1 kHz, 16 bits resolution for 60 s. We let one voluntary annotator manually label a small, random amount of these recordings on file level. Each sample is hand-labeled as class 0, no human speech present (negative samples), or as class 1, meaning the presence of human speech (positive samples). Additionally, to collect more positive samples, we followed a simple heuristic: Every time speech was detected in the current sample, we searched in a ± 10 min interval for more positive samples. An exemplary spectrogram of a recording with speech segments can be seen in Figure 2. Table 1 lists the complete dataset statistics.

This dataset poses two challenges: First, human speech is only sparsely present, both in terms of absolute numbers of samples and relative time within the samples. Analysing shorter windows would yield more samples but also lead to higher class imbalance towards class 0 which is why we kept the 60 s windows from the recordings. These samples come from a broad range of situations, from children playing far away (hardly audible) to adults talking next

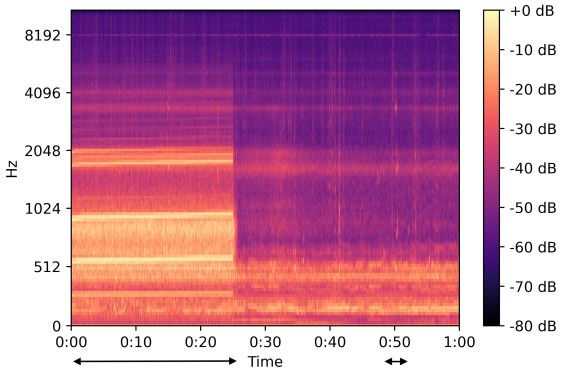


Figure 2: Sample audio file containing speech. A high noise level is created by a motor moving until 25 s, marked by the large arrow. The speech, located at 50 s (small arrow), is covered by the background noise and not visually discernible.

to the beehives (clearly audible). Second, the general limited number of annotated samples increases the difficulty. Large supervised audio datasets such as the ones used for the BADc have 24 000 annotated samples and more. In contrast, our labeled dataset only consists of 200 samples in total. The challenge is therefore to learn a model to classify a diverse, unbalanced, and small dataset.

4. METHODS

4.1. Siamese Neural Networks

Siamese neural networks [12] are a class of networks that learn the similarity of an input pair. Each sample is fed into the neural network to obtain a dense representation, termed embedding. A distance metric (i.e., Euclidean distance) is then used to calculate the distance between the embeddings. To interpret the result, one often applies a sigmoid activation function, forcing the values to lie between $[0, 1]$. An output close to 0 indicates highly similar samples, conversely values near 1 indicate high dissimilarity. The term "siamese" refers to the fact that the same set of weights is used to calculate each embedding of the input pair.

We train the Siamese networks to minimize the distance between audio pairs from the same class and to maximize the distance for opposite pairs. For this, we randomly draw an audio sample, and pair it both with a random sample from the same class (this pair is labeled as 0), and with a random sample from the opposite class (labeled as 1). The learned embeddings are used to train a kNN classifier [13, 14]. To obtain class predictions for the test samples, we first extract embeddings for the test data and then query the classifier.

4.2. Base Neural Networks

Motivated by the frequent usage of mel-scaled spectrograms (Mel spectrograms from now on) as input features, we utilize the two networks briefly introduced in Section 2 (the networks trained on the ESC dataset and submitted to the BADc, respectively) , and one published in an introductory article on sound classification with CNNs. We adapt each network to accept the raw audio and compute the

Mel spectrograms as part of the forward pass, using the `kapre` [15] package. With this modification, the computation is accelerated by the GPU, making a separate on-CPU pre-processing step obsolete.

Saeed The first network is from work published by Aaqib Saeed [16], which we name `Saeed` after its author. The input Mel spectrogram features are merged with their deltas, local estimates of the derivative that capture the transition dynamics of sound. The resulting two-channel representation is then framed, splitting the vector into small excerpts. These excerpts are processed by four sets of the following stack: convolution → batch normalization [17] → ReLU activation [18, 19] → max pooling. Additionally to the proposed architecture, we add a drop dropout [20] layer before each pooling operation and replace the max- with global max-pooling in the last stack. Afterwards, we add a further dropout operation prior to two ReLU dense layers. We modify the final layer to have a single neuron only and replace the softmax with a sigmoid activation.

Bulbul The highest-scoring submission in the aforementioned BADc is a CNN named Bulbul [9]. The key feature of this network is its wide receptive field, enabling it to find short local events. The input Mel spectrograms are normalized with batch normalization, followed by four sets of the following stack: convolution → leaky ReLU → max pooling layers. The output is then flattened, and followed by two blocks of dropout → dense → leaky ReLU layers. The final layer has a single neuron with sigmoid activation.

ESC The third model we used is the network introduced for the ESC dataset (see Section 2 and [7, 6]), which we call `ESC`. The Mel spectrogram is stacked with its delta features. The computation of these features is followed by a single stack of convolution → dropout → max-pooling. After another convolution and max pooling layer, the tensor is flattened and processed by two dense layers, each with dropout. The output activation is a binary sigmoid.

5. EXPERIMENTAL SETUP

For all experiments, we utilize the Adam optimizer [21] with default parameter values: A learning rate of 0.001, β_1 of 0.9, and β_2 of 0.999. Each model is trained and evaluated on our dataset five times, and the results are averaged. We use `librosa` [22] to load and downsample the audio to 22 050 Hz. As a metric function, we follow [9] and report the area under the receiver-operating curve (AUROC) score [23, 24, 25]. This metric first calculates recall versus fall-out at various threshold levels, yielding the ROC curve. The area under this curve captures the performance of a classifier in a single metric. A value of 0.5 equals random guessing, a value of 1.0 is equal to a perfect classifier. For our imbalanced two-class dataset, we use the AUROC metric, as opposed to misleading accuracy scores. Further, for the baseline networks, we interpret the binary output as class 1 if it is above the default threshold of 0.5, and as class 0 otherwise. Similarly, for the Siamese networks the default threshold is 0.5. Additionally, we report the recall for class 1 (speech). Since samples of class 1 might contain sensitive information, we are interested in particularly high recall. Therefore, regarding privacy, a false negative is more severe than a false positive.

5.1. Baselines

We initially tried various clustering algorithms, which scored only slightly better than random guessing and where thus not further evaluated. We therefore used the three CNNs introduced in Section 4.2 as baseline networks, without using a Siamese setting. For the training we follow the approach of [9]: We train the network for 100 epochs, use EarlyStopping [26] with a patience of 20 epochs and a batch size of 16. We reduce the learning rate by a factor of 10, if the area-under-curve score has not improved for ten consecutive epochs.

Table 2: Baseline networks on the test set, averaged over five runs.

Network	AUROC	Recall speech
Saeed	0.6125 ± 0.0400	0.0
Bulbul	0.7525 ± 0.0100	0.0
ESC	0.5017 ± 0.0300	0.0

5.2. Siamese Network

We use all networks introduced in Section 4.2 in a Siamese setting. For all networks, we replace the final hidden dense and any subsequent layer with a single dense layer of 128 neurons, which is interpreted as the embedding vector. All embedding vectors are normalized using $L2$ normalization.

We train our Siamese network for 100 epochs and use EarlyStopping on the validation AUROC score with a patience of 20 epochs. Since our audio samples are quite large, we use small batch sizes for the training. The default value is 4, which means that four pairs are created, using 8 individual audio samples in total.

To obtain better scores, we also try train-time augmentations on the raw audio. For this, we used the `audiomentations`[2] package. With a probability of 50 % each, we add gaussian noise, use time-shifting of ± 30 s, and shift the pitch ± 2 semitones.

Further, we experiment with more epochs (500) and try different EarlyStopping offsets (300 and 500), which is the number of guaranteed update steps done before the counter begins. We try different values for the number of neighbors $k \in \{1, 3, 5\}$.

6. RESULTS & DISCUSSION

As summarized in Table 2, of the baseline networks, the `ESC` network reaches the lowest AUROC score, with 50.17 %. The next best network, `Saeed`, scores more than 10 points higher, reaching 61.25 %, and the best network, `Bulbul`, reaches 75.25 %. However, our primary indicator of performance, the recall of the speech class, is drastically lower; all three networks achieve a recall of 0. Using a Siamese setting, we can improve the score for all three networks (Table 3), and our strongest candidate reaches 94 % speech recall and an AUROC score of 96.88 %. The `ESC` network does not benefit from a Siamese setup, it reaches only slightly better scores compared to the non-Siamese setting. Generally, a higher number of neighbors when classifying the test data via kNN improves scores.

When using train-time augmentations, as described in Section 5.2, we observe two primary effects: First, the scores are worse compared to no augmentation, as shown in Table 4. Secondly, training takes considerably longer since the computation is done on the CPU. On examination of the validation AUROC curve we noticed that the scores heavily oscillate in the beginning. Before the scores stabilize and increase, either the EarlyStopping patience prematurely terminates the run or the maximum number of epochs (100) is reached.

These instabilities can be overcome by using an offset for EarlyStopping and by training for more epochs. We find that an offset of 300 is sufficient when training for 500 epochs. Combined

[2] `https://git.io/JcQJQ`

Table 3: Scores for the Siamese networks on the test dataset, averaged over five runs, without augmentation.

	Saeed		Bulbul		ESC	
k	AUROC	Recall speech	AUROC	Recall speech	AUROC	Recall speech
1	0.8633 ± 0.1600	0.76 ± 0.31	0.9500 ± 0.0500	0.90 ± 0.01	0.5317 ± 0.0500	0.18 ± 0.05
2	0.8988 ± 0.1400	0.76 ± 0.31	0.9696 ± 0.5000	0.90 ± 0.10	0.5258 ± 0.0600	0.18 ± 0.05
3	0.9046 ± 0.1300	0.74 ± 0.29	0.9688 ± 0.0500	0.94 ± 0.10	0.5537 ± 0.0600	0.04 ± 0.05

Table 4: Scores for the Siamese networks on the test dataset, averaged over five runs, with augmentation. Augmenting the training data stops the training prematurely, as explained in Section 6.

	Saeed		Bulbul		ESC	
k	AUROC	Recall speech	AUROC	Recall speech	AUROC	Recall speech
1	0.6092 ± 0.1900	0.26 ± 0.34	0.4867 ± 0.0200	0.04 ± 0.05	0.5192 ± 0.0400	0.18 ± 0.11
2	0.6775 ± 0.1500	0.26 ± 0.34	0.4838 ± 0.5000	0.04 ± 0.05	0.5254 ± 0.0600	0.18 ± 0.12
3	0.6837 ± 0.1400	0.28 ± 0.31	0.5083 ± 0.0500	0.04 ± 0.09	0.4908 ± 0.1000	0.00 ± 0.00

Table 5: Scores for the Siamese networks on the test dataset, averaged over five runs, with augmentation. The training is done with an offset of 300 epochs for EarlyStopping. Compared to Table 4, using such an offset can lead to improved results.

	Saeed		Bulbul		ESC	
k	AUROC	Recall speech	AUROC	Recall speech	AUROC	Recall speech
1	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.495 ± 0.020	0.04 ± 0.09
2	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.51 ± 0.05	0.04 ± 0.09
3	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.52 ± 0.07	0.06 ± 0.13

with augmentation, we reach perfect AUROC and recall scores (100 %) with both `Bulbul` and `Saeed`. These perfect scores indicate overfitting, which might be due to the relatively small dataset. We plan to address this in further research by increasing both the dataset's size and diversity. Nonetheless, for this initial study the results are very encouraging. A sample embedding is visualized with t-SNE [27, 28] in Figure 3, which shows how the classes are separated well in space. As before, the `ESC` network does not benefit. We suspect that this is due to the relatively shallow architecture, which may prevent the network from learning meaningful features.

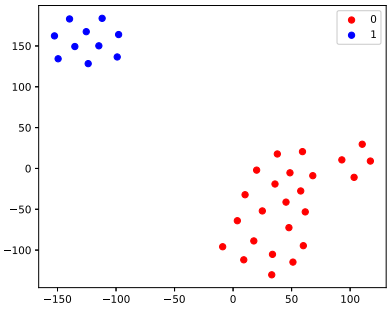


Figure 3: t-SNE plot (perplexity=12.5) of the learned embeddings, obtained from the Siamese `Bulbul` network for the test dataset. The network can perfectly separate the data.

7. CONCLUSION

Sound recordings of beehives are an important source of information for modeling the behavior of bees but it is not trivial to record publicly accessible hives in a privacy preserving manner. To allow for GDPR-compliant sound recordings of bee colonies we considered multiple approaches for presence of speech detection, allowing us to detect and remove human speech before storing the sound data. In this initial feasibility study, we used three convolutional neural networks to detect presence of speech in these challenging recordings. By using them in a Siamese setting, we achieve high recall. Motivated by the good results, we then used augmentation techniques and increased the number of epochs to achieve perfect recall and AUROC scores. For our small datasets, these results are promising, but open the opportunity for future work in several directions:

First, our current dataset is limited to recordings from a single beehive. In prospective work, it can be enriched with recordings from more beehives. This would capture more locations and characteristics, allowing to better examine the ability to generalize. Second, the code can be adapted for on-device inference. Currently, for the beehives we have permission to record, we upload the audio data to the cloud. Only then do we check for the presence of speech. However, this step can be greatly simplified by running the detection directly on the Raspberry Pi which powers all beehive sensors.

Acknowledgements All audio recordings were taken from Prof. Hotho's TBH with his written consent allowing the recording. We thank Albin Zehe for his helpful commentary on this paper.

8. REFERENCES

[1] General data protection regulation (gdpr). European Commission. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

[2] S. Cecchi, S. Spinsante, A. Terenzi, and S. Orcioni, "A smart sensor-based measurement system for advanced bee hive monitoring," *Sensors*, vol. 20, no. 9, p. 2726, 2020.

[3] A. Terenzi, S. Cecchi, and S. Spinsante, "On the importance of the sound emitted by honey bee hives," *Veterinary Sciences*, vol. 7, no. 4, p. 168, 2020.

[4] S. Ferrari, M. Silva, M. Guarino, and D. Berckmans, "Monitoring of swarming sounds in bee hives for early detection of the swarming period," *Computers and electronics in agriculture*, vol. 64, no. 1, pp. 72–77, 2008.

[5] P. Davidson, M. Steininger, F. Lautenschlager, K. Kobs, A. Krause, and A. Hotho, "Anomaly detection in beehives using deep recurrent autoencoders," in *Proceedings of the 9th International Conference on Sensor Networks (SENSORNETS 2020)*, no. 9. SCITEPRESS – Science and Technology Publications, Lda., 2020, pp. 142–149.

[6] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2015, pp. 1–6.

[7] ——, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018.

[8] D. Stowell, M. D. Wood, H. Pamuła, Y. Stylianou, and H. Glotin, "Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge," *Methods in Ecology and Evolution*, vol. 10, no. 3, pp. 368–380, 2019.

[9] T. Grill and J. Schlüter, "Two convolutional neural networks for bird detection in audio signals," in *2017 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 1764–1768.

[10] I. Nolasco and E. Benetos, "To bee or not to bee: Investigating machine learning approaches for beehive sound recognition," *arXiv preprint arXiv:1811.06016*, 2018.

[11] P. Manocha, R. Badlani, A. Kumar, A. Shah, B. Elizalde, and B. Raj, "Content-based representations of audio using siamese neural networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 3136–3140.

[12] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, vol. 6, pp. 737–744, 1993.

[13] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review/Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[14] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[15] K. Choi, D. Joo, and J. Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras," in *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML, 2017.

[16] A. Saeed. (2016) Urban Sound Classification. [Online]. Available: http://aqibsaeed.github.io/2016-09-24-urban-sound-classification-part-2/

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[18] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[19] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Citeseer, 2013, p. 3.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8. Citeseer, 2015, pp. 18–25.

[23] D. K. McClish, "Analyzing a portion of the roc curve," *Medical decision making*, vol. 9, no. 3, pp. 190–195, 1989.

[24] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[25] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine learning*, vol. 45, no. 2, pp. 171–186, 2001.

[26] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[27] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

A CONTRASTIVE SEMI-SUPERVISED LEARNING FRAMEWORK FOR ANOMALY SOUND DETECTION

Xinyu Cai[1,2], Heinrich Dinkel[2], Zhiyong Yan[2], Yongqing Wang[2], Junbo Zhang[2], Zhiyong Wu[1], Yujun Wang[2]

[1]Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[2]Xiaomi Corporation, Beijing, China
caixy19@mails.tsinghua.edu.cn
{dinkelheinrich,yanzhiyong, zhangjunbo1,wangyongqing3,wangyujun}@xiaomi.com
zywu@sz.tsinghua.edu.cn

ABSTRACT

Anomaly Sound Detection (ASD) is a popular topic in deep learning and has attracted the attention of numerous researchers due to its practical applications within the industry. In the case of unsupervised conditions, how to better discover the inherent consistency of normal sound clips has become a key issue in ASD. In this paper, we propose a novel training framework that jointly trains two different feature extractors using contrastive loss to obtain a better representation of normal sounds in the latent space. We evaluate our framework on the development dataset of DCASE 2021 challenge task 2. Our framework is a combination of two baseline systems from the challenge: 1) An AutoEncoder-based model and 2) a MobileNetV2-based model. Our approach trains two models, whereas during inference only model 2) is used. Experimental results indicate that the MobileNetV2-based model trained under our proposed training framework exceeds the baseline model in terms of the official score metric. Since we participated in the challenge and submitted the system trained on the proposed framework with some data augmentation methods, we also analyze the results of DCASE 2021 challenge task 2 and discuss the effect of the median filter as a data augmentation technique. Notably, our proposed approach achieves the first place for anomaly detection for the machine type "Fan" with an AUC of 90.68 and a pAUC of 79.99.

Index Terms— Unsupervised anomaly sound detection, autoencoder, convolutional nerual network, contrastive learning

1. INTRODUCTION

Anomaly sound detection (ASD) is the task of identifying whether the sound emitted from an object is normal or anomalous. It has a wide range of applications, such as machine condition monitoring and home monitoring.

In this paper, we focus on ASD in an unsupervised setting, which means that only normal (positive) sound samples can be accessed during the training phase, while during evaluation abnormal (negative) samples need to be ascertained. These settings commonly occur in real-world scenarios, where diverse anomalous sounds rarely occur. Therefore, collecting a dataset that contains exhaustive anomalous patterns is hard.

The main idea of unsupervised ASD is to learn the inherent consistency of the normal sounds, and then classify samples as anomalous or normal by the deviation of a sample from normal sound properties. Early researchers adopted statistic-based methods such as Hidden Markov Model [1] (HMM) and Gaussian Mixture Model [2](GMM) to model the probability distribution of normal sound. Anomalous sounds are usually outside of the normal sound distribution, thus we can determine whether the sound is abnormal by its posterior probability. Other researchers used generative models such as Non-negative Matrix Factorization [3] (NMF) and Autoencoder approaches [4]. These models are trained to compress and reconstruct normal sounds to learn a normal sound's properties in latent space. If an abnormal sample is fed into a generative model, the model will likely produce large reconstruction errors, meaning that the sample has not been seen during training and thus is abnormal.

Recently in the DCASE challenges, the classifier-based method showed promising performance [5, 6, 7]. Supervised training is made possible since the challenge training data is composed of normal sounds from different operating conditions with different section IDs. Classifier based ASD method uses the section ID as a label and then performs classification on latent features. Since we have access to the section ID during inference, a classifier could perform anomaly sound detection by identifying misclassified samples (wrong section ID) as anomaly sounds.

As we can see from previous works, for deep learning based anomaly sound detection methods, a key issue to improve the performance is to obtain better latent space features of normal sounds, both for the widely used Autoencoder method and classifier-based method. Inspired by the recent success of contrastive learning approaches for self-supervised audio pretraining [8, 9, 10], we aim to enhance the model's capability to detect unseen events by linking multiple views together. Our proposed learning framework is a novel combination of two mainstream anomaly detection models trained with an additional contrastive loss function.

The paper is structured as follows: In Section 2 we introduce our proposed learning framework and its components. Further, in Section 3 details regarding the dataset and experimental setup are provided. Results are provided in Section 4 and the conclusion is given in Section 5.

2. PROPOSED APPROACH

During the training phase, our approach jointly trains two individual models: an unsupervised AE-based model combined with a supervised convolutional neural network (CNN). Once the loss converges, inference can be performed using either model independently. The architecture can be seen in Figure 1.

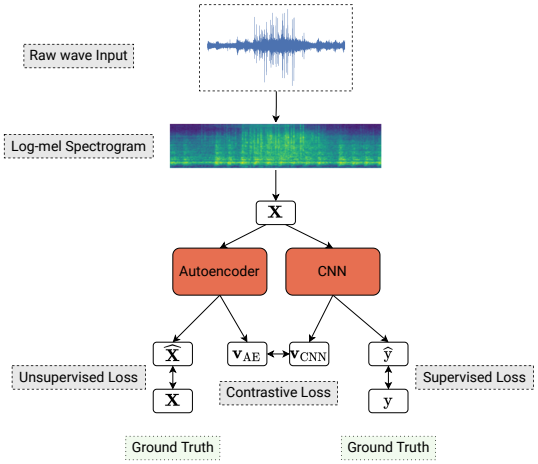


Figure 1: The proposed architecture used in this work. A spectrogram feature is first extracted from the input waveform. Then the feature is fed into two separate models: an Autoencoder (AE) and a Convolutional neural network (mainly MobileNetV2). The models are jointly optimized to reconstruct the input spectrogram, predict the section label and minimize the contrastive loss between the two models' hidden representations.

2.1. Autoencoder-based unsupervised classification

Our AE baseline model is a fully connected neural network with a bottleneck structure and trained to reconstruct a given input sound (normal sound). Ideally, a well-trained AE will produce a low error if a new data sample has been seen during the training phase (normal sample) and a large error when it encounters unseen anomalous sounds.

Formally, let x be an input sample and AE be the autoencoder, our training objective follows:

$$\begin{aligned} \mathrm{AE}(x) &\mapsto \hat{x}, \\ \mathcal{L}_{\text{unsup}}(\cdot) = \mathcal{L}_{\text{AE}}(x) &= \mathcal{L}_{\text{MSE}}(\hat{x} - x), \end{aligned} \tag{1}$$

where the training loss is chosen to be the mean square error (MSE).

2.2. MobileNet-based supervised classification

Our supervised approach uses the provided section ID as classification targets and predicts each section's probability. Formally, for a sample x and corresponding one-hot target y, we compute the standard cross-entropy (CE) loss, as seen in Equation (2).

$$\begin{aligned} \mathrm{CNN}(x) &\mapsto \hat{y}, \\ \mathcal{L}_{\text{sup}}(\cdot) = \mathcal{L}_{\text{CE}}(\hat{y}, y) &= -\frac{1}{N}\sum_{i}^{N} y_i \log \hat{y}_i, \end{aligned} \tag{2}$$

where CNN represents the CNN-based classifier and N the number of samples. Then the anomaly score $A(x)$ is calculated as:

$$A(x) = \log\left(\frac{1-\hat{y}_i}{\hat{y}_i}\right), \tag{3}$$

where $\hat{y}_i$ is the softmax output for the correct section. Note that if the sample x is divided into consecutive segments $(x_1, x_2, ..., x_P)$, the anomaly score will be $\frac{1}{P}\sum_i^P A(x_i)$.

2.3. Proposed contrastive semi-supervised learning

We train these models with an additional contrastive loss [11]. The contrastive loss $\mathcal{L}_{\text{contrastive}}$ is added between the hidden representations of both models ($\mathbf{v}_{\text{AE}}, \mathbf{v}_{\text{CNN}}$) as:

$$\begin{aligned} \mathbf{p} &= \mathbf{v}_{\text{AE}}, \\ \mathbf{u} &= \mathbf{v}_{\text{CNN}}, \\ \mathcal{L}_{\text{contrastive}}(\cdot) &= -\sum_i \log \frac{\exp(\langle \mathbf{u}_i, \mathbf{p}_i \rangle / \rho)}{\sum_{j \neq i} \exp(\langle \mathbf{u}_i, \mathbf{p}_j \rangle / \rho)}, \end{aligned} \tag{4}$$

where $\langle , \rangle$ represents inner product, $\rho \in \mathbb{R}$ is a scalar hyperparameter and $\mathbf{p}, \mathbf{u} \in \mathbb{R}^{256}$ are hidden vector representations obtained by both models via projection. Concretely speaking, we transform the output vector of Autoencoder's bottleneck layer and CNN's feature layer into same dimension by linear transformation, then map representations to the space where the contrastive loss is applied via a shared MLP projection layer with one hidden layer. In most cases, the dimension of the bottleneck layer in the Autoencoder is much smaller than the dimension of the feature layer in the CNN model (8 *vs.* 1280 in this paper). We assume that the bottleneck layer output in the AE tends to represent the general structure of normal sound clips, while CNN extracted feature represents their microscopic structure. Our approach aims to obtain two different representations of a single sample, which is reminiscent of SimCLR [8], unsupervised data augmentation (UDA) [12] and other semi and self-supervised approaches.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{unsup}} + \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{contrastive}} \tag{5}$$

The final loss for optimization can be seen in Equation (5).

2.4. Data Augmentation

One of our contributions is the exploration of data augmentation techniques. Regarding conventional techniques, we explore the use of Mixup [13] along with time masking [14] and frame-shifting for model training during the DCASE challenge. Further, our intuition is that the input audio data contains large amounts of short-time noise, thus an input feature might contain a surplus of unreliable information, which can affect the performance of our supervised training method. We propose a median filtering approach applied on the input spectrogram feature along the frequency axis aiming to reduce the influence of distracting noise.

3. EXPERIMENTAL SETUP

Log Mel-spectrogram (LMS) features are chosen as the default front-end feature for the task. Overall, seven models are trained in our approach, one for every machine type.

For the supervised CNN training, each 128-filter LMS is extracted from a 64 ms window with a stride of 32 ms. We follow the baseline approach by concatenating 64 consecutive frames with a shift of 8 frames, resulting in an 128×64 dimensional input tensor. If segments are shorter than 10 seconds (or 311 samples), we zero-pad the input to the longest sample within a batch.

Regarding the AE training, we flatten the input tensor to a single input vector of size 8192 ($128 * 64$). All experiments are run for 100 epochs, with the learning rate halving every 30 epochs. The batchsize is set to 32 for training and we set the hyperparameter $\rho = 0.07$ for the contrastive loss. Our median filtering approach

Table 1: Performance of our models in comparison to other participants in the challenge on the official evaluation dataset. Best results are highlighted in bold.

Model	Official Score	Fan	Gearbox	Slider	Toy Train	Toy Car	Pump	Valve
AE Baseline	56.375	60.68	65.49	57.22	68.51	65.93	58.30	51.87
MBv2 Baseline	54.770	64.96	51.14	72.92	42.91	42.73	67.97	53.13
1st	66.798	61.01	63.07	83.18	**69.15**	**75.27**	**86.76**	65.36
2nd	64.956	86.48	67.45	83.05	45.60	60.88	85.04	**71.49**
3rd	64.201	88.98	57.75	**86.84**	57.50	69.83	74.82	62.74
4th	63.745	66.60	62.53	86.27	61.79	61.70	74.60	62.36
5th	62.593	68.98	**67.74**	79.88	61.71	73.32	71.87	63.73
6th	62.239	82.65	57.20	83.76	53.43	58.67	85.54	60.54
7th	61.480	87.68	56.56	76.66	48.24	70.60	72.54	60.70
8th	61.186	73.17	64.70	69.89	51.71	68.23	78.65	53.93
Ours best	60.966	**90.68**	58.00	77.34	47.49	53.81	77.82	53.53

uses a window size of 31 frames (i.e., 1 second) for each filter bank respectively.

PyTorch [15] was used as the default neural network toolkit[1].

3.1. Evaluation metrics

The evaluation metrics used in the challenge is the area under curve (AUC) and partial-AUC (pAUC) scores respectively [16]. The final official score Ω is computed as the harmonic mean of the AUC and pAUC scores.

3.2. Dataset

The data used for this task consists of running sounds of seven machine types being "ToyCar", "Fan", "ToyTrain","Valve", "Gearbox", "Silder" and "Pump", including two recent machine audio datasets, ToyADMOS [17] and MIMII [18].

Notably, all provided data samples by the challenge authors have a length of 10 seconds, and each section, as well as machine type, has a near uniformly distributed duration. The overall data length is 70 hours of which the large majority belongs to the source domain.

Model	Fan	Gearbox	Slider	Toy Train	Toy Car	Pump	Valve	Score
MBv2	60.30	57.43	59.43	51.10	53.60	56.17	55.19	56.01
+ CL	60.61	58.87	60.70	50.92	52.51	56.90	54.38	56.18
+ MF	64.08	65.38	59.83	49.69	55.38	59.50	53.74	57.75
+ CL, MF	64.45	67.16	58.66	51.89	56.15	57.27	53.46	57.99

Table 2: Main results proposed in our work for the DCASE 2021 Task2 challenge on the held-out development dataset in regards to the main evaluation metric Ω (see [16]). "C" represents adding contrastive learning and "M" the addition of median filtering. Note that a single model is trained for each machine type.

The two models used in this work are described. First, our AE is the same as the one provided by the challenge baseline. Each hidden block has 128 units except for the bottleneck block, which has 8 units. Second, the MobileNetV2 (MBv2) architecture is directly taken from [19], where our approach differs from the standard architecture by using global average and max pooling (GAMP) as our aggregation method compared to the standard global average pooling (GAP). During training, both the AE and MBv2 models are jointly optimized given the total loss Equation (5), whereas during evaluation only the MBv2 model is used.

[1]The source code is available at `https://github.com/bibiaaaa/SmallRice_DCASE2021Challenge`

4. RESULTS

Our model's performance on the held-out development set is displayed in Table 2. As it can be seen, our MBv2 model trained in the proposed training framework shows improvement over the baseline model in some machine types such as "Fan" and "Gearbox".

For the DCASE challenge, we trained an EfficientNet-B0 based model under our proposed training framework along with median filter and other data augmentation techniques such as Mixup [13] and time masking. For the challenge, our method ranked 9th out of 27 participated methods. As shown in Table 1, our method lacks behind an absolute of 6 % against the winning system.

It is worth mentioning that Table 3 shows that our method performed best on the Fan dataset, especially from the perspective of pAUC metric, leading by a large margin of around 9% compared to the 2nd result. We believe that it contributes to the median filter applied on the log-mel spectrogram along the time axis since it can erase short-time noise and improve the generalization ability of the model.

Model	Fan (AUC)	Fan (pAUC)
AE Baseline	60.68	50.50
MBv2 Baseline	64.96	58.14
2nd	90.22	71.19
3rd	88.98	70.20
4th	88.09	70.84
Ours	**90.68**	**79.99**

Table 3: Top 5 best results in the Fan dataset in the challenge. Our result ranks 1st both in AUC and pAUC.

5. CONCLUSION

This paper proposes a novel contrastive loss training framework for anomaly sound detection. Experimental results indicate that the MobileNetV2-based model trained under our proposed training

framework exceeds the baseline model for some machine types in the DCASE 2021 challenge task 2, while no additional parameters are introduced during inference. Notably, our model achieves the best performance for the "Fan" machine type. We conclude that anomaly sounds greatly vary between different machine types, thus finding a universal anomaly sound detection method suitable for machine condition monitoring is still a problem worthy of research.

6. REFERENCES

[1] E. Dorj and E. Altangerel, "Anomaly detection approach using hidden markov model," in *Ifost*, vol. 2. IEEE, 2013, pp. 141–144.

[2] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "Probabilistic novelty detection for acoustic surveillance under real-world conditions," *IEEE Transactions on Multimedia*, vol. 13, no. 4, pp. 713–719, 2011.

[3] A. Sasou and N. Odontsengel, "Acoustic novelty detection based on ahlac and nmf," in *2012 International Symposium on Intelligent Signal Processing and Communications Systems*, 2012, pp. 872–875.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[5] P. Primus, "Reframing unsupervised machine condition monitoring as a supervised classification task with outlier-exposed classifiers," DCASE2020 Challenge, Tech. Rep., July 2020.

[6] R. Giri, S. V. Tenneti, K. Helwani, F. Cheng, U. Isik, and A. Krishnaswamy, "Unsupervised anomalous sound detection using self-supervised classification and group masked autoencoder for density estimation," DCASE2020 Challenge, Tech. Rep., July 2020.

[7] P. Daniluk, M. Gozdziewski, S. Kapka, and M. Kosmider, "Ensemble of auto-encoder based systems for anomaly detection," DCASE2020 Challenge, Tech. Rep., July 2020.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *CoRR*, vol. abs/2002.05709, 2020. [Online]. Available: https://arxiv.org/abs/2002.05709

[9] L. Wang, K. Kawakami, and A. van den Oord, "Contrastive Predictive Coding of Audio with an Adversary," in *Proc. Interspeech 2020*, 2020, pp. 826–830. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-1891

[10] L. Wang, P. Luc, A. Recasens, J. Alayrac, and A. van den Oord, "Multimodal self-supervised learning of general audio representations," *CoRR*, vol. abs/2104.12807, 2021. [Online]. Available: https://arxiv.org/abs/2104.12807

[11] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised Contrastive Learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 18 661–18 673. [Online]. Available: http://arxiv.org/abs/2004.11362

[12] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised Data Augmentation for Consistency Training," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2019, pp. 6256–6268. [Online]. Available: http://arxiv.org/abs/1904.12848

[13] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[14] D. S. Park, W. Chan, Y. Zhang, C. C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2019-September. International Speech Communication Association, 2019, pp. 2613–2617.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8026–8037.

[16] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and discussion on dcase 2021 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring under domain shifted conditions," *arXiv preprint arXiv:2106.04492*, 2021.

[17] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, "Toyadmos2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions," *arXiv preprint arXiv:2106.02369*, 2021.

[18] R. Tanabe, H. Purohit, K. Dohi, T. Endo, Y. Nikaido, T. Nakamura, and Y. Kawaguchi, "Mimii due: Sound dataset for malfunctioning industrial machine investigation and inspection with domain shifts due to changes in operational and environmental conditions," *arXiv preprint arXiv:2105.02702*, 2021.

[19] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

A LIGHTWEIGHT APPROACH FOR SEMI-SUPERVISED SOUND EVENT DETECTION WITH UNSUPERVISED DATA AUGMENTATION

Heinrich Dinkel, Xinyu Cai, Zhiyong Yan, Yongqing Wang, Junbo Zhang, Yujun Wang

Xiaomi Corporation, Beijing, China
{dinkelheinrich,caixinyu,yanzhiyong, wangyongqing3,zhangjunbo1,wangyujun}@xiaomi.com

ABSTRACT

This paper describes our submission to the DCASE 2021 challenge. Different from most other approaches, our work focuses on training a lightweight and well-performing model which can be used in real-world applications. Compared to the baseline, our model only contains 600k parameters, resulting in a size of 2.7 Mb on disk, making it viable for applications on low-resource devices such as mobile phones. As a novelty, our approach uses unsupervised data augmentation (UDA) as the primary consistency criterion, which we show can achieve competitive performance to the more common mean teacher paradigm. Our submitted results on the validation set result in a single model peak performance of 36.91 PSDS-1 and 57.17 PSDS2, outperforming the baseline by an absolute of 2.7 and 5.0 points respectively. The best submitted ensemble system using a 5-way fusion achieves a PSDS-1 of 38.23 and PSDS-2 of 62.29 on the validation dataset. Our system ranks 7th in the official DCASE2021 Task4 challenge ranking and is the best performing model without post-processing while also having the least amount of parameters (3.4 M) by a large margin. Post-challenge evaluation reveals that by applying simple median post-processing, our approach achieves comparable performance to the 5th place.

Index Terms— Semi-supervised learning, Convolutional recurrent neural networks, Weakly supervised learning, unsupervised domain adaptation.

1. INTRODUCTION

This work focuses on modeling audio signals for sound event detection (SED). The main objective within SED is to categorize (i.e., tag) an event, with its respective on- and offsets. A core difficulty in this task is that multiple sound events can simultaneously occur during a time window.

One possible method to train a SED model is by using fully supervised labels, where on- and offsets for each event of interest are provided. However, obtaining fully supervised labels via manual labeling is expensive and thus might be a hindrance for SED systems at scale. To the best of our knowledge, there currently only exists a single large-scale manual labeled dataset, being Audioset [1], which provides full annotation for around 200 hours of data.

This paper focuses on semi-supervised sound event detection, where the provided training data is largely incomplete. Specifically, the DCASE2021 Task4 challenge focuses on low-cost sound event detection, where only a small fraction of data (4 hours) is manually weakly annotated. All other available data sources are either generated or do not contain labels.

Currently, SED can be used for a variety of applications, query-based sound retrieval [2, 3], smart cities, and homes [4, 5], voice activity detection [6, 7] as well as an important component of audio captioning [8, 9]. Most current approaches within SED utilize neural networks, in particular convolutional neural networks [10] (CNN), convolutional recurrent neural networks [11] (CRNN) and other models such as transformers and conformers [12, 13].

CNN models excel at audio tagging [14] and scale with data, yet falling behind CRNNs and transformer approaches in onset and offset estimations [15].

1.1. Problem statement

In the following, assume that x is an input (either raw-waveform or some spectrogram) and $\hat{y}$ is a predicted label.

Weakly supervised SED models commonly have two outputs: A clip-level prediction head $C(x) \mapsto \hat{y} \in \{0,1\}^E$ and a frame-level output $F(x) \mapsto \hat{y}_t \in \{0,1\}^E, t = 1, \dots, T$ for a frame at time t with E events. Both of these heads are directly connected via an aggregation function: $C(\cdot) = \text{agg}(F(\cdot))$, which summarizes the frame-level predictions to a single clip-level response. When training in strictly weakly supervised fashion, only the clip-level prediction head C can be learned, while F needs to be inferred by the model.

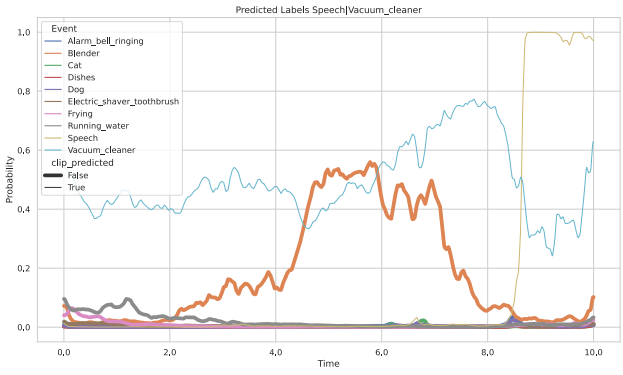


Figure 1: Inconsistent predictions between the two output heads in weakly-supervised SED are tackled in this work. The clip-level prediction $\hat{y}$ estimates the presence of "Speech" and "Vaccum cleaner", but the frame-level output $\hat{y}_t$ additionally predicts the presence of "Blender" (in bold) and some other noisy event outputs.

One of the key problems regarding training of weakly supervised SED models is that both heads can predict contradictory results since only errors in C are back-propagated, while F cannot be directly controlled. For example, the frame head F might predict

the presence of a sound event e during some time frame, due to factors such as noise or general similarity of a sound event to another (e.g., Blender, Vacuum Cleaner), while the clip head C predicts that e is not present. We show an example of this behavior in Figure 1, which is a prediction done by our baseline model.

Since in this challenge, the only human-annotated training data is provided on clip-level, meaning that the clip head C should provide reliable outputs, additional predictions from F can be considered as an inconsistency between both heads. In order to mitigate the inconsistency problem, we propose a simple learnable clip-smoothing algorithm.

2. PROPOSED APPROACH

2.1. Learnable clip-smoothing

We propose learnable clip-smoothing to combat the problem of inconsistent predictions the C and F heads. This technique is identical to clip-thresholding for weakly supervised SED [11], but since the DCASE2021 Task4 dataset provides strong frame-level labels, the clip-smoothing threshold can now be jointly optimized with the weak labels.

In particular, clip-smoothing is computed as in Equation (1), where $\hat{y}(t)^{\dagger}$ is the clip-smoothed output of our model for event e and $\hat{y}_t(e)$ is the model's frame head output (F):

$$\hat{y}_t^{\dagger}(e) = \hat{y}_t(e) * \hat{y}(e). \tag{1}$$

This approach should reduce false alarms, since the clip-level output will squash the frame-level probabilities for any non-occurring event. Subsequently, by using learnable clip-smoothing, the head F will only output events which also have a large probability score (≈ 1) in C.

2.2. Unsupervised data augmentation for consistency training

Many techniques exist to utilize unlabeled data to improve model performance. Mean Teacher (MT) [16] is a popular technique used in recent DCASE challenges [17].

In this work we propose the use of unsupervised domain (UDA) [18] for consistency training in SED. The advantages of UDA are:

1. In the vision domain, UDA has been seen to outperform other unsupervised methods such as MT [18].
2. Since only a single model is trained, performance evaluation is simpler compared to MT, where two models need to be evaluated.
3. We believe that the main contributing factor of MT is that it enables the usage of unlabeled data to improve performance. Our work shows that unsupervised data augmentation is equally effective in boosting performance.

The idea of UDA is to compute a consistency loss for unlabeled data between an augmented and a non-augmented (or differently augmented) sample. To the best of our knowledge, UDA has not been previously used in SED.

$$\begin{aligned} x^{\dagger} &= \text{Aug}(x), \\ \mathcal{M}(x) &\mapsto (\hat{y}, \hat{y}_t), \\ \mathcal{M}(x^{\dagger}) &\mapsto (\hat{y}^{\dagger}, \hat{y}_t^{\dagger}), \\ \mathcal{L}_{\text{UDA}}(x) &= \mathcal{L}_{\text{consistency}}(\hat{y}^{\dagger}, \hat{y}) + \mathcal{L}_{\text{consistency}}(\hat{y}_t^{\dagger}, \hat{y}_t). \end{aligned} \tag{2}$$

The UDA consistency training scheme is defined as in Equation (2). Here, a sample x is fed through a trainable neural network $\mathcal{M}$ where clip ($\hat{y}$) and frame-level ($\hat{y}_t$) predictions are obtained. The consistency between these predictions ($\hat{y}, \hat{y}_t$) and the predictions obtained by augmenting the input sample x denoted as $x^{\dagger}$ and predict ($\hat{y}^{\dagger}, \hat{y}_t^{\dagger}$) is the training objective. Note that in our work, we use UDA for both model heads, whereas it would be possible to use UDA for only weak or strong labels respectively. Also, it is worth mentioning that gradients are not computed during $\mathcal{M}(x)$.

3. EXPERIMENTAL SETUP

Log Mel-spectrogram (LMS) features are chosen as the default front-end feature for the task. Each 64-filter LMS is extracted from a 25 ms window with a stride of 10 ms, resulting in an approximately 1001×64 dimensional input tensor. If segments are shorter than 10 seconds, we zero-pad the input to the longest sample within a batch. During inference, we use a batch size of 1, such that padding has no effect on the final evaluation.

All experiments start with a learning rate of 0.001 and are run for at most 200 epochs, with a linear warmup duration of 20 batches using the Adam optimizer. The learning rate is halved every 1000 batches. Batch sizes are set to be 32 for weak and synthetic data and 64 for unlabeled data. The available weak training data is split into a 90% training and a 10% cross-validation portion. Cross-validation is done on the 10% held-out weak subset with the additional synthetic validation data. The training objective is the sum of the weak F1 and the intersection-F1 score, whereas training is stopped if the model did not improve for 15 epochs. Pytorch [19] was used as the neural network back-bone.[1].

3.1. Dataset

The dataset used in this work is the DCASE2021 dataset, which focuses on sound event detection in domestic environments.

The DCASE 2021 dataset is split into a development (used for training) and an evaluation section. The development set is further split into training and validation sections. The training section contrains three datasets $\mathcal{D}_{weak}, \mathcal{D}_{syn}, \mathcal{D}_{un}$, as seen in Equation (3).

$$\begin{aligned} \mathcal{D}_{weak} &= \{(x_1, y_2), (x_2, y_2), \ldots, (x_N, y_N)\}, \\ \mathcal{D}_{syn} &= \{(x_1, y_2), (x_2, y_2), \ldots, (x_M, y_M)\}, \\ \mathcal{D}_{un} &= \{x_1, \ldots, x_P\}. \end{aligned} \tag{3}$$

Note that the labels for $\mathcal{D}_{weak}$ are provided on clip-level, i.e., $y_j \in \{0,1\}^E, j \leq N$, while labels for $\mathcal{D}_{syn}$ are provided at frame-level, i.e., $y_k \in \{0,1\}^{ET}, k \leq M$ for each timestep in T. The unlabeled dataset contains only samples with target events also seen in the weak training data.

3.2. Model

Our model named CDur is a lightweight (in terms of parameters) 5-layer CRNN directly taken from the previous work in [11].

$$\hat{y} = \frac{\sum_t \hat{y}_t^2}{\sum_t \hat{y}_t} \tag{4}$$

[1]The source is available at `https://github.com/bibiaaaa/SmallRice_DCASE2021Challenge`

CDur subsamples the time-dimension by a factor of 4 and uses linear-softmax [20] as its aggregation method defined in Equation (4). The frame-level output is upsampled by a non-learnable transformation. For more information about the model, please refer to [11]. One of the benefits of the proposed model is its size, it only contains around 600k parameters, making it a lightweight alternative to the larger baseline model (1.1M parameters). Note however, that CDur requires slightly more FLOPs (3.36 GFlops) compared to the baseline (2.97 GFlops).

Three losses are used, one for each respective training data subset. Note that we experimented with additional losses such as asymmetric focal loss (AFL) [21], but did not observe gains in performance.

$$\mathcal{L}_{\text{sup}} = \text{BCE}(\hat{y}, y), \{y, \hat{y}\} \in \mathcal{D}_{weak}, \quad (5)$$

$$\mathcal{L}_{\text{syn}} = \text{BCE}(\hat{y}_t, y_t), \{y_t, \hat{y}_t\} \in \mathcal{D}_{syn}, \quad (6)$$

$$\mathcal{L}_{\text{unsup}} = \mathcal{L}_{\text{UDA}}(x) = \text{BCE}(\hat{y}^{\dagger}, \hat{y}) + \text{BCE}(\hat{y}_t, \hat{y}_t), x \in \mathcal{D}_{un}. \quad (7)$$

The model is optimized using the sums of all introduced losses seen in Equation (8).

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{syn}} + \mathcal{L}_{\text{unsup}} \quad (8)$$

As the default in our work use UDA for both C and F heads. Augmentation in regards to UDA is applied on raw-wave level, where the torchaudio[2] and torch-audiomentations[3] packages are used. Specifically, we apply random *Gain* (in range -20, 10 db), *Polarityinversion* (with both probability 50%), and time masking (zeroing a sequence of at most 2 seconds, similar to SpecAug) to an input sample.

4. RESULTS

We report our results in terms of Event-F1 (E-F1) [22], Intersection-F1 (I-F1), and the two main challenge metrics denoted as PSDS-1 and PSDS-2 [23]. Additionally, we provide the d-prime d' score, which represents our model's capability to detect the presence of an event on clip-level and takes values in range $d' \geq 0$, where higher values are better.

Note that for *all* results, *no* post-processing is used and the Event-F1 score is calculated from the thresholded $\hat{y}_t > 0.5$ frame-predictions.

Data	d'	E-F1	I-F1	**PSDS-1**	**PSDS-2**
Weak	2.28	22.71	49.06	15.17	33.47
+ Syn	2.23	30.39	49.63	19.01	28.12
++ Unlabel	2.47	32.11	52.14	26.87	42.19

Table 1: Baseline results using CDur training with amounts of training data. All results are an average over 5 individual runs on the development dataset. Highlighted scores are the main challenge evaluation metrics. Higher is better.

The baseline experiments using the proposed CDur model can be seen in Table 1. The additional data synthetic data seems to decrease d', which likely stems from the mismatch between the synthetic and real data. With the addition of the unlabeled data, however, d' largely enhances, since the model now has access to larger amounts of real-world samples. This enhancement is then reflected on the PSDS-1 and PSDS-2 scores since the clip-smoothing technique's filtering capability is now enhanced.

Data	d'	E-F1	I-F1	**PSDS-1**	**PSDS-2**
Weak	2.27	22.99	49.14	19.98	46.57
+ Syn	2.21	35.31	54.84	29.85	47.34
++ Unlabel	2.50	37.21	57.12	34.41	54.90

Table 2: Development dataset results using the proposed clip-smoothing with CDur. All results are an average over 5 individual runs. Highlighted scores are the main challenge evaluation metrics. Higher is better.

Our results with the proposed clip-smoothing technique can be observed in Table 2. Comparing to our baseline, clip-smoothing leads to a large improvement for all metrics, leading to a comparable performance in terms of PSDS-1 and -2 against the strong baseline.

4.1. Data Augmentation

Two augmentation methods, namely SpecAug [24] and Mixup are used to enhance performance. The results can be seen in Table 3. Adding SpecAug to our model training decreases all metrics except PSDS-2, while the addition of SpecAug + Mixup shows improvements for both PSDS-1 and PSDS-2 scores. In the following, every experiment denoted as Aug uses SpecAug and Mixup as default.

Aug	d'	E-F1	I-F1	**PSDS-1**	**PSDS-2**
Base	2.50	37.21	57.12	34.41	54.90
+ SpecAug	2.64	35.68	57.06	32.60	56.26
++ Mixup	2.60	35.76	56.01	34.59	57.11

Table 3: Results with additional data augmentation in form of SpecAug and Mixup on the development dataset. All results are an average over 5 individual runs. Highlighted scores are the main challenge evaluation metrics.

4.2. Ensemble and submissions

The ensemble submissions seen in Table 4 named S1, S2 and S3 are frame-level averaged over the respective single models, which are:

- Aug, which uses clip-smoothing and additional specaug + mixup during training (see Table 3).
- *Heavy* uses much stronger augmentations during UDA than the default ones. Time Masking with a maximal length of 5s as well as a 70 % probability to apply volume gain in the range of -20 to 20 dB.
- *MSE* uses the mean square error criterion for UDA training instead of the default BCE.
- *WeakShift* Uses an additional augmentation via shifting of the time domain (with rollover) during UDA training. Note that the training criterion becomes $\mathcal{L}_{\text{UDA}} = \text{BCE}(\hat{y}^{\dagger}, \hat{y})$.
- *Sub-8* subsamples the time dimension by a factor of 8, leading to an output resolution of 80ms instead of 40ms.

[2] https://github.com/pytorch/pytorch

[3] https://github.com/asteroid-team/torch-audiomentations

Model	d'	E-F1	I-F1	**PSDS-1**	**PSDS-2**
Baseline	-	40.10	76.60	34.20	52.70
Aug (A)	2.66	36.80	58.94	33.63	57.43
Heavy (B)	2.56	39.02	58.09	35.21	58.00
MSE (C)	2.46	35.08	56.69	34.24	55.07
WeakShift (D)	2.50	39.29	59.02	36.91	57.17
Sub-8 (E)	2.66	36.05	57.17	33.00	59.38
S1 (A+B+C)	2.70	40.89	59.13	37.25	61.99
S2 (S1 + D)	2.70	40.90	59.61	**38.23**	62.29
S3 (S2 + E)	**2.75**	**41.06**	59.71	38.13	**62.98**

Table 4: Performance for the best single model results on the development dataset and the submitted ensemble models. Best results are highlighted in bold. Ensembles are generated by averaging the frame-level outputs of each respective model.

Compared to the baseline, our model falls behind in terms of Intersection-F1 and Event-F1, which is likely due to our neglect of post-processing methods largely affecting those metrics. However, in terms of PSDS, our model largely outperforms the baseline approach by an absolute of at least 3 and 9 points, respectively. Our submissions to the challenge include the ensemble systems S1, S2 and S3 as well as our best performing single model (D).

4.3. Challenge results

After the challenge ended, the results of all teams participated were published. Within the challenge, our method scored the 7th overall place in terms of the averaged PSDS-1 and PSDS-2 metrics as well as the average of both metrics (PSDS-Avg). A comparison of our method against other challenge participants can be seen in Table 5. Notably, our approach is the best performing approach in the challenge without requiring post-processing.

Model	**PSDS-1**	**PSDS-2**	**PSDS-Avg**	Post
Baseline	31.5	54.7	43.1	Median
1st	**45.2**	**74.6**	**59.9**	
2nd	44.2	67.4	55.8	
3rd	39.9	71.5	55.7	
3rd	41.9	68.6	55.2	Median
4th	41.6	63.7	52.6	
5th	41.3	58.6	49.9	
6th	37.0	62.6	49.8	
S1	36.1	58.4	47.2	
S2	37.3	58.5	47.9	-
S3	37.0	59.6	48.3	
S4 (Single)	33.9	50.4	42.1	
Ours (best)	**37.3**	**59.6**	**48.4**	-

Table 5: Performance of our models in comparison to other participants in the challenge on the official evaluation dataset. Best models with postprocessing (Post) using median filtering and without are displayed in bold.

4.4. Post-challenge post-processing

Since all other participants opt to use median filtering, we also provide our results on the development set in comparison to theirs using an adaptive window size for each event. The window sizes are estimated from the validation dataset, where one-third of each event's average duration is used as the window sizes.

Model	#Param (M)	**PSDS-1**	**PSDS-2**	**Score**	Single?
1st	14.3	**45.2**	**74.6**	**1.40**	N
2nd	20.2	44.2	67.4	1.32	Y
3rd	79.2	33.9	71.5	1.29	N
3rd	50.0	41.9	68.6	1.29	N
4th	119.8	41.6	63.7	1.24	N
S3	**3.4**	38.2	65.4	1.20	Y
S2	**2.7**	37.9	64.3	1.19	Y
5th	8.5	41.3	58.6	1.19	Y
S1	**2.0**	36.1	64.3	1.16	Y
6th	6.7	37.0	62.6	1.16	Y

Table 6: Post-challenge performance of our models in comparison to other participants using median post-processing on the evaluation set. "Single" refers to whether the results stem from a single submission or two different submissions. "Score" represents the challenge ranking score, where 1.0 is the challenge baseline. If multiple models were used, the reported parameter count represent the sum of each individual model's parameters.

As we can see from the results in Table 6, our model compares favorably against other participants in terms of parameter count to performance ratio[4]. Further, the submission S3 achieves a noticeable boost of 2 and 5 points in terms of PSDS-1 and PSDS-2 scores respectively on the evaluation dataset when using median filtering. Within the top-performing submissions, our proposed method is the most lightweight by a large margin as seen in Table 6, achieving comparable performance to the 4th place, while using only a fraction (2 %) of its parameters. Finally, our method ranks overall second if we only compare scores obtained by a single submission i.e., a model which performs well in terms of both PSDS-1 and PSDS-2 scores.

5. CONCLUSION

This paper proposes our submission to the DCASE2021 Task4 challenge. The approach uses clip-smoothing in combination with a small parameter model to outperform the provided baseline in terms of PSDS-1 and PSDS-2 scores. Our best single model achieves a PSDS-1 of 36.91 and 33.9 and a PSDS-2 of 57.17 and 50.4 on the validation and evaluation datasets, respectively. Moreover, our 4-model ensemble approach achieves a PSDS-1 of 38.23 and a PSDS-2 of 62.29, significantly outperforming the challenge baseline by an absolute of 4.03 and 9.6 points respectively. In terms of the official evaluation, our method scored seventh place, while being the only top-ranking method not using post-processing. When utilizing common adaptive median post-processing our approach achieves comparable performance to the 5th place, while having the fewest parameters amongst all top-ranked methods.

6. REFERENCES

[1] S. Hershey, D. P. W. Ellis, E. Fonseca, A. Jansen, C. Liu, R. Channing Moore, and M. Plakal, "The Benefit of

[4]We thank Romain Serizel for re-evaluating the post-processed results.

Temporally-Strong Labels in Audio Event Classification," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Institute of Electrical and Electronics Engineers (IEEE), may 2021, pp. 366–370.

[2] F. Font, G. Roma, and X. Serra, *Sound Sharing and Retrieval*. Springer International Publishing, 2018, pp. 279–301.

[3] A.-M. Oncescu, A. S. Koepke, J. F. Henriques, Z. Akata, and S. Albanie, "Audio Retrieval with Natural Language Queries," in *Proc. Interspeech 2021*, 2021, pp. 2411–2415.

[4] J. P. Bello, C. Mydlarz, and J. Salamon, *Sound Analysis in Smart Cities*. Cham: Springer International Publishing, 2018, pp. 373–397.

[5] S. Krstulović, *Audio Event Recognition in the Smart Home*. Cham: Springer International Publishing, 2018, pp. 335–371.

[6] Y. Chen, H. Dinkel, M. Wu, and K. Yu, "Voice activity detection in the wild via weakly supervised sound event detection," *Proc. Interspeech 2020*, pp. 3665–3669, 2020.

[7] H. Dinkel, S. Wang, X. Xu, M. Wu, and K. Yu, "Voice Activity Detection in the Wild: A Data-Driven Approach Using Teacher-Student Training," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1542–1555, 2021.

[8] X. Xu, H. Dinkel, M. Wu, Z. Xie, and K. Yu, "Investigating local and global information for automated audio captioning with transfer learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 905–909.

[9] M. Wu, H. Dinkel, and K. Yu, "Audio caption: Listen and tell," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 830–834.

[10] L. Lin, X. Wang, H. Liu, and Y. Qian, "Specialized Decision Surface and Disentangled Feature for Weakly-Supervised Polyphonic Sound Event Detection," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 28, pp. 1466–1478, may 2020.

[11] H. Dinkel, M. Wu, and K. Yu, "Towards duration robust weakly supervised sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 887–900, 2021.

[12] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, "Sound Event Detection of Weakly Labelled Data with CNN-Transformer and Automatic Threshold Optimization," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 28, pp. 2450–2460, 2020.

[13] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Convolution-augmented transformer for semi-supervised sound event detection," DCASE2020 Challenge, Tech. Rep., June 2020.

[14] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 28, pp. 2880–2894, dec 2020.

[15] N. Turpault, R. Serizel, and E. Vincent, "Limitations of weak labels for embedding and tagging," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 131–135.

[16] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1195–1204.

[17] J. Yan, Y. Song, L.-R. Dai, and I. McLoughlin, "Task-Aware Mean Teacher Method for Large Scale Weakly Labeled Semi-Supervised Sound Event Detection," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings 2020*. Institute of Electrical and Electronics Engineers (IEEE), apr 2020, pp. 326–330.

[18] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised Data Augmentation for Consistency Training," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2019, pp. 6256–6268.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8026–8037.

[20] Y. Wang, J. Li, and F. Metze, "A Comparison of Five Multiple Instance Learning Pooling Functions for Sound Event Detection with Weak Labeling," *ICASSP 2019 - ICASSP 2019 IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May, pp. 31–35, oct 2019.

[21] K. Imoto, S. Mishima, Y. Arai, and R. Kondo, "Impact of sound duration and inactive frames on sound event detection performance," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 860–864.

[22] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences (Switzerland)*, vol. 6, no. 6, p. 162, may 2016.

[23] C. Bilen, G. Ferroni, F. Tuveri, J. Azcarreta, and S. Krstulovic, "A Framework for the Robust Evaluation of Sound Event Detection," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, may 2019, pp. 61–65.

[24] D. S. Park, W. Chan, Y. Zhang, C. C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2019-September. International Speech Communication Association, 2019, pp. 2613–2617.

IMPROVING THE PERFORMANCE OF AUTOMATED AUDIO CAPTIONING VIA INTEGRATING THE ACOUSTIC AND SEMANTIC INFORMATION

Zhongjie ye[1], *Helin Wang*[1], *Dongchao Yang*[1], *Yuexian Zou*[1,2,*]

[1] ADSPLAB, School of ECE, Peking University, Shenzhen, China
[2] Peng Cheng Laboratory, Shenzhen, China
{zhongjieye@stu.pku.edu.cn, wanghl15@pku.edu.cn
dongchao98@stu.pku.edu.cn, zouyx@pku.edu.cn}

ABSTRACT

Automated audio captioning (AAC) has developed rapidly in recent years, involving acoustic signal processing and natural language processing to generate human-readable sentences for audio clips. The current models are generally based on the neural encoder-decoder architecture, and their decoder mainly uses acoustic information that is extracted from the CNN-based encoder. However, they have ignored semantic information that could help the AAC model to generate meaningful descriptions. This paper proposes a novel approach for automated audio captioning based on incorporating semantic and acoustic information. Specifically, our audio captioning model consists of two sub-modules. (1) The pre-trained keyword encoder utilizes pre-trained ResNet38 to initialize its parameters, and then it is trained by extracted keywords as labels. (2) The multi-modal attention decoder adopts an LSTM-based decoder that contains semantic and acoustic attention modules. Experiments demonstrate that our proposed model achieves state-of-the-art performance on the Clotho dataset. Our code can be found at https://github.com/WangHelin1997/DCASE2021_Task6_PKU.

Index Terms— Audio captioning, pre-training, multi-modal attention, keyword classification

1. INTRODUCTION

Automated audio captioning (AAC) is a cross-modal task of generating a natural language description for an audio clip. It is different from audio tagging (AT), acoustic scene classification (ASC) and automatic speech recognition. The purpose of AAC is not only to analyze acoustic scenes, events, and concepts in a given audio clip, but also to find the relationships among them to produce human-readable sentences. Applications of automated audio captioning are diverse such as assisting the hearing impaired people by converting audio signals into a text, and content-based audio retrieval task which uses the free-form natural language queries to retrieve the audio [1].

AAC has aroused a lot of interest among researchers since the Detection and Classification of Acoustic Scenes and Events (DCASE) 2020 challenge. Nowadays, the mainstream framework is based on neural encoder-decoder systems which have achieved success in some relevant fields such as image captioning [2]. The current AAC models consist of a convolutional neural network (CNN) encoder and a recurrent neural network (RNN) (or Transformer) decoder with an attention mechanism. The inputs used could be log-mel energies, Mel-Frequency Cepstral Coefficients(MFCCs), or other acoustic features which are extracted from raw audio clips. They are firstly encoded by a CNN encoder into a set of feature vectors. Then, they are decoded into sentences by an RNN-based or Transformer-based decoder with (or without) an attention mechanism.

Over past few years, there are amounts of methods proposed in AAC task [3, 4, 5, 6, 7] based on neural encoder-decoder systems. M. Wu *et al.* [3] straightly takes the mean of the feature vectors that are the outputs of the encoder in the time dimension, and uses them as the input of the decoder. H. Wang *et al.* [5] proposed a temporal attention mechanism in the decoder, which could utilize more acoustic information for each time step. In contrast to previous work in AAC, Y. Wu *et al.* [4] and X. Xu *et al.* [6] explore transfer learning method to help AAC models to get better performance. The strategy of their proposed methods could be divided into two stages. In the first stage, a tagging system is pre-trained by ASC or AT task. Then the parameters of the audio encoder are initialized by the pre-trained tagging system. In the second stage, the whole AAC model is trained end-to-end by minimizing the cross-entropy (CE) loss. With these methods mentioned above, they generally only consider acoustic information while ignoring semantic information when the AAC model generates sentences. Specifically, the semantic information could contain keywords that are from the encoder, previously predicted words in the decoding time, and so on. In this paper, we introduce semantic information with acoustic information to assist the decoder to generate higher quality sentences. Furthermore, to better make use of semantic and acoustic information, we propose a novel multi-modal attention mechanism. In summary, our contributions are as follows:

1. We propose a **m**ulti-modal **a**ttention-based **a**udio **c**aptioning model with a pre-trained keyword encoder, named **MAAC**. It could utilize both acoustic and semantic information to generate the description. The semantic information includes keywords from the pre-trained keyword encoder and the previously decoding information from the decoder.
2. Our MAAC achieves a new state-of-the-art performance on the Clotho dataset. We present the ablation analysis of the components of our MAAC and demonstrate that semantic information could improve the performance of the AAC model.

The organization of the paper is as follows. Section 2 introduces our proposed model. We present our experimental results

*Yuexian Zou is the corresponding author.

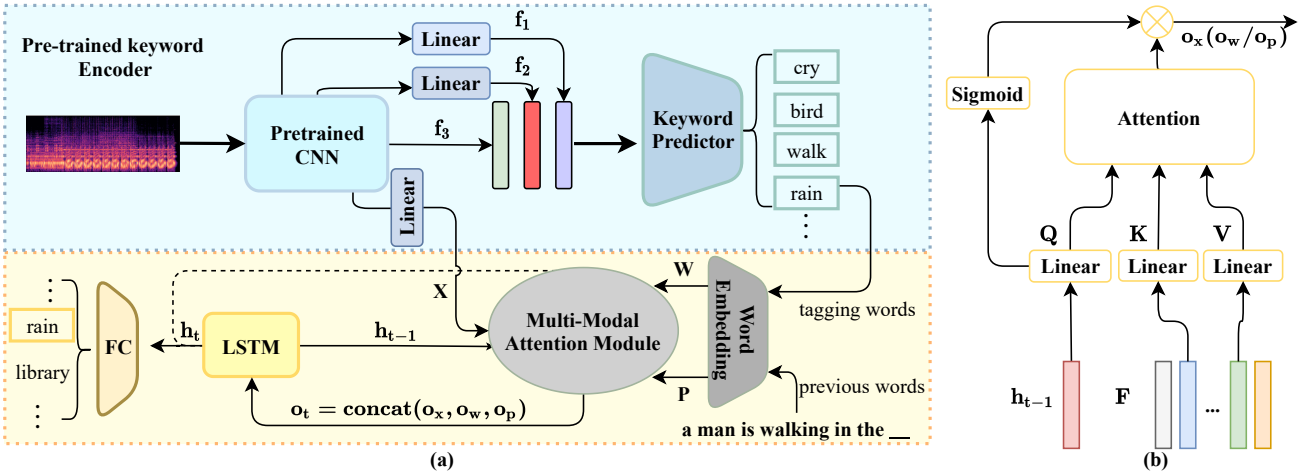


Figure 1: (a) Our proposed MAAC includes two submodules: the pre-trained keyword encoder is on the top and the LSTM-based decoder with a multi-modal attention module is on the bottom. (b) The architecture of the attention mechanism. F could represent acoustic features or semantic features.

and evaluations in Section 3. Finally, we give concluding remarks and possible future directions in Section 4.

2. SYSTEM ARCHITECTURE

In this section, our proposed MAAC is introduced and its architecture is shown in Figure 1. Specifically, our MAAC consists of two submodules: a pre-trained keyword encoder and an LSTM-based decoder with a multi-modal attention module. In the following subsections, we will introduce details about it.

2.1. Pre-trained Keyword Encoder

The CNN encoder, which is widely used in the AAC challenge [4, 5], plays an important role in extracting acoustic information from raw audios. In this work, we extract keywords from captions as training labels and use the pre-trained ResNet38[1] [8] that performs well in the AudioSet dataset [9] as our backbone network.

Constructing Audio-Keyword Training Pairs Firstly, Natural Language Toolkit (NLTK[2]) is a powerful open-source tool applied to extract words from each caption. We choose the nouns and verbs to construct the keyword table by getting rid of some useless words such as *make*, *go*, *others*, etc. The verbs in the keyword table are transformed into their original forms and the nouns are not changed, because plural forms of the nouns have different meanings. Then, we choose N keywords with the highest frequency from the modified keyword table and use them as labels for pre-training.

We combine all the keywords from the 5 captions of each audio clip to form the training label which is a multi-hot vector. Each word of captions is transformed into its original forms according to the above rules. When a word occurs in the keyword table, the corresponding position of the multi-hot vector is set to 1, otherwise 0.

[1] https://github.com/qiuqiangkong/audioset_tagging_cnn

[2] https://github.com/nltk/nltk

Training the Keyword Encoder As Figure 1 illustrates, the pre-trained ResNet38 is used as our backbone, which consists of 6 convolutional blocks. We refine it with a feature hierarchy structure to combine multi-level features, *i.e.* the features after the third, fourth, and last convolution block. Then all of them are passed into different linear layers after the global average pooling (GAP) method to obtain f_1, f_2 and f_3. Finally, we use them to obtain the predictions $\hat{y} \in \mathbb{R}^N$ and N is the number of keywords.

$$\hat{y} = \sigma(Linear(concat(f_1, f_2, f_3))) \tag{1}$$

where σ denotes sigmoid activation function. Given the ground-truth $y \in \mathbb{R}^N$, the pre-trained keyword encoder could be optimized by minimizing the binary cross-entropy loss:

$$\mathcal{L}_{bce}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} y(i) log\ \hat{y}(i) \tag{2}$$

2.2. Multi-modal Attention Decoder

Unlike the existing audio captioning models, we further incorporate acoustic with semantic information into generating captions: we propose a multi-modal attention module to incorporate them. The high-level representation of acoustic information denoted as $\boldsymbol{X} = \{x_1, ..., x_L\} \in \mathbb{R}^{L \times C_1}$, is the output of a linear layer whose input is the output of the last convolution block of the pre-trained keyword encoder. The semantic features contain the keywords $\boldsymbol{W} = \{w_1, ..., w_K\}$ that is the K outputs of the pre-trained keyword encoder, and the previously predicted words $\boldsymbol{P} = \{p_1, ..., p_{t-1}\}$ that contain all the generated words before time step t. Both of them are transformed into continuous vectors by a randomly initialized embedding layer **Emb**, $\boldsymbol{W} \in \mathbb{R}^{K \times C_2}$ and $\boldsymbol{P} \in \mathbb{R}^{(t-1) \times C_2}$. The implementation process of the multi-modal attention module is as follows.

Firstly, all of them are transformed into the same latent space, where X is turned to $\hat{\boldsymbol{X}} \in \mathbb{R}^{T \times C}$, W becomes $\hat{\boldsymbol{W}} \in \mathbb{R}^{K \times C}$ and P becomes $\hat{\boldsymbol{P}} \in \mathbb{R}^{(t-1) \times C}$. Then the hidden states as intermediaries

Table 1: Single-model performances on the Clotho [10] evaluation splits in the CE and RL training period. B1, B4, RG, ME, CD, SP, and SD denote BLEU-1, BLEU-4, METEOR, ROUGE-L, CIDEr-D, SPICE, and SPIDEr, respectively. For all metrics, higher values indicate better performance.

	Cross-entropy							CIDEr-D optimization						
Model	B1	B4	RG	ME	CD	SP	SD	B1	B4	RG	ME	CD	SP	SD
Baseline [10]	37.8	1.7	26.3	7.8	7.5	2.8	5.1	-	-	-	-	-	-	-
TAM [5]	48.9	10.7	32.5	14.8	25.2	9.1	17.2	-	-	-	-	-	-	-
TM [4]	53.4	15.1	35.6	16.0	34.6	10.8	22.7	-	-	-	-	-	-	-
UNIS's model [11]	-	-	-	-	-	-	-	62.5	17.8	40.1	17.6	42.8	12.6	27.7
SJTU's model [12]	56.5	15.5	37.4	17.4	39.9	11.9	25.9	64.0	16.3	40.4	17.8	44.9	12.3	28.6
MAAC (Ours)	57.7	17.4	37.7	17.4	41.9	11.9	26.9	**64.8**	**18.1**	**40.8**	**19.0**	**49.1**	**13.1**	**31.1**

Table 2: Settings and results of ablation studies. The results are reported after CE training stage. SAM denotes the semantic attention module.

Model	B4	CD	SD
Base	16.5	40.6	26.4
+ Previously predicted words	17.1	41.1	26.4
+ Keywords	can not converge		
+ Both (w/o sharing SAM)	16.8	41.1	26.7
proposed MAAC	**17.4**	**41.9**	**26.9**

connect $\hat{\boldsymbol{X}}$, $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{P}}$, through a multi-modal attention mechanism that is shown in Figure 2. Taking the acoustic information for example: given the previous LSTM hidden state h_{t-1}, we use a single fully-connected layer followed by a softmax function to generate the attention distributions α of acoustic features in the time dimension. Finally, the gated linear unit (GLU) is applied to the output of the attention module, to control how much information should flow into the next layer. Formula (3)-(5) are the definitions of the acoustic attention module $\boldsymbol{\Psi_x}$:

$$\boldsymbol{A} = ReLU((\hat{\boldsymbol{X}}\boldsymbol{W_i^T} + b_i) \oplus (h_{t-1}\boldsymbol{W_s^T} + b_s)) \tag{3}$$

$$\alpha = softmax(\boldsymbol{A}\boldsymbol{W_n} + b_n) \tag{4}$$

$$o_x = GLU([\hat{\boldsymbol{X}} \otimes \boldsymbol{\alpha}, h_{t-1}]) \tag{5}$$

where $\boldsymbol{W_s} \in \mathbb{R}^{M\times H}$, $\boldsymbol{W_i} \in \mathbb{R}^{M\times C}$, $\boldsymbol{W_n} \in \mathbb{R}^{M}$ are transformation matrixes that map acoustic features and hidden states to the same dimension. Here are $b_s \in \mathbb{R}^M$, $b_i \in \mathbb{R}^M$, and $b_n \in \mathbb{R}^1$. We denote $\oplus$ as the element-wise addition of a matrix and a vector, and $\otimes$ as the element-wise multiplication of a matrix and a vector. We choose the GLU operation to obtain the output $o_x \in \mathbb{R}^C$, which implements a simple gating mechanism over the output $\mathcal{Y} = [\mathcal{A}, \mathcal{B}] \in \mathbb{R}^{2d}$:

$$GLU([\mathcal{A}, \mathcal{B}]) = \mathcal{A} \otimes \sigma(\mathcal{B}) \tag{6}$$

where $\mathcal{A} \in \mathbb{R}^d, \mathcal{B} \in \mathbb{R}^d$ are the inputs to the non-linearity, and the output $GLU([\mathcal{A}, \mathcal{B}]) \in \mathbb{R}^d$ is half the size of $\mathcal{Y}$ [13].

As for the semantic information, the same structure of the attention module is applied to keywords and previously predicted words, and the outputs are $o_w \in \mathbb{R}^C$ and $o_p \in \mathbb{R}^C$ respectively. Note each part of semantic information shares an attention module. We add o_x, o_w, o_p with w_{t-1} which is a predicted word of the last time step to obtain the output o_t. Then, o_t and h_{t-1} are sent to calculate the hidden state h_t which is used to predict word probability distribution v_t. Finally, the current word w_t is chosen from v_t with the highest probability and added to previously predicted words $\boldsymbol{P}$ for the next iteration of LSTM. Formula (7) is the operation of the multi-modal attention module described above:

$$\begin{aligned} h_0 &= GAP(\hat{\boldsymbol{X}}) \\ h_t &= LSTM(h_{t-1}, Add(o_x, o_w, o_p, \mathbf{Emb}(w_{t-1}))) \\ v_t &= Softmax(Linear(h_t)) \end{aligned} \tag{7}$$

where h_0 represents the global information of acoustic features in the time dimension. $v_t \in \mathbb{R}^{|\Sigma|}$ is a probability vector, and $|\Sigma|$ is a predefined dictionary including all words.

3. EXPERIMENT

3.1. Dataset and Experiment Setup

Clotho v2 We evaluate our proposed method on the Clotho v2 dataset [10], which is published in DCASE 2020 and expanded in DCASE 2021. Nowadays it contains 5,929 audio clips labeled with 5 captions for each, including 3,839 training, 1,045 validation, and 1,045 testing audio clips. We convert all sentences to lower case and remove all punctuation marks, ending up with a vocabulary $|\Sigma|$ of 4368 words including special tokens "BOS", "EOS", and "PAD". For evaluation, we employ standard evaluation metrics: BLEU [14], ROUGE-L [15], METEOR [16], CIDEr-D [17], SPICE [18] and SPIDEr that is the mean of CIDEr-D and SPICE. All metrics are computed with the audio captioning evaluation tool[3].

Implementation Details We choose $N = 300$ keywords for pre-training encoder, $K = 5$ keywords and the dimension of fully-connected layers C_1, C_2 and C are 512. The decoder LSTM has 512 hidden units, word embedding size is also set to 512. To mitigate overfitting, dropout regularization [19] is used in the word embedding layer with a rate of 0.5, and the word classification layer with a rate of 0.25.

The training strategy of the MAAC could be divided into two stages: encoder pre-training and the whole MAAC model training. In the phase of training the encoder, firstly the CNN backbone is frozen up, trained with the initial learning rate of 1×10^{-3} for 80

[3] https://github.com/audio-captioning/caption-evaluation-tools

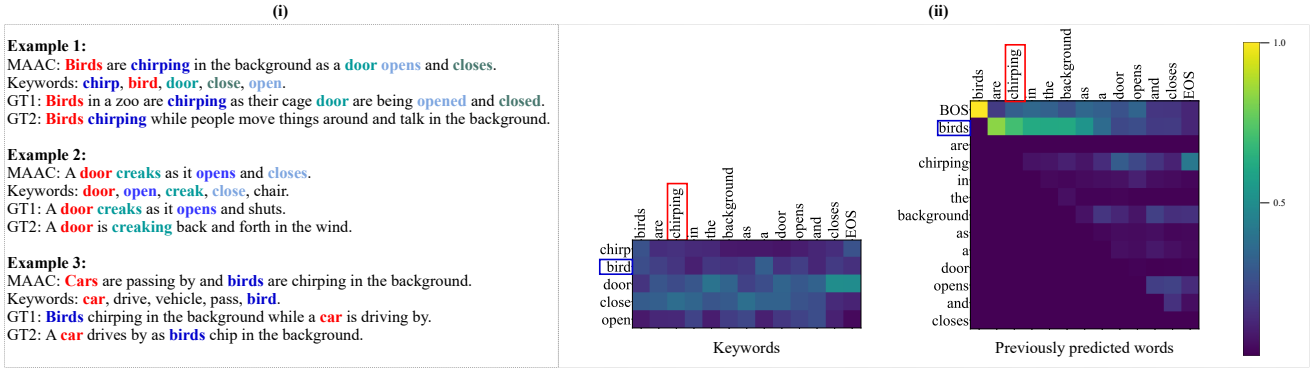


Figure 2: (i) It shows some examples of MAAC outputs and colored words indicate that keywords appear in both predicted and ground-truth sentences. (ii) The visualization for attention matrices of keywords and previously predicted words in the semantic attention module of example 1.

epochs. Next, we finetune the whole keyword encoder with the learning rate of 5×10^{-4} for 25 epochs. Then, it can be divided into two parts for training the whole MAAC: CE training and RL fine-tuning. CE training takes 30 epochs while the parameters of the pre-trained keyword encoder are frozen. Finally, the 30^{th} CE training model is used for reinforcement learning (RL) fine-tuning 55 epochs. In all training stages, we adopt an Adam optimizer with a mini-batch size of 32, and exponential decay to adjust the learning rate with a factor of 0.98 every epoch. The initial learning rates are set to 3×10^{-4} and 5×10^{-5} for two parts of training the whole MAAC. In the inference stage, we adopt beam search with a beam size of 4 that is implemented to achieve the best decoding performance.

In order to avoid over-fitting and increase data diversity, SpecAugment [20], SpecAugment++ [21], Mixup [22], Label smoothing [23] and teacher forcing [24] are used in the training phase. For Mixup method, it is just used in the training of the keyword encoder. The label smoothing and teacher forcing are just used while training the whole MAAC.

3.2. Result Analysis

We compare our proposed MAAC with the following current models: (1) Baseline [10] is proposed by K. Drossos *et al.*, which employs a GRU-GRU encoder-decoder framework; (2) Temporal attention model (TAM) [5] uses the CNN encoder and the LSTM-based decoder with the temporal attention mechanism; (3) Transformer-based model (TM) [4] adopts a pre-training strategy to improve captioning performance; (4) UNIS's model [11] uses PANNs to initialize the parameters of the encoder and is pre-trained on AudioCaps dataset [25]; (5) SJTU's model [12] utilizes AudioSet to pre-train its encoder in order to enhance the ability of the encoder to recognize audio concepts. Both (5) and (6) adopt RL training to obtain the final models.

Table 1 lists the results of various single models on the Clotho dataset. Our MAAC achieves the highest score on all metrics in the CIDEr-D optimization stage. In addition, the CIDEr-D score of the proposed MAAC improves from 41.9 to 49.1 after further optimizing CIDEr-D.

Through Figure 2 (i), we can find that the pre-trained keyword encoder can almost recognize the main concepts *i.e.* keywords (e.g. *bird* and *chirp* in example 1) of a given audio clip, and the keywords may appear in different states in the ground-truth captions and the predicted sentences. Figure 2 (ii) further shows that keywords and previously predicted words are concerned to generate the current word. For instance, when the decoder is generating "chirping", it pays more attention to the "birds" in the previously predicted words but pays less attention to "birds" in the keywords. That is to say, previously predicted words and keywords are complementary to each other in the semantic attention module.

3.3. Ablative Analysis

To quantify the impact of the proposed multi-modal attention module, we compare our MAAC against a set of other ablated models with different settings. The results of various models are shown in Table 2. We firstly design the "base" model which does not use previously predicted words and keywords (*i.e.* the semantic attention module). Then we add the information of previously predicted words or keywords to the "base" model. We find that it has little impact on the performance of the model by only introducing previously predicted words. It might be that previously predicted words would contain wrong words that destroys the input information of the decoder. In addition, the model which only uses the keywords in the semantic attention module could not converge. From section 3.2, we know that keywords contain the main concepts of an audio clip. When we only utilize them in the semantic attention module, they will cause the decoder to pay more attention to the part of the keywords and ignore the overall semantic relationship. Moreover, we examine the performance of using a shared (or not) semantic attention module on its performance and find that a sharing semantic attention module could further improve the CIDEr-D score.

4. CONCLUSION

In this paper, we propose a novel audio captioning model based on the multi-modal attention module which utilizes both acoustic and semantic information to generate captions. In addition, the performance of the MAAC achieves a new state-of-the-art under the two stages of training. The ablation experiments further demonstrate the effectiveness of the multi-modal attention module. In future work, we would concentrate on how to align the multi-modal information more effectively to improve the performance of the AAC.

5. REFERENCES

[1] A.-M. Oncescu, A. Koepke, J. F. Henriques, Z. Akata, and S. Albanie, "Audio retrieval with natural language queries," *arXiv preprint arXiv:2105.02192*, 2021.

[2] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2048–2057.

[3] M. Wu, H. Dinkel, and K. Yu, "Audio caption: Listen and tell," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 830–834.

[4] Y. Wu, K. Chen, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on transformer and pre-training for 2020 DCASE audio captioning challenge," DCASE2020 Challenge, Tech. Rep., 2020.

[5] H. Wang, B. Yang, Y. Zou, and D. Chong, "Automated audio captioning with temporal attention," DCASE2020 Challenge, Tech. Rep., 2020.

[6] X. Xu, H. Dinkel, M. Wu, Z. Xie, and K. Yu, "Investigating local and global information for automated audio captioning with transfer learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 905–909.

[7] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A transformer-based audio captioning model with keyword estimation," *arXiv preprint arXiv:2007.00222*, 2020.

[8] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[9] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[10] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 736–740.

[11] X. Mei, Q. Huang, X. Liu, G. Chen, J. Wu, Y. Wu, J. Zhao, S. Li, T. Ko, H. L. Tang, X. Shao, M. D. Plumbley, and W. Wang, "An encoder-decoder based audio captioning system with transfer and reinforcement learning for DCASE challenge 2021 task 6," DCASE2021 Challenge, Tech. Rep., July 2021.

[12] X. Xu, Z. Xie, M. Wu, and K. Yu, "The SJTU system for DCASE2021 challenge task 6: Audio captioning based on encoder pre-training and reinforcement learning," DCASE2021 Challenge, Tech. Rep., July 2021.

[13] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1243–1252.

[14] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[15] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004, pp. 74–81.

[16] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005, pp. 65–72.

[17] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4566–4575.

[18] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *European Conference on Computer Vision*. Springer, 2016, pp. 382–398.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[21] H. Wang, Y. Zou, and W. Wang, "Specaugment++: A hidden space data augmentation method for acoustic scene classification," *arXiv preprint arXiv:2103.16858*, 2021.

[22] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[24] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015.

[25] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 119–132.

AUDIO-VISUAL SCENE CLASSIFICATION: ANALYSIS OF DCASE 2021 CHALLENGE SUBMISSIONS

Shanshan Wang, Toni Heittola, Annamaria Mesaros, Tuomas Virtanen

Computing Sciences
Tampere University, Finland
{shanshan.wang, toni.heittola, annamaria.mesaros, tuomas.virtanen}@tuni.fi

ABSTRACT

This paper presents the details of the Audio-Visual Scene Classification task in the DCASE 2021 Challenge (Task 1 Subtask B). The task is concerned with classification using audio and video modalities, using a dataset of synchronized recordings. This task has attracted 43 submissions from 13 different teams around the world. Among all submissions, more than half of the submitted systems have better performance than the baseline. The common techniques among the top systems are the usage of large pretrained models such as ResNet or EfficientNet which are trained for the task-specific problem. Fine-tuning, transfer learning, and data augmentation techniques are also employed to boost the performance. More importantly, multi-modal methods using both audio and video are employed by all the top 5 teams. The best system among all achieved a logloss of 0.195 and accuracy of 93.8%, compared to the baseline system with logloss of 0.662 and accuracy of 77.1%.

Index Terms— Audio-visual scene classification, DCASE Challenge 2021

1. INTRODUCTION

Acoustic scene classification (ASC) has been an important task in the DCASE Challenge throughout the years, attracting the largest number of participants in each edition. Each challenge included a supervised, closed set classification setup, with increasingly large training datasets [1], [2], [3], which has allowed the development of a wide variety of methods. In recent years, the task has focused on robustness to different devices and low-complexity solutions [4].

Scene classification is commonly studied in both audio and video domains. For acoustic scene classification the input is typically a short audio recording, while for visual scene classification tasks the input can be an image or a short video clip. State-of-the-art solutions for ASC are based on spectral features, most commonly the log-mel spectrogram, and convolutional neural network architectures, often used in large ensembles [3]. In comparison, visual scene classification (VSC) from images has a longer history and more types of approaches, e.g. global attribute descriptors, learning spatial layout patterns, discriminative region detection, and more recently hybrid deep models [5]. The classification performance for images has been significantly increased when large-scale image datasets like ImageNet [6] became available. Various network structures have been explored over these years, e.g. CNN [7], while more recently, ResNet [8] and EfficientNet [9] have been proposed to further increase the performance.

Motivated by the fact that we humans perceive the world through multiple senses (seeing and hearing), and in each individual domain methods have reached maturity, multimodal analysis has become a pursued research direction for further improvement. Recent work has shown the joint learning of acoustic features and visual features could bring additional benefits in various tasks, allowing novel target applications such as visualization of the sources of sound in videos [10], audio-visual alignment for lip-reading [11], or audio-visual source separation [12]. Feature learning from audio-visual correspondence (AVC) [13], and more recent work that learns features through audio-visual spatial alignment from 360 video and spatial audio [14], have shown significant improvement in performance in various downstream tasks.

Audio-visual scene classification (AVSC) is introduced in DCASE 2021 Challenge for the first time, even though research on audio-visual joint analysis has been active already for many years. The novelty of the DCASE task is use of a carefully curated dataset of audio-visual scenes [15], in contrast to the use of audio-visual material from YouTube as in the other studies. Audio-visual data collected from the Youtube mostly has automatically generated label categories, which makes the data quality irregular. Besides, most of the datasets based on material from Youtube are task specific, e.g., action recognition [16], sport types [17], or emotion [18]. In [15], the dataset is carefully planned and recorded using the same equipment, which gives it a consistent quality.

In this paper we introduce the audio-visual scene classification task setup of DCASE 2021 Challenge. We shortly present the dataset used in the task and the given baseline system. We then present the challenge participation statistics and analyze the submitted systems in terms of approaches. Since visual data has a large number of large datasets, e.g. ImageNet [6], CIFAR [19], most methods employing the visual modality are expected to use pretrained models or transfer learning based on the pretrained models. A number of resources have been listed and allowed as external data in the DCASE 2021 website.

The paper is organized as follows: Section 2 introduces the dataset, system setup and baseline system results. Section 3 presents the challenge results and Section 4 gives an analysis of selected submissions. Finally, Section 5 concludes this paper.

2. AUDIO-VISUAL SCENE CLASSIFICATION SETUP

In DCASE 2021 Challenge, an audio-visual scene classification task, illustrated in Fig.1, is introduced for the first time. The input to the system is both acoustic and visual signals. Single- and multi-

This work was supported in part by the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND.

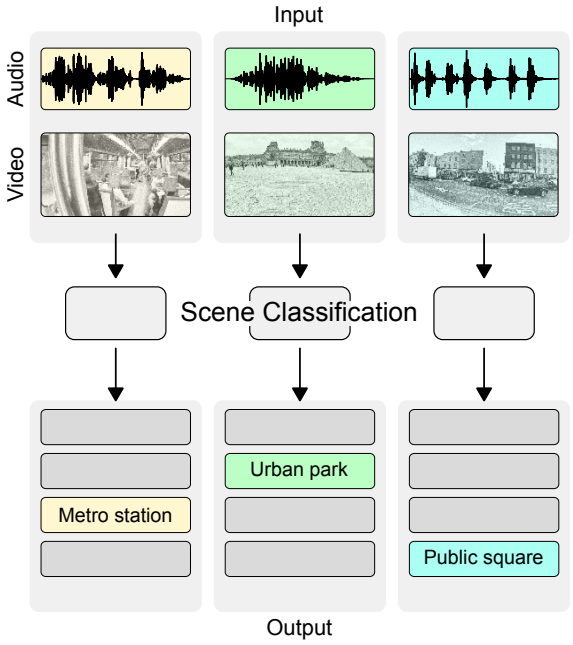


Figure 1: Overview of audio-visual scene classification

modal approaches were allowed, therefore there was no requirement to use both audio and video data.

2.1. Dataset

The dataset for this task is **TAU Urban Audio-Visual Scenes 2021**. The dataset was recorded during 2018-2019 and consists of ten scene classes: airport, shopping mall (indoor), metro station (underground), pedestrian street, public square, street (traffic), traveling by tram, bus and metro (underground), and urban park, from 12 European cities: Amsterdam, Barcelona, Helsinki, Lisbon, London, Lyon, Madrid, Milan, Prague, Paris, Stockholm, and Vienna. The audio content of the dataset is a subset of TAU Urban Acoustic Scenes 2020, in which data was recorded simultaneously with four different devices [2].

The video content of the dataset was recorded using a GoPro Hero5 Session; the corresponding time-synchronized audio data was recorded using a Soundman OKM II Klassik/studio A3 electret binaural in-ear microphone and a Zoom F8 audio recorder with 48 kHz sampling rate and 24 bit resolution. The camera was mounted at chest level on the strap of the backpack, therefore the captured audio and video have a consistent relationship between moving objects and sound sources. Faces and licence plates in the video were blurred during the data postprocessing stage, to meet the requirements of the General Data Protection Regulation law by the European Union.

The development dataset contains 34 hours of data, provided in files with a length of 10 seconds. Complete statistics of the dataset content can be found in [15]. The evaluation set contains 20 hours of data from 12 cities (2 cities unseen in the development set), in files with a length of 1 second.

2.2. Performance Evaluation

Evaluation of systems is performed using two metrics: multiclass cross-entropy (log-loss) and accuracy. Ranking of submissions is performed using the log-loss. Accuracy is provided for comparison with the ASC evaluations from the challenge previous editions.

2.3. Baseline system and results

The baseline system is based on OpenL3 [13] and uses both audio and video branches in the decision. The audio and video embeddings are extracted according to the original OpenL3 architecture, then each branch is trained separately for scene classification based on a single modality. The trained audio and video sub-networks are then connected using two fully-connected feed-forward layers of size 128 and 10.

Audio embeddings are calculated with a window length of 1 s and a hop length of 0.1 s, 256 mel filters, using the ”environment” content type, resulting in an audio embedding vector of length 512. Video embeddings are extracted using the same variables as the audio embedding, excluding the hop length, resulting in a video embedding vector of length 512. Embeddings are further preprocessed using z-score normalization for bringing them to zero mean and unit variance. For training the joint network, Adam optimizer [20] is used with a learning rate set to 0.0001 and weight decay of 0.0001. Cross-entropy loss is used as the loss function. The models with best validation loss are retained. More details on the system are presented in [15].

The baseline system results are presented in Table 1. In the test stage, the system predicts an output for each 1 s segment of the data. The results from Table 1 are different than the ones presented in [15], because in the latter the evaluation is done for the 10 s clip. In that case, the final decision for a clip is based on the maximum probability over 10 classes after summing up the probabilities that the system outputs for the 1 s segments belonging to the same audio or video clip. In DCASE challenge, the evaluation data contains clips with a length of 1 s, therefore the baseline system is evaluated on segments of length 1 s also in development.

The results in Table 1 show that the easiest to recognize was the ”street with traffic” class, having the lowest log-loss of all classes at 0.296, and an accuracy of 89.6%. At the other extreme is the ”airport” class, with a log-loss of 0.963 and accuracy 66.8%, and an average performance of 0.658 log-loss, with 77.0% accuracy. The class-wise log loss is calculated taking into account only the test items belonging to the considered class (splitting the classification task into ten different sub-problems), while overall log loss is calculated taking into account all test items.

3. CHALLENGE RESULTS

There are altogether 13 teams that participated to this task with one to four submission entries from each team, summing up to 43 entries. Of these, systems of 8 teams outperformed the baseline system. The top system, Zhang_IOA_3 [21], achieved a log loss of 0.195 and accuracy of 93.8%. Among all submissions, 15 systems achieved an accuracy higher than 90% and a log loss under 0.34. There are 11 systems which use only the audio modality, three that use only video, and 27 multimodal systems. The best performing audio-only system, Naranjo-Alcazar_UV_3 [22], is ranked 32nd with 1.006 logloss and 66.8% accuracy. The best performing video-only system, Okazaki_LDSLVision_1 [23], is ranked 12th

	Baseline (audio-visual)	
Scene class	Log loss	Accuracy
Airport	0.963	66.8%
Bus	0.396	85.9%
Metro	0.541	80.4%
Metro station	0.565	80.8%
Park	0.710	77.2%
Public square	0.732	71.1%
Shopping mall	0.839	72.6%
Street pedestrian	0.877	72.7%
Street traffic	0.296	89.6%
Tram	0.659	73.1%
Overall	**0.658**	77.0%

Table 1: Baseline system performance on the development dataset

with 0.312 log loss and 91.6% accuracy, while the top 8 systems belong to 2 teams, and are all multimodal.

4. ANALYSIS OF SUBMISSIONS

A general analysis of the submitted systems shows that the most popular approach is usage of both modalities, with multimodal approaches being used by 26 of the 43 systems. Log-mel energies are the most widely used acoustic features among the submissions. Data augmentation techniques, including mixup, SpecAugment, color jitter, and frequency masking, are applied in almost every submitted system. The usage of large pretrained models such as ResNet, VGG, EfficientNet trained on ImageNet or Places365 and fine-tuned on the challenge dataset is employed in most systems to extract the video embeddings. The combination of information from the audio and video modalities is implemented as both early and late fusion. The main characteristics and performance on the evaluation set of the systems submitted by the top 5 teams are presented in Table 2.

4.1. Characteristics of top systems

The top ranked system [21] adopts multimodality to solve the task. In the audio branch, authors employed 1D deep convolutional neural network and investigated three different acoustic features: mel filter bank, scalogram extracted by wavelets, and bark filter bank, calculated from the average and difference channels, instead of left and right channels. In the video branch, authors studied four different pretrained models: ResNet-50, EfficientNet-b5, EfficientNetV2-small, and swin transformer. Authors use the pretrained model trained on ImageNet, and fine-tune it first on Places365, then on TAU Urban Audio-Visual Scenes 2021 dataset. RandomResizedCrop, RandomHorizontalFlip, and Mixup data augmentation techniques are also applied. This approach takes the top 4 ranks, with the best system being based on the combination of EfficientNet-b5 and log-mel acoustic features, a hybrid fusion comprised of model-level and decision-level fusion.

The team ranked on second place [24] used an audio-visual system and explored various pretrained models for both audio and video domain. The systems also include data augmentation through SpecAugment, channel confusion, and pitch shifting. Specifically, for the audio embedding, authors investugated use of the pretrained VGGish and PANN networks, both trained on AudioSet, with transfer learning applied to solve the AVSC task. Authors propose use of FCNN and ResNet to extract high-level audio features, to better leverage the acoustic presentations in these models. For the video embeddings, authors adopt the pretrained model trained on ImageNet and Places365. Authors also propose to use embeddings extracted from an audio-visual segment model (AVSM), to represent a scene as a temporal sequence of fundamental units by using acoustic and visual features simultaneously. The AVSM sequence is translated into embedding through a text categorization method, and authors call this a text embedding. The combination of audio, video, and text embeddings significantly improves their system's performance compared to audio-video only.

The team ranked third [23] also used audio, video and text for solving the given task. Authors use log-mel spectrogram, frame-wise image features, and text-guided frame-wise features. For audio, the popular pretrained CNN model trained on AudioSet is used, with log-mel spectrogram; for video, authors select three backbones ResNeSt, RegNet, and HRNet; finally, for the text modality, authors use CLIP image encoders trained on image and text caption pairs using contrastive learning, to obtain text-guided frame-wise image features. The three domain-specific models were ensembled using the class-wise confidences of the separate outputs, and post-processed using the confidence replacement approach of thresholding the log-loss. In this way, the system can avoid the large log-loss value corresponding to a very small confidence. Authors show that this approach has significantly improved the log-loss results.

4.2. Systems combinations

Confidence intervals for the accuracy of the top systems presented in Table 2 are mostly not overlapping (small overlap between ranks 6 and 9). Logloss confidence intervals are ± 0.02 for all systems. Because the systems are significantly different, we investigate some systems combinations. We first calculate the performance when combining the outputs of the top three systems with a majority vote rule. The obtained accuracy is 94.9%, with a 95%CI of ± 0.2. Even though modest, this increase is statistically significant, showing that the systems behave differently for some of the test examples.

Looking at the same systems as a best case scenario, we calculate accuracy by considering a correct item if *at least one* of the systems has classified it correctly. In this case, we obtain an accuracy of 97.5% with a 95% CI of ± 0.1, showing that the vast majority of the test clips are correctly classified by at least one of the three considered systems, and that if the right rules for fusion can be found, performance can be brought very close to 100%.

4.3. General trends

An analysis over the individual modalities among the submissions reveals that video-based methods have advantages over audio-based ones. The best audio-only model achieves a logloss of 1.006 and accuracy of 66.8%, while the best video-only model has a much lower logloss of 0.132 and much higher accuracy of 91.6%. This is due to several reasons. Firstly, image classification domain has a relatively longer history than the audio scene classification, which allowed the development of mature and large pretrained models with millions of parameters, for example, CNNs, ResNet, VGG, EfficientNet and so forth. Secondly, the large-scale image datasets and the variety of the available image data, such as ImageNet, COCO and so forth, help making the model more robust. Thirdly, image domain has attracted much attention throughout the years, including

Rank	Team	Logloss	Accuracy (95% CI)	Fusion Methods	Model Complexity
1	Zhang_IOA_3	0.195	93.8% (93.6 - 93.9)	early fusion	110M
5	Du_USTC_4	0.221	93.2% (93.0 - 93.4)	early fusion	373M
9	Okazaki_LDSLVision_4	0.257	93.5% (93.3 - 93.7)	audio-visual	636M
10	Yang_THU_3	0.279	92.1% (91.9 - 92.3)	early fusion	121M
16	Hou_UGent_4	0.416	85.6% (85.3 - 85.8)	late fusion	28M
24	DCASE2021 baseline	0.662	77.1% (76.8 - 77.5)	early fusion	711k

Table 2: Performance and general characteristics of top 5 teams (best system of each team). All these systems use both audio and video.

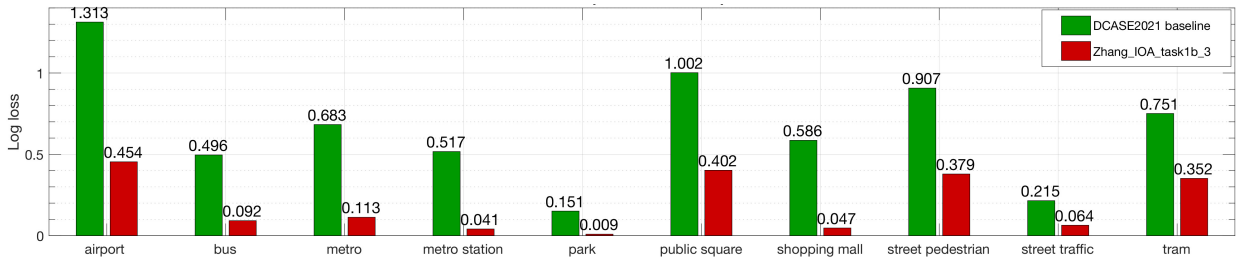


Figure 2: Class-wise performance comparison between the top 1 system and the baseline system on the evaluation set.

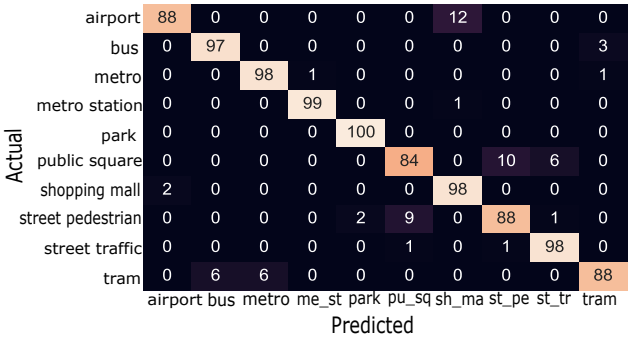


Figure 3: Confusion matrix of the top-performing system [21].

large numbers of participants in various challenges, such as Kaggle challenges, therefore promoting the rapid development of this field.

Even though the audio-only models achieve lower performance than video-only ones, the best performance was obtained by systems which combined the two modalities. This validates our initial idea that joint modeling of audio and visual modalities can bring significant performance gain compared to state-of-the-art uni-modal systems.

An analysis of the machine learning characteristics of the submitted systems reveals that there is a direct relationship between the performance and the model complexity, that is, in general, the top-performing systems tend to have more complex models with larger numbers of trainable parameters. Indeed, Spearman's rank correlation coefficient [25] between the model complexity and system rank is 0.75, indicating that they are highly correlated. Considering both complexity and the performance, the baseline system is a balanced choice with a satisfactory performance.

The choice of evaluation metrics does not affect the ranking drastically. We found that the top team stays the same position, the system Okazaki_LDSLVision_4 would jump to the second instead of the third spot, Hou_UGent_4 would drop to the seventh instead of the fifth, and Wang_BIT_1 would jump to the tenth from the thirteenth. The Spearman's rank correlation between accuracy and logloss indicates a very strong correlation, at 0.93.

In general, no significant changes have been found in terms of the system performance between the development dataset and the evaluation dataset, which shows that the dataset is well balanced and the systems have consistent behavior and good generalization properties. Most of the system performance shows only a very slight drop in performance on the evaluation dataset, which is explained by the data from two cities unseen in training.

The confusion matrix of the top system is shown in Fig.3. In general, the top system performs very good in all classes; the lowest class performance is 84%, and the highest is 100%. In particular, the system achieves the best performance in "park"(100) and excellent in "metro station"(99), "metro"(98), "shopping mall"(98), "street traffic"(98), "bus"(97). We observe that "airport" class is mostly misclassified as "shopping mall"(12); "public square" is often misclassified as "street pedestrian"(10) and "street traffic"(6); and "tram" is mostly misclassified as "bus" and "metro". This behavior is rather intuitive, since inside the airport there are many shops which may resemble "shopping mall", and inside the "tram" there are mostly seats and people which may also resemble "bus" or "metro".

A bar plot comparison of the class-wise performance on the evaluation set between the baseline and the top system is shown in Fig.2. It can be seen that the top system has significantly higher performance in all classes, especially "airport" (logloss 0.859 smaller) and "public square" (logloss 0.6 smaller). Some similarities between the baseline system and the top system can be observed in the bar plot, with "park", "street traffic" being the easiest to classify among all classes for both systems, and "airport", "public square" being the most difficult ones.

5. CONCLUSIONS AND FUTURE WORK

Audio-visual scene classification task was introduced in the DCASE2021 challenge for the first time, and had a high number participants and submissions. More than half of the submissions

outperformed the baseline system. Multimodal approaches were widely applied among the submissions, and also achieved the best performance compared to uni-modal methods. The choice of models used by the top systems reveals that large and well-trained pre-trained models are important for this task, while data augmentation and fine-tuning techniques help making the system more robust.

6. REFERENCES

[1] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, Hungary, 2016.

[2] ——, "A multi-device dataset for urban acoustic scene classification," in *Proc. of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 9–13.

[3] ——, "Acoustic scene classification in DCASE 2019 challenge: Closed and open set classification and data mismatch setups," in *Proc. of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, October 2019, pp. 164–168.

[4] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions," *arXiv preprint arXiv:2005.14623*, 2020.

[5] L. Xie, F. Lee, L. Liu, K. Kotani, and Q. Chen, "Scene recognition: A comprehensive survey," *Pattern Recognition*, vol. 102, p. 107205, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S003132032030011X

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[10] R. Arandjelovic and A. Zisserman, "Objects that sound," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[11] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Lip reading sentences in the wild," in *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3444–3453.

[12] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba, "The sound of pixels," in *Proc. of the European Conf. on Computer Vision (ECCV)*, September 2018.

[13] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 609–617.

[14] P. Morgado, Y. Li, and N. Vasconcelos, "Learning representations from audio-visual spatial alignment," *arXiv preprint arXiv:2011.01819*, 2020.

[15] S. Wang, A. Mesaros, T. Heittola, and T. Virtanen, "A curated dataset of urban scenes for audio-visual scene analysis," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021. [Online]. Available: https://arxiv.org/abs/2011.00030

[16] K. Soomro, A. Zamir, and M. Shah, "UCF0101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[17] R. Gade, M. Abou-Zleikha, M. Graesboll Christensen, and T. B. Moeslund, "Audio-visual classification of sports types," in *Proc. of the IEEE Int.Conf. on Computer Vision (ICCV) Workshops*, December 2015.

[18] S. Zhang, S. Zhang, T. Huang, W. Gao, and Q. Tian, "Learning affective features with a hybrid deep model for audio–visual emotion recognition," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 3030–3043, 2018.

[19] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[20] J. Kingma, D.and Ba, "Adam: A method for stochastic optimization," in *Proc. Int.Conf. on Learning Representations*, 2014.

[21] M. Wang, C. Chen, Y. Xie, H. Chen, Y. Liu, and P. Zhang, "Audio-visual scene classification using transfer learning and hybrid fusion strategy," DCASE2021 Challenge, Tech. Rep., June 2021.

[22] J. Naranjo-Alcazar, S. Perez-Castanos, M. Cobos, F. J. Ferri, and P. Zuccarello, "Task 1B DCASE 2021: Audio-visual scene classification with squeeze-excitation convolutional recurrent neural networks," DCASE2021 Challenge, Tech. Rep., June 2021.

[23] S. Okazaki, K. Quan, and T. Yoshinaga, "Ldslvision submissions to dcase'21: A multi-modal fusion approach for audio-visual scene classification enhanced by clip variants," DCASE2021 Challenge, Tech. Rep., June 2021.

[24] Q. Wang, S. Zheng, Y. Li, Y. Wang, Y. Wu, H. Hu, C.-H. H. Yang, S. M. Siniscalchi, Y. Wang, J. Du, and C.-H. Lee, "A model ensemble approach for audio-visual scene classification," DCASE2021 Challenge, Tech. Rep., June 2021.

[25] C. Spearman, "The proof and measurement of association between two things," *The American journal of psychology*, vol. 15, no. 1, pp. 72–101, 1904.

TOWARD INTERPRETABLE POLYPHONIC SOUND EVENT DETECTION WITH ATTENTION MAPS BASED ON LOCAL PROTOTYPES

Pablo Zinemanas[1], *Martín Rocamora*[2], *Eduardo Fonseca*[1], *Frederic Font*[1] *and Xavier Serra*[1]

[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain, name.surname@upf.edu
[2] Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, rocamora@fing.edu.uy

ABSTRACT

Understanding the reasons behind the predictions of deep neural networks is a pressing concern as it can be critical in several application scenarios. In this work, we present a novel interpretable model for polyphonic sound event detection. It tackles one of the limitations of our previous work, i.e. the difficulty to deal with a multi-label setting properly. The proposed architecture incorporates a prototype layer and an attention mechanism. The network learns a set of local prototypes in the latent space representing a patch in the input representation. Besides, it learns attention maps for positioning the local prototypes and reconstructing the latent space. Then, the predictions are solely based on the attention maps. Thus, the explanations provided are the attention maps and the corresponding local prototypes. Moreover, one can reconstruct the prototypes to the audio domain for inspection. The obtained results in urban sound event detection are comparable to that of two opaque baselines but with fewer parameters while offering interpretability.

Index Terms— interpretability, sound event detection, prototypes

1. INTRODUCTION

After significant advances in computer vision, speech recognition, and natural language processing, deep learning models have also become the standard approach in environmental sound processing tasks, such as sound event detection, audio tagging, and acoustic scene classification [1, 2]. The increasing complexity of such models makes it difficult to explain the process that leads to its output in a way that humans can understand. This can be problematic in some real–world deployment scenarios. Therefore, research on interpretability and accountability of predictive models are steadily growing. In addition, interpretable models make it easier to debug, detect biases, and design defenses for adversarial attacks [3].

Instead of creating intrinsically interpretable deep neural networks, most existing works follow a *post hoc* approach, i.e., they try to explain the input-output behavior of a *black-box* model. For example, training a linear proxy model that imitates the behavior of the original model but is easier to interpret is a common approach [4]. However, since the proxy model is typically a local linear approximation of a non-linear model, it can fall short of providing a reliable explanation [5]. Other post hoc explanation methods focus on studying the representations of the input data learned by the network or highlighting the input characteristics that strongly influence the output. For instance, saliency maps are a typical example of this approach [6], where the gradient of the output with respect to the input is used to identify the most relevant portions of the input. However, this is incomplete as an explanation, as it provides no clue about how the relevant information is being used [5].

Rather than producing explanations of black-box models, some research seeks to develop inherently interpretable neural networks that provide faithful explanations to what the model actually computes [5]. By adding specific components, one can strive for rendering some form of interpretability while being as accurate as a black-box model. An example of this is the incorporation of attention mechanisms, which are network components that learn to select the part of the input that the rest of the model should focus on. Thus, besides improving predictive performance, the relative importance of the input units offers insights into the model's decision-making process. Yet, whether attention mechanisms can provide faithful explanations is a matter of current debate, as it depends on how they are implemented and the degree of interpretability pursued [7].

Learning through prototypes is another approach that can provide inherent interpretability to deep neural networks. Decision are based on a few relevant examples known as *prototypes* that serve as a distillation of the data and have a high interpretable value [8, 9]. A prototype is a vector that is close or identical to an instance of the training set. Deep neural networks can learn those prototypes in a flexible latent space. For example, the interpretable network proposed by Li et al. [10] for image classification is based on prototypes. The architecture appends a special prototype layer and uses an autoencoder. The prototypes are learned in the low-dimensional latent space produced by the encoder, and they can be reconstructed by applying the decoder. The predictions are based on the distance from the data instance to each prototype in the latent space. Thus, the explanations are the prototypes and the distances to them, which are the actual computations of the model to generate the output.

Our previous work extended this approach to audio classification [11]. There, we proposed the Audio Prototype Network (APNet) and showed compelling results when applied to speech, music, and environmental audio, for problems with a single class label per audio clip. However, in a polyphonic setting (i.e multi-label), an input instance corresponding to several classes should be simultaneously close to prototypes of those classes in the latent space. Unfortunately, learning such latent space proved challenging in practice, thus motivating the alternative approach proposed herein.

In this work, we propose a novel interpretable deep neural network for polyphonic sound event detection. To provide interpretability, we leverage the prototypes network approach and attention mechanisms. The network learns *local prototypes*, i.e. data points in the latent space representing a patch in the input representation. The approach is similar to that of [12] for single class image classification, which compares image parts to learned prototypes. However, we extend the scope to a multi-label setting with promising results in sound event detection. Besides, the proposed model learns attention maps used for positioning the local prototypes and reconstructing the latent space properly. Then, the detec-

tion is solely based on the attention maps. Thus, the explanations of the network are in the form of local prototypes and attention maps.

2. RELATED WORK

Some post hoc visualization methods have been applied in the audio domain. For instance, in [13] a convolutional–recurrent network trained for polyphonic sound event detection was evaluated using saliency maps. In [14], a gradient–based approach was proposed for visualizing heat maps in the first layer of an end–to–end convolutional neural network. Regarding proxy models, SLIME [15] is a variation of the LIME [4] algorithm for audio content analysis, which produces visual explanations in the form of temporal, frequency, and time-frequency segmentation. The model we propose herein also generates visual explanations of its predictions, but these are faithful to its computations instead of post hoc explanations.

Prototypical learning has been applied to audio problems but not necessarily looking for interpretability. For example, Pons et al. [16]—following [17]—used prototypical networks for audio classification with few data. However, their system is not intended to be interpretable, so one can not reconstruct the prototypes to the input space. In contrast, APNet and the model we propose herein allow for reconstructing the prototypes to the input space through the decoder and then mapping them to the audio domain.

The first models that used attention mechanisms in the audio domain applied them in conjunction with recurrent networks for speech recognition [18, 19]. Nowadays, attention mechanisms are widely used for speech, music, and other audio-related problems because of their ability to capture long-term temporal information. Self-attention mechanisms are used instead of recurrent layers to integrate temporal information; for instance, they were applied for music generation [20] and tagging [21]. Attention mechanisms can also be used for weighting the frequency dimension to create interpretable adaptive filter banks [22]. In contrast, our model does not use attention maps to weight input's features. Instead, we use them as the only information to classify sound sources. Furthermore, since we devise the attention maps for proper reconstruction from the local prototypes, they are interpretable by design.

Some other models combine attention mechanisms and prototypes. For instance, ProtoAttend [23] selects input-dependent prototypes based on a relational attention mechanism that connects the encoded representation and the prototype candidates. In this case, the prototypes are instances from the training data. However, other methods use mean vectors as prototypes for few-shot learning [24].

3. PROPOSED MODEL

Let $\boldsymbol{X}_i \in \mathbb{R}^{\mathcal{T}\times\mathcal{F}}$ be the i-th mel-spectrogram where $\mathcal{T}$ and $\mathcal{F}$ are the number of time frames and frequency bins, respectively. Therefore we define the training set as $\{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^{N}$, where $\boldsymbol{Y}_i \in \mathbb{R}^K$ are the one-hot encoded labels, N is the number of instances and K is the number of classes. APNet is formed by two main components: an autoencoder and a classifier [11]. Our proposed model uses the same autoencoder from APNet, which is represented in the upper branch of Figure 1, and utilizes a novel classifier. The encoder is aimed at extracting meaningful features from the input: $\boldsymbol{Z}_i = f(\boldsymbol{X}_i)$, where $\boldsymbol{Z}_i$ is a tensor of shape (T, F, C) and represents the transformed input in the latent space. C is the number of channels of the encoder's last convolutional layer. The decoder part of the autoencoder is used for reconstructing the mel-spectrogram: $\widetilde{\boldsymbol{X}}_i = g(\boldsymbol{Z}_i) \in \mathbb{R}^{\mathcal{T}\times\mathcal{F}}$. Both the encoder and the decoder are

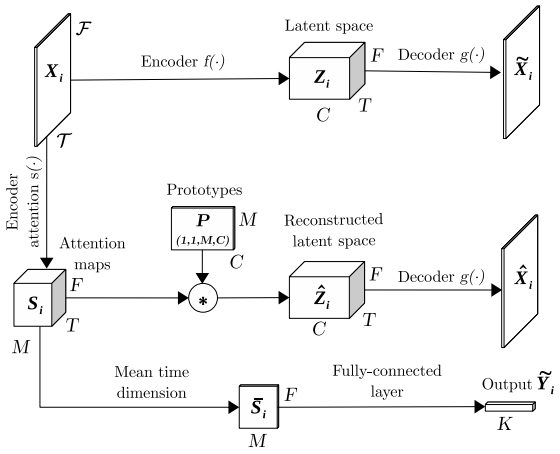


Figure 1: Diagram of the proposed model.

formed by three convolutional layers with leaky ReLu activations. The encoder includes two max-pooling layers interspersed between the convolutions and the decoder applies the corresponding unpooling layers. Please refer to [11] for more details. The classifier of APNet is based on the distance from $\boldsymbol{Z}_i$ to a set of M prototypes with the same shape (T, F, C). Therefore, a prototype is a point in the latent space corresponding to the full mel-spectrogram representation in the input space. This makes it troublesome for APNet to represent a multi-label input instance as it should be close to prototypes from different classes.

The model proposed in this work is devised to overcome this limitation, i.e. it is capable of detecting various simultaneous sound events. The middle and bottom branches in the diagram of Figure 1 show this novel classifier. We use another encoder, $s(\cdot)$, to extract M attention maps in the latent space: $\boldsymbol{S}_i = s(\boldsymbol{X}_i)$, where $\boldsymbol{S}_i$ is a tensor of shape (T, F, M). The encoder $s(\cdot)$ is similar to the autoencoder's one, $f(\cdot)$, but with ReLu activations to force a non-negative output. Each attention map is related to one prototype. Therefore the network learns a set of M prototypes of shape $(1, 1, C)$. We represent the M prototypes as a tensor $\boldsymbol{P}$ of shape $(1, 1, M, C)$. Note that each prototype represents one point in the time-frequency plane in the latent space. Therefore, in the input space these prototypes represent a patch of shape equal to the receptive field of the encoder network (32×32 in this work).

Using the attention maps and the learnable prototypes the model tries to reconstruct the latent representation $\boldsymbol{Z}_i$. This is done by multiplying each attention map by its corresponding prototype and then summing all maps. Note that this is equivalent to a 1×1 2D convolutional layer: $\widehat{\boldsymbol{Z}}_i = \boldsymbol{S}_i * \boldsymbol{P}$, or a dense layer $\widehat{\boldsymbol{Z}}_i = \boldsymbol{S}_i \cdot \boldsymbol{P}_s$, where $\boldsymbol{P}_s$ is the squeezed version of the tensor $\boldsymbol{P}$ with shape (M, C).

Therefore $\widehat{\boldsymbol{Z}}_i$ has the same shape of $\boldsymbol{Z}_i$ and aims to be a reconstruction of the latent space. In summary the attention maps represent the specific weight of each local prototype in each time-frequency point in order to have a good reconstruction of the latent space. Using the decoder $g(\cdot)$ from the top branch, we can project this reconstructed tensor into the input space, $\widehat{\boldsymbol{X}}_i = g(\widehat{\boldsymbol{Z}}_i) \in \mathbb{R}^{\mathcal{T}\times\mathcal{F}}$. In this way, we can visualize the reconstruction of the latent space in the input space to inspect it.

Finally, the bottom branch deals with the detection task, which

is solely based on the attention maps. First, we average the time dimension of $\boldsymbol{S}_i$:

$$\bar{S}_i[f,m] = \frac{1}{T}\sum_{t=1}^{T} S_i[t,f,m] \tag{1}$$

where $\bar{\boldsymbol{S}}_i$ has shape (F, M) and integrates the attention map for each prototype and frequency bin in the latent space. Then, a dense layer connects a flattened version of $\bar{\boldsymbol{S}}_i$ with the classification output:

$$\widetilde{\boldsymbol{Y}}_i = \text{sigmoid}(\bar{\boldsymbol{S}}_i \cdot \boldsymbol{W}), \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{MF\times K}$ is the kernel of the layer and $\widetilde{\boldsymbol{Y}}_i = \{\widetilde{Y}_{ik}\} \in \mathbb{R}^{1\times K}$. We do not use bias in order to keep this layer more interpretable. We seek to audit how the model connects each prototype and each frequency bin to the corresponding output.

3.1. Objective function

We want the model to be able to detect sound events while maintaining the interpretability of the parameters and the explainability of the predictions. For this purpose, we define three losses to train the model. First we have a loss for learning the detection task. Since this is a multi-label problem, we use binary cross-entropy, $\mathcal{L}_c$. Then we define a mean squared error loss to have good reconstruction quality in the autoencoder of the top branch: $\mathcal{L}_r = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{X}_i - \widetilde{\boldsymbol{X}}_i\right\|_2^2$. This loss ensures that we can transform the data from the latent space back to the input space, in particular the learned prototypes.

Finally, we define a loss for enforcing a correct process of reconstruction using the attention maps and the prototypes. In other words, we let the network learn how to position the prototypes using the attention maps. In this sense, we define a mean squared error loss in both latent and input spaces:

$$\mathcal{L}_p = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{Z}_i - \widehat{\boldsymbol{Z}}_i\right\|_2^2 + \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{X}_i - \widehat{\boldsymbol{X}}_i\right\|_2^2. \tag{3}$$

This loss ensures two assets of the model related to its interpretability. First, this loss establishes that the attention maps are learned to be an explicit explanation of how the model makes its predictions. Note that the attention maps are the only information used for the final prediction. And these maps are interpretable since they show how to position each prototype in the latent space in order to have a good reconstruction. Moreover this loss ensures that the prototypes are similar to the data and therefore we can transform them to the input space and audit them.

Besides, we use $l1$ regularization to force some sparsity in the attention map: $\mathcal{R}_s = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{Z}_i\|_1$. This is to prevent the network from reconstructing the latent space by mixing many prototypes. We also apply the same type of regularization to the kernel of the dense layer that connects the attention maps and the output: $\mathcal{R}_w = \|\boldsymbol{W}\|_1$. The idea is that the output for a given class is activated with only a few points on the attention map, both in the frequency and prototype dimension. Therefore we keep the explanations as simple as possible.

While training the proposed system, we optimize the weighted sum of all losses and regularization terms defined previously:

$$\mathcal{L} = \alpha\mathcal{L}_c + \beta\mathcal{L}_r + \gamma\mathcal{L}_p + \delta\mathcal{R}_s + \epsilon\mathcal{R}_w \tag{4}$$

where the weights $(\alpha, \beta, \gamma, \delta, \epsilon)$ are real-valued hyperparameters.

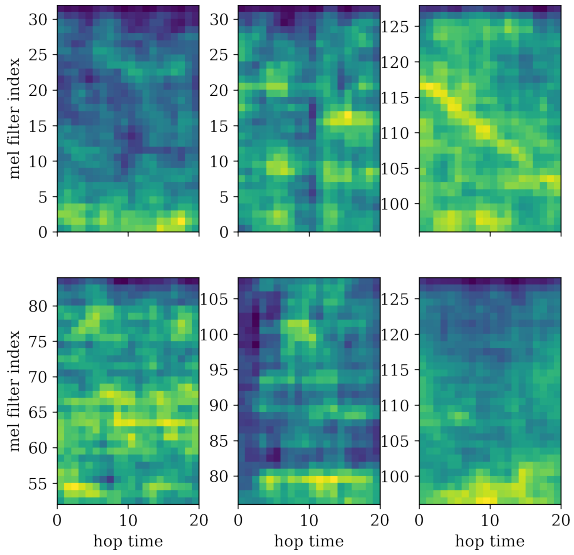


Figure 2: Reconstructed learned local prototypes. The y axis represents the mel bands where the prototypes were reconstructed.

4. EXPERIMENTS AND RESULTS

We train the proposed model by optimizing the objective function defined in Eq. (4). We use Adam optimizer with a learning rate of 0.001 for 50 epochs and we select the model with the top performance in the validation set. We use the following set of hyperparameters $(10, 5, 5, 10^{-5}, 10^{-6})$ and a batch size of 256. The experiments are conducted using the DCASE-models library [25] and the code is available under an open-source license[1].

We compare the performance of the proposed model to that of two different opaque baselines: (1) a convolutional neural network (CNN) formed by three convolution layers and two dense layers [26]; and (2) a multi-layer perceptron (MLP) whose input is the embedding vector extracted from the pre-trained Openl3 model [27]. We optimize a binary cross-entropy loss with the same optimizer and strategy for these baselines as for the proposed model.

We train and evaluate the proposed model and the baselines on the URBAN-SED dataset v2.0 [26]. This is formed by 10-second length audio files corresponding to synthetic mixtures of sound sources obtained from the UrbanSound8k dataset. Each sound event is tagged with one of the following classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. The three models use log-scaled mel-spectrogram as input representation, but with different parameters. Both the CNN and the proposed model uses 128 mel bands and a sampling rate of 22050 Hz. The proposed model uses a window size of 4096 and hop size of 1024 for calculating the spectrograms. On the other hand, CNN uses a window size of 512 and hop size of the same length. Openl3 has predefined parameters [27].

To evaluate the models, we use F-measure ($F1$) and error rate (ER) in a 1-second grid as commonly used for sound event detection [28]. We run the training 10 times and calculate the mean and standard deviation of both metrics. Table 1 shows the performance comparison of the three models along with their number of

[1] https://github.com/pzinemanas/attprotos

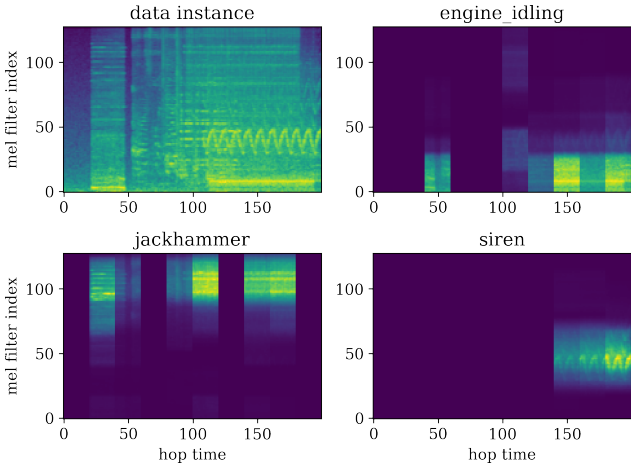


Figure 3: Example of an input instance from the test set masked by the attention maps. At the top left plot we show the mel-spectrogram, which includes sound events of six different classes. The other plots are the same mel-spectrogram but masked by each of the reconstructed attention maps for the corresponding classes.

parameters. Note that the performance of the proposed model is comparable to that of the baselines, but with fewer parameters.

Table 1: Performance comparison of the proposed model and the two baselines. The performance metrics are the F-measure ($F1$) and the error rate (ER). The number (#) of parameters in millions (M) are also included in the comparison.

Network	$\boldsymbol{F1}$ **(%)**	$\boldsymbol{ER}$	**# Params. (M)**
CNN	57.3 ± 0.6	0.568 ± 0.006	0.5
Openl3+MLP	58.2 ± 0.3	0.558 ± 0.004	9.5
Ours	58.8 ± 0.9	0.572 ± 0.007	0.15

4.1. Prototypes

The reconstruction of the latent space helps the network to learn prototypes similar to patches from the training data. We use the decoder part of the autoencoder, $g(\cdot)$, to reconstruct the learned prototypes in the input space. We follow a process similar to that performed in APNet for this purpose [11]. But in this case, we have to extend the prototypes tensor $\boldsymbol{P}$ to have the same shape of the latent space, i.e. (M, T, F, C). To this end, we create a zero tensor of this shape and select a point in the time-frequency plane where to position each prototype. The time is selected arbitrarily at the center, and the frequency is selected by minimizing the distance of each prototype to the data instances. By doing this, we reconstruct the patches in the frequency bands where the closest data instances have these prototypes present. Figure 2 shows a set of selected prototypes. Note that the network learns different types of shapes and textures related to environmental sounds present in the data set.

4.2. Attention maps

For each data instance it is possible to extract the corresponding attention maps to provide and explanation on how the model makes its predictions. For a given class k, we follow the following process:

1. Mask the prediction $\widetilde{\boldsymbol{Y}}_i \in \mathbb{R}^{1\times K}$ by a unit vector of the same shape whose k-th component is the only one equal to 1: $\widetilde{\boldsymbol{Y}}_i^{(k)} = \widetilde{\boldsymbol{Y}}_i \odot \mathbf{1}_k$
2. Get the points of the previous layer that are more connected to the output k by calculating the gradient: $\nabla \bar{\boldsymbol{S}}_i^{(k)} = \widetilde{\boldsymbol{Y}}_i^{(k)} \cdot \boldsymbol{W}^T \in \mathbb{R}^{1\times FM}$
3. Reshape the gradient to (F, M), apply a half-wave rectifier to keep only positive connections and multiply it by the time-averaged attention maps: $\bar{\boldsymbol{S}}_i^{(k)} = \text{ReLu}\left(\nabla \bar{\boldsymbol{S}}_i^{(k)}\right) \odot \bar{\boldsymbol{S}}_i$. This represents the attention maps masked by the most important connections to the output k.
4. Find the most connected prototype by maximizing the energy of the masked attention map: $\widehat{m} = \arg\max_{m\in[1,\dots,M]} \sum_{f=1}^{F} \left(\bar{S}_i^{(k)}[f, m]\right)^2$
5. Extract the frequency-dependent attention function: $\mathcal{S}_i^{(k)}[f] = \bar{S}_i^{(k)}[f, \widehat{m}]$
6. Convert the attention function to the input space. To do this, we first upsample the sequence by a rate of 4 to emulate the two max-pooling operations. Then we apply a moving-average filter to emulate the receptive field. Thus, the length of the filter is equal to the receptive field (32).

Figure 3 shows an example of the attention maps for three different classes. We multiply the attention maps in the input space by the mel-spectrograms, similarly to how the model does in the latent space. Note that the model can detect simultaneous sound events whose energy is concentrated in different frequency bands. Since the attention maps are designed to reconstruct the latent space and are the only information used for classification, these represent the inherent explanation of how the network makes its predictions.

5. CONCLUSION

In this work, we present a novel interpretable model for polyphonic sound event detection. Its predictions are based on attention maps learned for reconstructing the latent space by positioning a set of local prototypes. The network also learns the local prototypes as data points in the latent space representing a patch in the input representation. The attention maps provide a form of explanation that is faithful to the model computations and can give valuable insights into its decision process. Moreover, the prototypes can be reconstructed and thus can be listened to and audited.

The proposed model achieves encouraging results in urban sound event detection for a data set of synthetic mixtures, which are comparable to that from two opaque baselines but with fewer parameters, while at the same time offering interpretability. This is consistent with some previous work that claims that it is often possible to incorporate interpretability into deep learning models to tackle complex tasks without sacrificing performance [10, 12, 5].

Future work includes ablation studies to understand better the impact of the proposed losses and regularization terms in the final model. In addition, more experiments are needed to evaluate the effect of some hyperparameter values, such as the loss weights and the number of prototypes. Besides, we should evaluate the model with datasets recorded in natural conditions. Finally, we seek further development of interpretable models to analyze environmental sounds, including those that learn disentangled representations.

6. REFERENCES

[1] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic Scene Classification in DCASE 2020 Challenge: Generalization Across Devices and Low Complexity Solutions," in *Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, Nov. 2020, pp. 56–60.

[2] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen, "Overview and evaluation of sound event localization and detection in DCASE 2019," *Trans. on Audio, Speech, and Language Processing*, vol. 29, pp. 684–698, 2021.

[3] C. Molnar, *Interpretable Machine Learning*, 2019, https://christophm.github.io/interpretable-ml-book/.

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the predictions of any classifier," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, New York, NY, USA, 2016, p. 1135–1144.

[5] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, p. 206–215, May 2019.

[6] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

[7] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, Nov. 2019, pp. 11–20.

[8] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, vol. 5, no. 4, Dec 2011.

[9] B. Kim, C. Rudin, and J. A. Shah, "The bayesian case model: A generative approach for case-based reasoning and prototype classification," in *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[10] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, vol. 32, 2018.

[11] P. Zinemanas, M. Rocamora, M. Miron, F. Font, and X. Serra, "An interpretable deep learning model for automatic sound classification," *Electronics*, vol. 10, no. 7, 2021.

[12] C. Chen, O. Li, D. Tao, A. J. Barnett, J. Su, and C. Rudin, "This looks like that: Deep learning for interpretable image recognition," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *Trans. on Audio, Speech and Language Processing*, vol. 25, no. 6, pp. 1291–1303, Jun. 2017.

[14] H. Muckenhirn, V. Abrol, M. Magimai-Doss, and S. Marcel, "Understanding and Visualizing Raw Waveform-Based CNNs," in *Interspeech 2019*, 2019, pp. 2345–2349.

[15] S. Mishra, B. L. Sturm, and S. Dixon, "Local interpretable model-agnostic explanations for music content analysis." in *18th International Society for Music Information Retrieval Conference*, Suzhou, China, 2020, pp. 537–543.

[16] J. Pons, J. Serrà, and X. Serra, "Training neural audio classifiers with few data," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 16–20.

[17] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[18] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results," *arXiv e-prints*, p. arXiv:1412.1602, Dec. 2014.

[19] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.

[20] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. M. Shazeer, A. M. Dai, M. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *ICLR*, 2019.

[21] M. Won, S. Chun, and X. Serra, "Toward Interpretable Music Tagging with Self-Attention," *arXiv e-prints*, p. arXiv:1906.04972, Jun. 2019.

[22] P. Agrawal and S. Ganapathy, "Interpretable Filter Learning Using Soft Self-attention For Raw Waveform Speech Recognition," *arXiv e-prints*, p. arXiv:2001.07067, Jan. 2020.

[23] S. O. Arik and T. Pfister, "Protoattend: Attention-based prototypical learning," *Journal of Machine Learning Research*, vol. 21, no. 210, pp. 1–35, 2020.

[24] S. Sun, Q. Sun, K. Zhou, and T. Lv, "Hierarchical attention prototypical networks for few-shot text classification," in *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, Nov. 2019, pp. 476–485.

[25] P. Zinemanas, I. Hounie, P. Cancela, F. Font, M. Rocamora, and X. Serra, "DCASE-models: a Python library for computational environmental sound analysis using deep-learning models," in *Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, Nov. 2020, pp. 240–244.

[26] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello., "Scaper: A library for soundscape synthesis and augmentation," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2017.

[27] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen and learn more: Design choices for deep audio embeddings," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019, p. 3852–3856.

[28] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

COMBINING MULTIPLE DISTRIBUTIONS BASED ON SUB-CLUSTER ADACOS FOR ANOMALOUS SOUND DETECTION UNDER DOMAIN SHIFTED CONDITIONS

Kevin Wilkinghoff

Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE
Fraunhoferstraße 20, 53343 Wachtberg, Germany
kevin.wilkinghoff@fkie.fraunhofer.de

ABSTRACT

Systems based on sub-cluster AdaCos yield state-of-the-art performance on the DCASE 2020 dataset for anomalous sound detection. In contrast to the previous year, the dataset belonging to task 2 "Unsupervised Anomalous Sound Detection for Machine Condition Monitoring under Domain Shifted Conditions" of the DCASE challenge 2021 contains not only source domains with 1000 normal training samples for each machine but also so-called target domains with different acoustic conditions for which only 3 normal training samples are available. To address this additional problem, a novel anomalous sound detection system based on sub-cluster AdaCos for the DCASE challenge 2021 is presented. The system is trained to extract embeddings whose distributions are estimated in different ways for source and target domains, and utilize the resulting negative log-likelihoods as anomaly scores. In experimental evaluations, it is shown that the presented system significantly outperforms both baseline systems on the source and target domains of the development set. On the evaluation set of the challenge, the proposed system is ranked third among all 27 teams' submissions.

***Index Terms*—** anomalous sound detection, machine listening, representation learning, angular margin loss, domain shift

1. INTRODUCTION

The goal of semi-supervised anomalous sound detection is to decide whether a given audio sample resembles the training data i.e. is *normal* or substantially differs from the training data and thus is *anomalous*. Basically, one can distinguish two major strategies for anomalous sound detection: The first approach is based on training autoencoders to encode normal data into a lower-dimensional space and then reconstruct it again [1, 2]. The underlying assumption is that normal data can be reconstructed well after training while anomalous data cannot, leading to a higher reconstruction error. Thus, the reconstruction error can be used as an anomaly score. The second approach is to train neural networks to discriminate among classes as for example machine types and utilize the trained neural network to extract representation of the data, so-called embeddings, as features [3, 4, 5, 6, 7]. Here, the assumption is that the information needed to discriminate among the classes and thus is contained in the embeddings is also sufficient to distinguish normal from anomalous data. Angular margin losses such as ArcFace [8] or AdaCos [9], which ensure a margin between the classes, have been shown to outperform standard softmax losses in this context. To our knowledge, the best performing system on the anomalous sound detection dataset belonging to task 2 of the DCASE challenge 2020 [10] uses an extension of AdaCos, called sub-cluster AdaCos [11]. This loss learns more than a single mean value for each class to estimate less restrictive distributions of the embeddings than standard AdaCos and utilizes Gaussian mixture models (GMMs) to estimate these distributions for the normal data instead of comparing embeddings to the learned mean values by using the cosine similarity. This superior performance is the reason why this work focuses entirely on a system based on the sub-cluster AdaCos loss.

The system presented in this paper is designed for and submitted to task 2 "Unsupervised Anomalous Sound Detection for Machine Condition Monitoring under Domain Shifted Conditions" of the DCASE challenge 2021 [12]. The dataset of this task consists of audio recordings with a length of 10 seconds and a sampling rate of 16 kHz belonging to the machine types "ToyCar" and "ToyTrain" from ToyADMOS2 [13] and the machines types "fan", "gearbox", "pump", "slide rail" and "valve" from MIMII DUE [14]. The organizers of the challenge also provided two baseline systems: An autoencoder, which is the same as the baseline system of the previous edition of the task, and a discriminatively trained MobileNetV2-based baseline that predicts the section, which is a subset of the data within one machine type, a given audio sample belongs to.

In contrast to the DCASE challenge 2020, there are several differences for this year's task: First and foremost, the dataset is split into source domains for which about 1000 normal training samples are provided for each of the 6 sections per machine type and so-called target domains for the same sections with different acoustic conditions than the source domains for which only 3 normal training samples are available. For both domains, the same number of test samples is provided, about 100 normal samples and 100 anomalous samples. Furthermore, the dataset is split into a development set consisting of half of the sections and an evaluation set consisting of the other half of the sections. Another difference between the datasets is, that the sections do not directly correspond to specific products of a machine type but the same products can appear in different sections or different products can appear in the same sections. Both of these changes make the task much more challenging than before. Last but not least, the DCASE 2020 dataset consists of slightly different machine types, namely "ToyCar" and "ToyConveyor" from the ToyADMOS dataset [15] and the machine types "fan", "pump", "slide rail" and "valve" from the MIMII dataset [16].

The goal of this work is to investigate how to utilize the sub-cluster AdaCos loss for the DCASE 2021 anomalous sound detection dataset with its novel challenges. To this end, a system based on the sub-cluster AdaCos loss is presented. As a second contribution, different ways to compute anomaly scores for the source and target domains are proposed. Furthermore, it is shown how to decide whether samples are normal or anomalous only based on these

layer name	structure	output size
input	-	313×128
2D convolution	7×7, stride$= 2$	$157 \times 64 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride$= 1$	$78 \times 31 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride$= 1$	$39 \times 16 \times 32$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride$= 1$	$20 \times 8 \times 64$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride$= 1$	$10 \times 4 \times 128$
max pooling	10×1, stride$= 1$	4×128
flatten	-	512
dense (representation)	linear	128
sub-cluster AdaCos	-	42
sub-cluster AdaCos	-	199

Table 1: Modified ResNet architecture used for all experiments.

scores from normal data. In experimental evaluations, it is shown that the proposed system significantly outperforms both baseline systems on the source and target domains of the development set.

2. METHODOLOGY

2.1. Sub-Cluster AdaCos loss

The standard AdaCos [9] loss is an angular margin loss that uses an adaptive scale parameter and thus does not require any hyperparameter to be set. For detecting anomalous sounds, this loss has been extended to the sub-cluster AdaCos loss [11] that learns multiple mean values for each class instead of a single one. This loss has been shown to significantly outperform the standard AdaCos loss on the DCASE 2020 dataset and thus is the only loss used for training the proposed system. The probability of sample $x_i \in \mathbb{R}^D$ belonging to class j of the $C \in \mathbb{N}$ classes is given by

$$\hat{P}_{i,j} := \sum_{l \in \mathcal{M}^{(j)}} \frac{\exp(\hat{s} \cdot \cos\theta_{i,l})}{\sum_{k=1}^{CS} \exp(\hat{s} \cdot \cos\theta_{i,k})} \tag{1}$$

where $\mathcal{M}^{(j)}$ denotes all sub-clusters belonging to class j, $S \in \mathbb{N}$ denotes the number of sub-clusters, $\hat{s}$ denotes the adaptive scale parameter as defined in [11], and $\theta_{i,k} \in [0, \pi]$ denotes the angle defined through the cosine similarity $\cos\theta_{i,k} = \langle x_i, W_k \rangle / \|x_i\| \|W_k\|$ for a learned class center $W_k \in \mathbb{R}^D$.

2.2. Data preprocessing

To compute input features for the neural network, log-Mel spectrograms with 128 Mel-bins, a window size of 1024 and a hop size of 512 are extracted from all raw waveforms with a sampling rate of 16 kHz resulting in features of size 313×128. These features are then standardized by subtracting the temporal mean and dividing by the temporal standard deviation estimated from all training files.

2.3. Neural network architecture

The network architecture used throughout this work is the same as used in [11] and can be found in Tab. 2.3. It consists of several residual blocks [17] whose output is further processed by max-pooling over time, flattening and using a final linear dense layer to obtain the embeddings of size $D = 128$. In each residual block, batch normalization [18] is applied and LeakyReLu [19] with $\alpha = 0.1$ is used as a non-linear transfer function.

To train the neural network, two sub-cluster AdaCos losses with equal weights are minimized using Adam [20]. One is for classifying jointly among the sections and machine types and the other one for classifying among the different attribute information given in the filenames. When training, all normal data contained in the training set and the additional training set has been used resulting in a total of 42 sections and 199 different types of attribute information. Furthermore, mixup [21] is used during training to avoid overfitting of the model to the training data. The network is implemented in Tensorflow [22] and trained for 400 epochs with a batch size of 64.

2.4. Calculating anomaly scores

Throughout this work, all anomaly scores are computed by training Gaussian mixture models (GMMs) on the embeddings and utilizing negatively weighted log-likelihoods as scores. In [11], it has been shown that using GMMs to estimate the underlying distribution of the embeddings outperforms other backends such as using cosine similarity to the class means. Unless stated otherwise, all GMMs are realized using scikit-learn [23], initialized with the learned mean values of the sub-cluster AdaCos loss, and have a regularized covariance matrix by adding 10^{-3} to the diagonal. To calculate the anomaly scores, two different strategies for the source and target domain are used.

For the source domain, one GMM is trained for all normal data of the source domain belonging to a single section. Another GMM is trained for all normal data of the source domain labeled with different attribute information. Let $x \in \mathbb{R}^D$ denote an embedding, $s(x) \in \mathcal{S}$ denote its section and $a(s(x)) \subset \mathcal{A}$ denote all attribute information that are present in this section. Then, the anomaly score $Z_{\text{source}}(x)$ for x is the given by

$$\begin{aligned} Z_{\text{source}}(x) := &- \max_k \log P(x|s(x), k) \\ &- \max_k \max_{a \in a(s(x))} \log P(x|a, k) \end{aligned} \tag{2}$$

where $P(.|s, k), P(.|a, k)$ denote the weighted likelihoods of component k of the GMMs trained for section $s \in \mathcal{S}$ and target information $a \in \mathcal{A}$, respectively.

For the target domain, the same GMMs trained on the normal data of the target domain belonging to single sections are used. Furthermore, another GMM with three components is trained on the three target samples and thus the cosine distance to the closest normal target sample is also utilized. Using the above notation, the anomaly score $Z_{\text{target}}(x)$ for embedding x is given by

$$\begin{aligned} Z_{\text{target}}(x) := &- \max_k \log P(x|s(x), k) \\ &- \max_{k=1,2,3} \log P(x|X_{\text{target}}(s(x)), k) \end{aligned} \tag{3}$$

where $X_{\text{target}}(s(x)) \subset \mathbb{R}^D$ denotes the normal training samples of the target domain belonging to the section of x.

In [11], it has also been shown that a simple representation derived from the input features leads to surprisingly good performance for the machine type "valve" on the DCASE 2020 dataset. This is the reason why an additional term based on the temporal maximum of the log-Mel spectrogram, denoted by $t_{\max}(x) \in \mathbb{R}^D$ for embedding x, is introduced when calculating the anomaly score for the source domain of the machine type "valve". To this end, a

GMM with a single Gaussian component is trained and the altered anomaly score $\widetilde{Z}_{\text{source}}(x)$ for embedding x belonging to machine type "valve" is given by

$$\widetilde{Z}_{\text{source}}(x) := Z_{\text{source}}(x) - \max_{a \in a(s(x))} \log P_{t_{\max}}(t_{\max}(x)|a) \quad (4)$$

where $P_{t_{\max}}(.|a)$ denotes the weighted likelihoods of the single Gaussian trained on the temporal maxima of the log-Mel spectrograms belonging to target information $a \in \mathcal{A}$.

2.5. Ensembling strategy

As done in [11], the proposed neural network for extracting the embeddings is trained with a different number of sub-clusters ranging from 2^0 to 2^4. The same value is used for both sub-cluster AdaCos losses. Thus, there are 5 differently trained versions of each network to extract embeddings. Furthermore, after each 100 epochs of training, the embeddings are extracted and GMMs are trained to calculate the anomaly detection scores. Then, all of these scores are summed-up resulting in 4 subsystems for each network with a specified number of sub-clusters and hence an ensemble consisting of a total of $4 \times 5 = 20$ models.

In addition to that, the described ensembling procedure is repeated by using only a single sub-cluster AdaCos loss classifying among the sections and machine types only and thus removing the second sub-cluster AdaCos loss. This led to slightly better performance for some machine types and to slightly worse performance for other machine types. To obtain anomaly scores for each machine type, the single system is used that led to better performance for the given machine type. More concretely, for the machine types "ToyCar", "ToyTrain", "pump" and "slide rail" the anomaly scores obtained by using the model trained on both losses are used and for "fan", "gearbox" and "valve" the anomaly scores obtained with the models trained on only a single loss are used.

2.6. Setting decision thresholds

Next, it is described how decision thresholds for deciding whether a given test sample is normal or anomalous solely based on the anomaly score are obtained. To this end, the 90th percentile of the anomaly scores of all normal training samples belonging to a given section and a given domain is calculated. Then, all anomaly scores of test samples belonging to the same section and domain that are above this threshold are marked as *anomalous*. For the source domain, $Z_{\text{source}}(x)$ as defined in Eq. (1) is used but for the target domain, only the first term of $Z_{\text{target}}(x)$ is used. The reason is that the likelihoods from the second term belonging to the training data are inappropriately high since the three means of the corresponding GMM are initialized as the three training samples. Hence, when also using the second term of Eq. (2) the decision threshold would also be overestimated and thus all test data samples belonging to the target domain would be considered anomalous.

3. EXPERIMENTAL RESULTS

3.1. Performance on the development set

The experimental results obtained with the proposed system on the development set compared to both baseline systems can be found in Tab. 2. It can be seen that the proposed system significantly outperforms both baseline systems, which both have roughly the same overall performance, on the source and target domains. However, the improvement in terms of AUC is much greater than for pAUC. For nearly all dataset splits the proposed system has a higher AUC than both baseline systems. But for some dataset splits the MobileNetV2-based baseline system has a higher pAUC than the proposed system. For the machine type "gearbox" the harmonic mean of all pAUCs belonging to the proposed system is even slightly worse than the harmonic mean of the MobileNetV2-based baseline system.

dataset split			baseline				proposed system	
			autoencoder		MobileNetV2			
machine type	section	domain	AUC	pAUC	AUC	pAUC	AUC	pAUC
ToyCar	0	source	67.63%	51.87%	66.56%	66.47%	75.03%	60.32%
ToyCar	1	source	61.97%	51.82%	71.58%	66.44%	89.30%	73.26%
ToyCar	2	source	74.36%	55.56%	40.37%	47.48%	92.27%	75.32%
ToyCar	0	target	54.50%	50.52%	61.32%	52.61%	92.41%	78.11%
ToyCar	1	target	64.12%	51.14%	72.48%	63.99%	78.34%	61.47%
ToyCar	2	target	56.57%	52.61%	45.17%	48.85%	56.78%	57.26%
ToyCar	harmonic mean		62.49%	52.36%	56.04%	56.37%	78.37%	66.64%
ToyTrain	0	source	72.67%	69.38%	69.84%	54.43%	95.62%	88.74%
ToyTrain	1	source	72.65%	62.52%	64.79%	54.09%	90.67%	76.26%
ToyTrain	2	source	69.91%	47.48%	69.28%	47.66%	87.78%	47.37%
ToyTrain	0	target	56.07%	50.62%	46.28%	51.27%	71.32%	48.47%
ToyTrain	1	target	51.13%	48.60%	53.38%	49.60%	49.76%	50.89%
ToyTrain	2	target	55.57%	50.79%	51.42%	53.40%	92.98%	76.58%
ToyTrain	harmonic mean		61.71%	53.81%	57.46%	51.61%	77.17%	60.71%
fan	0	source	66.69%	57.08%	43.62%	50.45%	72.17%	62.95%
fan	1	source	67.43%	50.72%	78.33%	78.37%	89.24%	84.68%
fan	2	source	64.21%	53.12%	74.21%	76.80%	83.03%	74.58%
fan	0	target	69.70%	55.13%	53.34%	56.01%	55.30%	48.47%
fan	1	target	49.99%	48.49%	78.12%	66.41%	87.68%	74.58%
fan	2	target	66.19%	56.93%	60.35%	60.97%	72.43%	70.63%
fan	harmonic mean		63.24%	53.38%	61.56%	63.02%	74.66%	67.34%
gearbox	0	source	56.03%	51.59%	81.35%	70.46%	85.80%	74.56%
gearbox	1	source	72.77%	52.30%	60.74%	53.88%	85.37%	52.54%
gearbox	2	source	58.96%	51.82%	71.58%	62.23%	61.39%	48.23%
gearbox	0	target	74.29%	55.67%	75.02%	64.77%	81.93%	69.25%
gearbox	1	target	72.12%	51.78%	56.27%	53.30%	86.02%	51.29%
gearbox	2	target	66.41%	53.66%	64.45%	55.58%	65.26%	49.92%
gearbox	harmonic mean		65.97%	52.76%	66.70%	59.16%	76.13%	56.00%
pump	0	source	67.48%	61.83%	64.09%	62.40%	77.49%	63.47%
pump	1	source	82.38%	58.29%	86.27%	66.66%	98.26%	91.21%
pump	2	source	63.93%	55.44%	53.70%	50.98%	78.56%	63.68%
pump	0	target	58.01%	51.53%	59.09%	53.96%	57.00%	51.74%
pump	1	target	47.35%	49.65%	71.86%	62.69%	88.82%	62.42%
pump	2	target	62.78%	51.67%	50.16%	51.69%	72.88%	57.26%
pump	harmonic mean		61.92%	54.41%	61.89%	57.37%	76.59%	63.00%
slide rail	0	source	74.09%	52.45%	61.51%	53.97%	96.28%	83.16%
slide rail	1	source	82.16%	60.29%	79.97%	55.62%	93.73%	69.16%
slide rail	2	source	78.34%	65.16%	79.86%	71.88%	84.01%	77.30%
slide rail	0	target	67.22%	57.32%	51.96%	51.96%	82.06%	60.63%
slide rail	1	target	66.94%	53.08%	46.83%	52.02%	62.92%	49.76%
slide rail	2	target	46.20%	50.10%	55.61%	55.71%	72.39%	55.72%
slide rail	harmonic mean		66.74%	55.94%	59.26%	56.00%	80.16%	63.86%
valve	0	source	50.34%	50.82%	58.34%	54.97%	81.47%	63.00%
valve	1	source	53.52%	49.33%	53.57%	50.09%	90.09%	64.37%
valve	2	source	59.91%	51.96%	56.13%	51.69%	98.07%	91.47%
valve	0	target	47.12%	48.68%	52.19%	51.54%	65.86%	64.74%
valve	1	target	56.39%	53.88%	68.59%	57.83%	81.38%	58.58%
valve	2	target	55.16%	48.97%	53.58%	50.86%	77.42%	56.95%
valve	harmonic mean		53.41%	50.54%	56.51%	52.64%	81.13%	64.92%
all	harmonic mean		61.93%	53.27%	59.72%	56.37%	77.69%	62.99%

Table 2: AUCs and pAUCs (with $p = 0.1$) obtained with the baseline systems and the proposed system on the development set. Highest AUCs and pAUCs in each row are underlined.

Next, the performances of all distributions and losses used for the subsystems of the ensemble have been evaluated and compared on the development set. The results can be found in Tab. 3 and Tab. 4, respectively. For the distributions it can be seen that in most cases the distribution conditioned on the sections has a lower AUC but a higher pAUC than the distribution conditioned on the file endings and the distribution conditioned on the target samples. One exception to this is the machine type "gearbox" for which $P(.|s,k)$ performs best and "ToyCar" and "ToyConveyor"

dataset split		distribution $P(.\|s,k)$		$P(.\|a,k)$		$P(.\|X_{\text{target}}(s),k)$		proposed ensemble	
machine type	domain	AUC	pAUC	AUC	pAUC	AUC	pAUC	AUC	pAUC
ToyCar	source	82.49%	65.82%	86.22%	69.39%	-	-	85.04%	69.25%
ToyCar	target	65.02%	58.45%	-	-	76.02%	64.13%	73.21%	64.55%
ToyTrain	source	90.54%	65.60%	91.55%	66.11%	-	-	91.30%	66.01%
ToyTrain	target	54.03%	53.21%	-	-	68.13%	57.29%	66.81%	56.14%
fan	source	81.05%	73.34%	80.55%	72.12%	-	-	80.98%	73.20%
fan	target	67.52%	65.71%	-	-	69.17%	60.20%	69.30%	62.20%
gearbox	source	75.20%	58.63%	75.66%	55.65%	-	-	75.88%	56.46%
gearbox	target	81.02%	57.93%	-	-	75.68%	55.34%	76.64%	55.59%
pump	source	83.63%	70.60%	83.83%	70.83%	-	-	83.70%	70.66%
pump	target	69.90%	60.11%	-	-	70.16%	55.51%	70.52%	56.81%
slide rail	source	91.10%	76.23%	90.97%	75.59%	-	-	91.11%	76.20%
slide rail	target	65.90%	56.17%	-	-	71.63%	54.99%	71.56%	54.94%
valve	source	83.85%	73.84%	89.34%	68.89%	-	-	89.33%	70.77%
valve	target	71.66%	61.64%	-	-	73.31%	59.16%	74.26%	60.00%
all	source	83.67%	68.66%	85.09%	67.81%	-	-	85.00%	68.38%
all	target	67.00%	58.80%	-	-	71.90%	57.93%	71.63%	58.41%

Table 3: Harmonic means of AUCs and pAUCs (with $p = 0.1$) obtained with different distributions on the development set. Highest AUCs and pAUCs in each row are underlined.

dataset split		sub-cluster AdaCos losses for sections		sections and file endings		proposed ensemble	
machine type	domain	AUC	pAUC	AUC	pAUC	AUC	pAUC
ToyCar	source	74.49%	60.67%	88.58%	71.58%	85.04%	69.25%
ToyCar	target	67.44%	61.55%	75.53%	65.84%	73.21%	64.55%
ToyTrain	source	86.88%	62.00%	91.15%	67.03%	91.30%	66.01%
ToyTrain	target	63.73%	54.18%	68.10%	56.43%	66.81%	56.14%
fan	source	81.66%	73.16%	80.22%	72.60%	80.98%	73.20%
fan	target	69.11%	62.55%	69.00%	61.16%	69.30%	62.20%
gearbox	source	74.97%	57.04%	75.34%	56.16%	75.88%	56.46%
gearbox	target	78.53%	59.53%	72.02%	51.90%	76.64%	55.59%
pump	source	83.44%	69.75%	83.75%	71.35%	83.70%	70.66%
pump	target	68.77%	56.31%	71.30%	56.34%	70.52%	56.81%
slide rail	source	90.44%	74.91%	90.99%	76.43%	91.11%	76.20%
slide rail	target	68.84%	54.44%	73.13%	55.41%	71.56%	54.94%
valve	source	88.90%	72.11%	89.51%	68.95%	89.33%	70.77%
valve	target	75.41%	61.12%	72.36%	56.55%	74.26%	60.00%
all	source	82.54%	66.44%	85.26%	68.58%	85.00%	68.38%
all	target	69.96%	58.34%	71.56%	57.37%	71.63%	58.41%

Table 4: Harmonic means of AUCs and pAUCs (with $p = 0.1$) obtained with different losses on the development set. Highest AUCs and pAUCs in each row are underlined.

for which $P(.|s,k)$ yields significantly lower AUCs and pAUCs. The ensemble obtained by taking the mean of the distributions performs relatively close to the best performing distribution for each machine type. For the two losses the same three machine types stated before have the largest differences in performance. When only using the sections for the loss, the AUCs and pAUCs are higher for the machine type "gearbox" but lower for machine types "ToyCar" and "ToyConveyor" than when also using the file endings for the loss. For each of the other machine types, no loss is preferable since both perform better than the other one for some classes but the differences in performance are relatively small compared to the three machine types mentioned before. Again, the ensemble yields results close to the best performing loss for each machine type and thus seems to combine the strengths of both losses.

3.2. Performance on the evaluation set

The experimental results obtained on the evaluation set compared to both baseline systems can be found in Tab. 5. Overall, it can be seen that the proposed system significantly outperforms both baseline systems. However, for some machine types as for example "gearbox" or the target domains of "ToyTrain", the autoencoder-based baseline system has a much better performance than the proposed system. This shows that not all information needed to detect anomalies is captured sufficiently well in the discriminatively trained embeddings but an autoencoder is able to identify these anomalous structures. Hence, also utilizing an autoencoder structure for training the embeddings may be helpful to further improve the results.

dataset split		baseline autoencoder		baseline MobileNetV2		proposed system	
machine type	domain	AUC	pAUC	AUC	pAUC	AUC	pAUC
ToyCar	source	76.33%	51.26%	34.32%	53.49%	67.07%	63.05%
ToyCar	target	58.02%	53.42%	56.62%	58.89%	72.83%	63.77%
ToyTrain	source	69.89%	55.49%	47.30%	52.49%	70.87%	56.19%
ToyTrain	target	67.18%	59.78%	39.27%	48.75%	48.38%	52.39%
fan	source	66.58%	51.36%	70.88%	57.76%	89.07%	69.85%
fan	target	55.74%	49.68%	59.96%	58.53%	88.89%	70.55%
gearbox	source	67.81%	55.71%	53.16%	53.47%	61.19%	50.97%
gearbox	target	63.32%	58.06%	49.27%	49.83%	54.68%	49.40%
pump	source	62.75%	51.18%	67.12%	60.77%	70.89%	65.52%
pump	target	54.43%	50.79%	68.85%	59.79%	79.20%	67.81%
slide rail	source	64.13%	50.91%	73.06%	60.47%	88.06%	64.38%
slide rail	target	51.65%	51.92%	72.78%	60.94%	85.66%	69.69%
valve	source	51.56%	50.89%	54.71%	53.03%	73.19%	55.97%
valve	target	52.19%	49.27%	51.64%	50.10%	54.90%	51.47%
all	source	64.76%	52.32%	53.82%	55.73%	73.13%	60.21%
all	target	57.03%	53.01%	54.80%	54.80%	65.76%	59.47%
all	both	56.38%		54.77%		64.20%	

Table 5: Harmonic means of AUCs and pAUCs (with $p = 0.1$) obtained with the baseline systems and the proposed system on the evaluation set. Highest AUCs and pAUCs in each row are underlined.

4. CONCLUSIONS AND FUTURE WORK

In this work, a system for detecting anomalous sounds based on the sub-cluster AdaCos loss function for domain shifted conditions has been presented. The proposed system consists of multiple discriminatively trained neural networks for extracting embeddings from log-Mel spectrograms and utilizes multiple GMMs for estimating distributions of the normal embeddings. These estimated distributions are then used to calculate log-likelihoods for test data that are combined into actual anomaly scores. To decide whether a given test sample is anomalous or normal, individual decision thresholds for each section are computed by taking the 90th percentile from the log-likelihoods of the corresponding normal training samples. In experimental evaluations conducted on the dataset of task 2 of the DCASE challenge 2021, it has been shown that the proposed system significantly outperforms both baseline systems of the challenge in terms of AUC and pAUC on source and target domains of the development and evaluation set. Moreover, the system ranked third among all 27 teams that participated in this challenge.

For the near future, it is planned to reduce the size of the ensemble and thus simplify the presented system. This can be achieved by identifying subsystems that only result in marginal gains and therefore can be removed from the ensemble. Furthermore, the performance can probably be improved by incorporating an autoencoder into the proposed system. Instead of building an even larger ensemble, it seems promising to use an additional reconstruction loss and therefore enforce the embeddings to capture more information about the structure of the data as for example done in [24, 25].

5. REFERENCES

[1] S. Kapka, "ID-conditioned auto-encoder for unsupervised anomaly detection," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 71–75.

[2] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Conformer-based sound event detection with semi-supervised learning and data augmentation," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 100–104.

[3] T. Inoue, P. Vinayavekhin, S. Morikuni, S. Wang, T. Hoang Trong, D. Wood, M. Tatsubori, and R. Tachibana, "Detection of anomalous sounds for machine condition monitoring using classification confidence," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 66–70.

[4] R. Giri, S. V. Tenneti, F. Cheng, K. Helwani, U. Isik, and A. Krishnaswamy, "Self-supervised classification for detecting anomalous sounds," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 46–50.

[5] Q. Zhou, "ArcFace based sound mobilenets for DCASE 2020 task 2," DCASE2020 Challenge, Tech. Rep., 2020.

[6] J. A. Lopez, H. Lu, P. Lopez-Meyer, L. Nachman, G. Stemmer, and J. Huang, "A speaker recognition approach to anomaly detection," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 96–99.

[7] K. Wilkinghoff, "Using look, listen, and learn embeddings for detecting anomalous sounds in machine condition monitoring," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 215–219.

[8] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4690–4699.

[9] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 10 823–10 832.

[10] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 81–85.

[11] K. Wilkinghoff, "Sub-cluster AdaCos: Learning representations for anomalous sound detection," in *International Joint Conference on Neural Networks (IJCNN)*, 2021.

[12] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and discussion on DCASE 2021 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring under domain shifted conditions," *In arXiv e-prints: 2106.04492, 1–5*, 2021.

[13] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, "ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions," *arXiv preprint arXiv:2106.02369*, 2021.

[14] R. Tanabe, H. Purohit, K. Dohi, T. Endo, Y. Nikaido, T. Nakamura, and Y. Kawaguchi, "MIMII DUE: Sound dataset for malfunctioning industrial machine investigation and inspection with domain shifts due to changes in operational and environmental conditions," *In arXiv e-prints: 2006.05822, 1–4*, 2021.

[15] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection," in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 313–317.

[16] H. Purohit, R. Tanabe, T. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, "MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. New York University, 2019, pp. 209–213.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.

[18] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 448–456.

[19] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *30th International Conference on Machine Learning (ICML)*, 2013.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2015.

[21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations (ICLR)*, 2018.

[22] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] X. Cai, H. Dinkel, Z. Yan, Y. Wang, J. Zhang, and Y. Wang, "The small rice camera ready submission to the DCASE2021," DCASE2021 Challenge, Tech. Rep., 2021.

[25] H. Narita and A. Tamamori, "Unsupervised anomalous sound detection using intermediate representation of trained models and metric learning based variational autoencoder," DCASE2021 Challenge, Tech. Rep., 2021.

EVALUATING OFF-THE-SHELF MACHINE LISTENING AND NATURAL LANGUAGE MODELS FOR AUTOMATED AUDIO CAPTIONING

Benno Weck[1,2]*, Xavier Favory*[2]*, Konstantinos Drossos*[3]*, Xavier Serra*[2]

[1] Huawei Technologies, Munich Research Center, Germany
{firstname.lastname}@huawei.com
[2] Music Technology Group, Universitat Pompeu Fabra, Spain
benno.weck01@estudiant.upf.edu,{firstname.lastname}@upf.edu
[3] Audio Research Group, Tampere University, Finland
{firstname.lastname}@tuni.fi

ABSTRACT

Automated audio captioning (AAC) is the task of automatically generating textual descriptions for general audio signals. A captioning system has to identify various information from the input signal and express it with natural language. Existing works mainly focus on investigating new methods and try to improve their performance measured on existing datasets. Having attracted attention only recently, very few works on AAC study the performance of existing pre-trained audio and natural language processing resources. In this paper, we evaluate the performance of off-the-shelf models with a Transformer-based captioning approach. We utilize the freely available Clotho dataset to compare four different pre-trained machine listening models, four word embedding models, and their combinations in many different settings. Our evaluation suggests that YAMNet combined with BERT embeddings produces the best captions. Moreover, in general, fine-tuning pre-trained word embeddings can lead to better performance. Finally, we show that sequences of audio embeddings can be processed using a Transformer encoder to produce higher-quality captions.

Index Terms— audio captioning, transfer learning, word embeddings, machine listening, transformer

1. INTRODUCTION

Automated audio captioning (AAC) is an inter-modal translation task, where existing methods take an audio signal as input and generate a textual description, i.e. a caption, of its contents [1]. The generated captions contain information about various aspects of the content of the audio signal, ranging from identification of sound events to knowledge about spatiotemporal interactions, foreground and background disambiguation, surroundings, textures, and other high-level information [2–4].

To our knowledge, all published works focusing on AAC solely employ deep learning methods [1–3, 5–17]. Most of them follow an encoder-decoder scheme and address the task as a sequence-to-sequence (seq2seq) learning problem [18]. Convolutional neural network (CNN)-based encoders are often utilized, for example, in [2, 3, 7, 8, 12, 17], and recurrent neural network (RNN)-based decoders can be used in order to generate the captions [1–3, 7, 9, 11–17]. Recently, more and more methods have involved an attention mechanism. For example, as a technique to enable the decoder to focus only on certain parts of the latent representation extracted by the encoder [1, 2, 17]. Or more generally, other approaches [6, 8, 10, 19, 20] employ a Transformer model [21]. This type of model seems particularly adequate for AAC since it led to groundbreaking results in multiple fields, such as natural language processing (NLP), computer vision, and audio processing [22].

Transfer learning is a popular technique often employed in NLP and machine listening (MaL) tasks. However, existing approaches in AAC often do not take advantage of any pre-trained resources and instead train their models from scratch. Only recently, a few published papers [2,3,5,6] propose to incorporate pre-trained audio models such as VGGish [23] or to rely on word embedding models such as word2vec [24]. Given the large number of available pre-trained models in MaL and NLP, it is still unclear which models are most suited for AAC. Moreover, incorporating these models into a Transformer-based AAC system can involve some specific design choices that are also yet overlooked.

In this paper, we focus on investigating the use of pre-trained models taken from MaL and NLP in the context of AAC. In particular, we are interested in identifying which available resources are the most valuable and how to combine them efficiently in a Transformer-based AAC system. We use various off-the-shelf pre-trained audio and word encoding methods. Our contributions are:

- We adapt a Transformer-based AAC method that can use different pre-trained MaL and NLP models,
- We conduct a thorough investigation of the performance of our method by combining pre-trained models in various settings,
- We identify what combinations of techniques make pre-trained resources specifically beneficial for an AAC system. We consider fine-tuning word embeddings, using an adapter to process audio embeddings, and the usage of overlap when extracting audio embeddings.

The rest of the paper is structured as follows: In Section 2 we present our method and in Section 3 we outline the evaluation process. The results are presented and discussed in Section 4. Finally, Section 5 concludes the paper.

2. METHOD

In our study, we adopt a Transformer-based model architecture which has been shown to produce state-of-the-art results for AAC [10, 19, 20]. An overview of our method is presented in

Figure 1. It consists of an audio encoder, E(·), an embeddings' adapter, A(·), and a decoder D(·). As E, we employ different pre-trained models for general audio processing. For A, we compare the use of no adapter, a multi-layer perceptron (MLP), and a multi-head attention (MHA) component. The output of A is used together with word embeddings of the previously predicted words as an input to D, a Transformer-based decoder[1].

A sequence of audio features $\mathbf{X} \in \mathbb{R}^{T \times F}$ with T vectors of F features is used as an input to the method, which outputs a sequence of one-hot encoded tokens $\mathbf{S} \in [0,1]^{K \times W}$, where K corresponds to the number of tokens in the generated caption and W to the size of the considered vocabulary. More specifically, $\mathbf{X}$ is used as an input to E as

$$\mathbf{Z} = \mathrm{E}(\mathbf{X}), \tag{1}$$

where $\mathbf{Z} \in \mathbb{R}^{T' \times F'}$ is a sequence of T' intermediate representations with F' features provided by the pre-trained model (i.e. an audio embedding sequence). Then, the adapter A will process $\mathbf{Z}$ as

$$\mathbf{Z}' = \mathrm{A}(\mathbf{Z}), \tag{2}$$

where $\mathbf{Z}' \in \mathbb{R}^{T' \times F''}$ and F'' is the dimensionality of the features that A outputs. Finally, the decoder D will predict the probability distribution of appearance over the W words at the k-th step, $\mathbf{S}_k$, as

$$\mathbf{S}_k = \mathrm{D}(\mathbf{Z}', \mathbf{S}'_0, \ldots, \mathbf{S}'_{k-1}), \tag{3}$$

where $\mathbf{S}'_i$ is a learned word embedding for step i, and $\mathbf{S}'_0 = \{0\}^{W'}$. As $\mathbf{S}'$, we make use of different pre-trained NLP models.

We employ different audio embedding models that are optimized for a task different from AAC. The extracted audio embeddings might contain information that is specific to the corresponding source task and not necessarily optimal for AAC. We do not fine-tune the models in our experiment, but instead, we study the usage of different adapters A that process the audio embeddings $\mathbf{Z}$.

The Transformer decoder D consists of N blocks, each of them having two serially cascaded MHA layers that perform self and cross-modal (i.e. between audio and words) attention, respectively. The output of the second, cross-modal attention, is given as an input to a linear layer and a layer normalization process. The word embeddings $\mathbf{S}'$ are used as an input to D. Following the original proposal of the Transformer model, we apply a positional encoding to the input word embeddings. To generate the captions, the decoder can be sampled until the desired caption length is met or a special token indicating the end of a sentence is produced.

3. EVALUATION

In our study, we compare the performance of four pre-trained audio processing models, two audio embedding adapters, and four pre-trained NLP models for AAC, using the AAC dataset Clotho [25].

3.1. Dataset, metrics, and experiments

The Clotho dataset contains a training, a validation, and an evaluation split, comprising 3839, 1045, and 1045 audio examples, respectively. Each audio example is annotated with five captions, and we consider one audio-caption pair a single training example. The performance is assessed using the SPIDEr score [26]. This metric is widely established in the community (e.g. in the AAC task from the 2021 DCASE Challenge[2]) as it highly correlates with the human judgment of caption quality [26].

[1]For a complete description of the Transformer decoder, refer to [21].

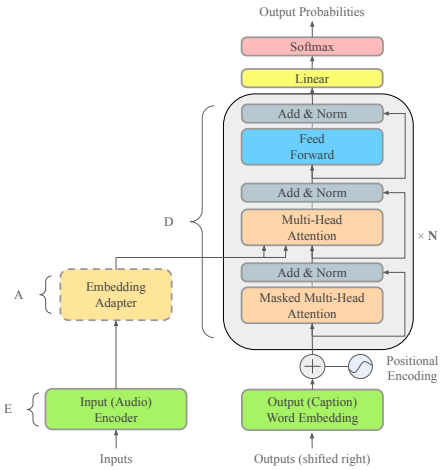


Figure 1: Model architecture.

In all our experiments, the decoder part of our model consists of $N = 3$ stacked Transformer decoder blocks with four 128-dimensional attention heads each, similar to what is used in [10,19]. During training, all models are optimized by minimizing the cross-entropy loss between the predicted sentence and the target caption using the Adam algorithm ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$) with a minibatch size of 256. Early stopping is applied after ten epochs with no improvement of the loss calculated on the held-out validation set. The best model according to this loss is then evaluated on the evaluation set.

To avoid bias in the results, we repeat all experiments ten times with different random initialization and report the mean statistic for all scores. We test all combinations of encoder models with different overlaps, audio embedding adapter layers, fixed or fine-tuned word embeddings. In total, it constitutes 264 different settings.

3.2. Audio embedding models

For each employed audio embedding model, we follow the authors' methodology to extract audio embeddings using their pre-trained models. We use the pre-trained models as encoders with frozen weights, i.e. without fine-tuning.

Existing AAC approaches use audio encoders with different hop-sizes. In this work, we study the impact of using overlap when extracting the audio embeddings. More specifically, we use two different settings for the embedding extraction hop-size, corresponding to 50% overlap and no overlap.

Table 1: Audio encoder models compared in this study.

Encoder	Dimensionality	Window	Learning
VGGish	128	0.96 s	supervised
YAMNet	1024	0.96 s	supervised
OpenL3	512	1.00 s	self-supervised
COALA	1152	2.20 s	contrastive

[2]`http://dcase.community/challenge2021/task-automatic-audio-captioning`

Table 1 gives an overview of the audio source models we compared. All four are CNN-based models. The first, **VGGish** [23], is inspired by the VGG architecture mainly used in computer vision [27]. It was trained on a preliminary version of the YouTube-8M dataset in a supervised fashion [28]. It extracts 128-dimensional embeddings from roughly 1 second of audio. The second audio embedding model is **YAMNet**, which also draws inspiration from computer vision models [23]. It employs a MobileNet [29] architecture to extract embeddings with dimensionality 1024 from almost 1 second of audio. The model was trained to predict 521 audio event classes on the AudioSet dataset [30]. The third model, **OpenL3** [31], is a modified and freely available version of the L^3-Net [32]. OpenL3 is trained in a self-supervised way in an audio-visual correspondence task, relying on videos from AudioSet. From the multiple variants that the authors provide, we chose the model configuration that produces an embedding of size 512, and that was trained with 128 Mel bands as input representation in the environmental sound setting. The fourth and final model is **COALA** [33], a model trained by taking advantage of user-provided tags in Freesound[3] [34]. During training, it employed a contrastive learning approach to align audio and associated tag embeddings, producing an audio embedding model that can extract semantically enriched audio representations. The model produces embeddings from 2.2-second patches.

3.3. Adapter Layers

We compare two different adapter architectures that are depicted in Figure 2 and contrast them with applying no adapter, which we refer to as the identity function. The aim is to investigate if one kind of adapter can improve the performance of a Transformer-based AAC method. Moreover, the adapters ensure a match in dimension between the audio embeddings and the internal dimension of the decoder. It enables us to compare embeddings of different sizes — from different models — with a fixed number of decoder parameters. When not using any adapter, the first decoder layer changes in size depending on the embeddings' dimensionality.

As the first adapter, we employ a two-layer **MLP** with a hidden layer of size 256 and rectified linear unit (ReLU) as activation function, as shown in Figure 2, to compute the adapted representation with weights shared across time. The second adapter layer is an **MHA** block, followed by a linear layer and a layer normalization process, also known as a Transformer encoder layer [21]. This type of network was previously combined with a VGGish embedding model in the context of AAC [5]. It complements our decoder in such a way that our model architecture is similar to a full Transformer model for AAC [6]. The MHA block employs four attention heads of 128 dimensions. A linear dimensionality reduction function as described in [6] and a positional encoding precede it.

3.4. Word embedding models

Our first word embedding model is **word2vec** [24], which is based on the skip-gram algorithm. We use the publicly available model pre-trained on three million words and phrases from Google News.

Our second model is **GloVe** [35], which takes a different approach by learning context information from corpus-level word-word co-occurrence statistics rather than local context windows. The authors show that GloVe is an improvement over the word2vec algorithm in downstream word analogy and Named Entity Recognition tasks. We employ the publicly available model trained on the combination of a 2014 Wikipedia copy and the Gigaword 5 corpus [36], which together contain six billion tokens.

[3]The training data from COALA and Clotho are disjoint sets.

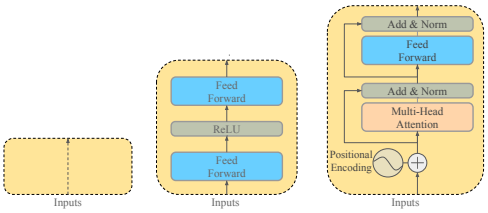


Figure 2: Embedding adapter architectures: Identity (left), MLP (middle), and MHA-based (right).

The third word embedding model is **fastText**, which implements several optimizations on top of the word2vec skip-gram algorithm [37]. FastText provides better handling of multi-word phrases, uses a weighted context, and considers subwords (i.e. character n-grams). We use the publicly available model trained with subword information on the Common Crawl corpus, which contains 600B tokens and is significantly larger than the corpora used for the Glove and word2vec model [38].

We employ **BERT** as our fourth model, which is a large language model based on the Transformer architecture and can be used as a feature extractor to extract word embeddings [39]. In contrast to models mentioned above, such as word2vec, BERT also takes the context of a token — the entire sentence — into account when extracting embeddings, i.e. producing embeddings that are context-sensitive. We use the $\text{BERT}_{\text{BASE}}$ configuration pre-trained on a Wikipedia copy (2.5B words) and the BookCorpus dataset (800M words) [40]. Different ways to use the layers of BERT as word embeddings have been discussed in the literature, and it is not clear what the best choice is for AAC. We decided to use the penultimate layer as embeddings as this can produce highly contextualized representations that are not too task-specific [41]. We extract the word embeddings from an entire caption. Due to the computational cost of the model, it will not be fine-tuned in our experiments.

Additionally, to explore if pre-trained word embeddings can be helpful, we also adopt randomly initialized word vectors and a continuous bag-of-words (CBOW) word2vec model [24] trained using the text of the captions in the Clotho training set. Finally, all word embedding models produce embeddings with $W' = 300$ dimensions, except for those from the BERT model with $W' = 768$.

4. RESULTS AND DISCUSSION

In this section, we show and discuss the results of our experiment. We organize our discussion around, first, the performance of the different pre-trained audio encoder models. Second, we discuss the usage of the different audio embedding adapter components. Third, we study the performance of different word embedding models and the impact of fine-tuning them. Finally, we show the potential of computing audio embeddings on overlapped audio frames.

Table 2 lists the optimal settings for each of the audio encoder models that we found, and Figure 3 displays box plots of the audio encoder models' performance for each adapter. The top-performing model in the best overall setting (YAMNet, BERT, MHA) achieved a SPIDEr score of 0.1914. Moreover, we found that YAMNet consistently outperforms the other audio models. Overall, audio en-

Table 2: Top-performing settings for pre-trained encoder models and their SPIDEr score when embeddings are extracted with no or 50% overlap ($\star$, and † respectively).

Encoder	Word embedding	Adapter	SPIDEr Mean	SPIDEr SD
COALA†	BERT	MHA-based	0.1495	0.0044
OpenL3*	BERT	MHA-based	0.1620	0.0051
VGGish*	BERT	MHA-based	0.1677	0.0052
YAMNet†	BERT	MHA-based	0.1793	0.0066

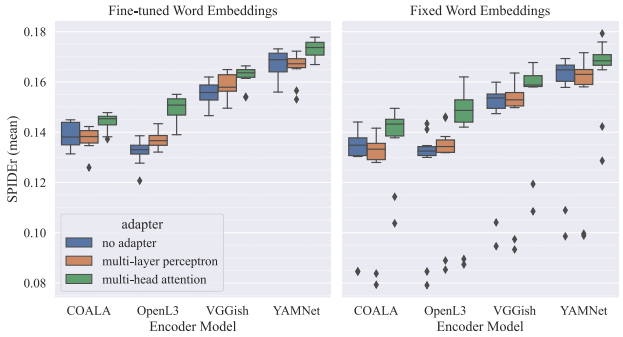


Figure 3: Comparison of SPIDEr score for different encoder models averaged over multiple different experiment settings.

coder models trained in a supervised classification task using large datasets, YAMNet & VGGish, are superior to the models that are trained in a self-supervised way or using contrastive learning. This highlights the potential of using large datasets for pre-training audio encoders in auto-tagging tasks and using them for AAC.

Using an MHA-based encoder as an adapter on top of the audio embeddings consistently provides the best results (Figure 3 and Table 2), whereas using the MLP does not provide any improvement in comparison with no adapter. Interestingly, the benefit of the MHA-based adapter is most prominent for OpenL3, which has been trained in a self-supervised way. This suggests that the audio embeddings extracted with OpenL3 contain some semantics useful for AAC that can be exploited using an adapter such as the MHA-based one. Our results suggest that employing a Transformer-based encoder using positional encoding and MHA can process sequences of audio embeddings, leading to better performance in AAC, which aligns well with findings from previous works [6, 10].

The left part of Figure 4 reports a performance comparison of the different word embeddings employed in our evaluation. On average, using the pre-trained BERT model to extract the word embeddings leads to the best performance. It is worth mentioning that we are using BERT as a fixed external word embedding model instead of using its full capacity, for example, by fine-tuning it for AAC. However, the latter would require much more computational resources (the BERT model has around 110M parameters).

Training word embedding representations from scratch during AAC provides already promising results. By using pre-trained word embeddings, such as word2vec, GloVe, and fastText, we can slightly improve this performance. Additionally, fine-tuning them can significantly improve their performance (one-sided Wilcoxon signed-rank tests $p < 0.001$, for each pre-trained word embedding model). Interestingly, optimizing the randomly initialized word

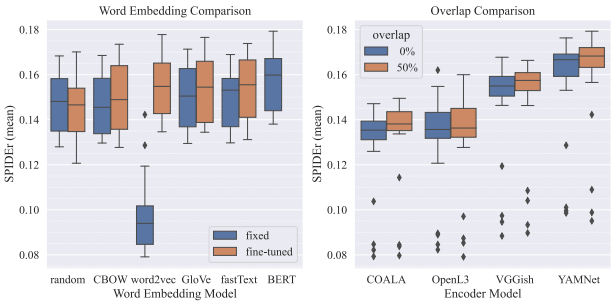


Figure 4: SPIDEr sores for word embedding (left) and audio embedding models (right). Scores are averaged over all combinations is each case.

representations does not improve their performance. This highlights the need for pre-trained word representations in the context of our AAC task.

The right side of Figure 4 displays the performance of the audio models with and without overlap when extracting the embeddings. In particular, we observe that computing the embeddings with 50% overlap leads to improved performance with two audio encoders. One-sided Wilcoxon signed-rank tests indicated that this improvement is significant for COALA ($W = 84, p < 8.08e^{-07}$) and YAMNet ($W = 112, p < 3.92e^{-06}$).

5. CONCLUSION

In this paper, we conduct a comparative analysis of many off-the-shelf resources from natural language processing (NLP) and the machine listening field for automated audio captioning (AAC). The core components of our method are a fixed audio encoder, an audio embedding adapter, and a Transformer-based decoder. Our results show that YAMNet outclasses the other audio embedding models when used as an encoder. The performance can be increased for two encoders (COALA & YAMNet) by computing the embeddings on overlapped frames. Processing the audio embeddings with a multi-head attention-based adapter can increase the performance of our captioning system while using a multi-layer perceptron is not different from not using any adapter. We found that pre-trained word embedding models are a valuable resource for AAC, particularly so when fine-tuned during the training. Using BERT as a fixed embedding extraction model gave the best results. This result motivates the usage of large pre-trained NLP models such as BERT to create better AAC methods.

Future work could investigate the impact of the positional encoding in the audio adapter by independently evaluating the multi-head attention adapter. Finally, fine-tuning the audio embedding models has not been studied in this work. However, it may be an essential technique that can benefit AAC approaches, as highlighted by the fact that adding an adaptation model to process the embeddings significantly increases the performance of our system.

6. ACKNOWLEDGMENT

K. Drossos has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, project MARVEL.

7. REFERENCES

[1] K. Drossos, S. Adavanne, *et al.*, "Automated audio captioning with recurrent neural networks," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2017, pp. 374–378.

[2] X. Xu, H. Dinkel, *et al.*, "Investigating local and global information for automated audio captioning with transfer learning," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 905–909.

[3] A. Ö. Eren and M. Sert, "Audio captioning based on combined audio and semantic embeddings," in *2020 IEEE International Symposium on Multimedia (ISM)*, 2020, pp. 41–48.

[4] S. Lipping, K. Drossos, *et al.*, "Crowdsourcing a dataset of audio captions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, October 2019, pp. 139–143.

[5] Y. Koizumi, Y. Ohishi, *et al.*, "Audio captioning using pre-trained large-scale language model guided by audio-based similar caption retrieval," *arXiv preprint arXiv:2012.07331*, 2020.

[6] Y. Koizumi, R. Masumura, *et al.*, "A transformer-based audio captioning model with keyword estimation," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association*. Shanghai, China: ISCA, Oct. 2020, pp. 1977–1981.

[7] A. Ö. Eren and M. Sert, "Audio captioning using gated recurrent units," *arXiv*, vol. abs/2006.03391, 2020.

[8] K. Chen, Y. Wu, *et al.*, "Audio captioning based on transformer and pre-trained CNN," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, Nov. 2020, pp. 21–25.

[9] K. Nguyen, K. Drossos, *et al.*, "Temporal sub-sampling of audio feature sequences for automated audio captioning," *arXiv*, vol. abs/2007.02676, 2020.

[10] A. Tran, K. Drossos, *et al.*, "Wavetransformer: An architecture for audio captioning based on learning temporal and time-frequency information," in *European Signal Processing Conference (EUSIPCO)*, Aug. 2021.

[11] E. Çakir, K. Drossos, *et al.*, "Multi-task regularization based on infrequent classes for audio captioning," *arXiv*, vol. abs/2007.04660, 2020.

[12] X. Xu, H. Dinkel, *et al.*, "A CRNN-GRU based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, Nov. 2020, pp. 225–229.

[13] D. Takeuchi, Y. Koizumi, *et al.*, "Effects of word-frequency based pre- and post- processings for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, Nov. 2020, pp. 190–194.

[14] M. Wu, H. Dinkel, *et al.*, "Audio caption: Listen and tell," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 830–834.

[15] S. Ikawa and K. Kashino, "Neural audio captioning based on conditional sequence-to-sequence model," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, Oct. 2019, pp. 99–103.

[16] X. Xu, H. Dinkel, *et al.*, "Audio caption in a car setting with a sentence-level loss," in *12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, Hong Kong, 2021, pp. 1–5.

[17] C. D. Kim, B. Kim, *et al.*, "Audiocaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, Minneapolis, MN, USA, June 2019, pp. 119–132.

[18] I. Sutskever, O. Vinyals, *et al.*, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014.

[19] Y. Wu, K. Chen, *et al.*, "Audio captioning based on transformer and pre-training for 2020 DCASE audio captioning challenge," DCASE2020 Challenge, Tech. Rep., June 2020.

[20] W. Yuan, Q. Han, *et al.*, "The DCASE 2021 challenge task 6 system: Automated audio captioning with weakly supervised pre-traing and word selection methods," DCASE2021 Challenge, Tech. Rep., 2021.

[21] A. Vaswani, N. M. Shazeer, *et al.*, "Attention is all you need," *arXiv*, vol. abs/1706.03762, 2017.

[22] T. Lin, Y. Wang, *et al.*, "A survey of transformers," *arXiv*, vol. abs/2106.04554, 2021.

[23] S. Hershey, S. Chaudhuri, *et al.*, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[24] T. Mikolov, I. Sutskever, *et al.*, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[25] K. Drossos, S. Lipping, *et al.*, "Clotho: an audio captioning dataset," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 736–740, 2020.

[26] S. Liu, Z. Zhu, *et al.*, "Improved image captioning via policy gradient optimization of spider," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 873–881, 2017.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[28] S. Abu-El-Haija, N. Kothari, *et al.*, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[29] A. G. Howard, M. Zhu, *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv*, vol. abs/1704.04861, 2017.

[30] J. F. Gemmeke, D. P. W. Ellis, *et al.*, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[31] J. Cramer, H.-H. Wu, *et al.*, "Look, listen, and learn more: Design choices for deep audio embeddings," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3852–3856.

[32] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 609–617.

[33] X. Favory, K. Drossos, *et al.*, "COALA: Co-aligned autoencoders for learning semantically enriched audio representations," in *Workshop on Self-supervision in Audio and Speech at the 37th International Conference on Machine Learning*, Vienna, Austria, 2020.

[34] F. Font, G. Roma, *et al.*, "Freesound technical demo," in *Proceedings of the 21st ACM International Conference on Multimedia*, ser. MM '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 411–412.

[35] J. Pennington, R. Socher, *et al.*, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[36] https://catalog.ldc.upenn.edu/LDC2011T07.

[37] T. Mikolov, E. Grave, *et al.*, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[38] http://commoncrawl.org/2017/06/.

[39] J. Devlin, M.-W. Chang, *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186.

[40] Y. Zhu, R. Kiros, *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[41] A. Rogers, O. Kovaleva, *et al.*, "A primer in bertology: What we know about how BERT works," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842–866, 2021.

MULTIPLE FEATURE RESOLUTIONS FOR DIFFERENT POLYPHONIC SOUND DETECTION SCORE SCENARIOS IN DCASE 2021 TASK 4

Diego de Benito-Gorron, Sergio Segovia, Daniel Ramos, Doroteo T. Toledano

AUDIAS Research Group
Universidad Autónoma de Madrid
Calle Francisco Tomás y Valiente, 11, 28049 Madrid, SPAIN
diego.benito@uam.es, sergio.segoviag@estudiante.uam.es,
daniel.ramos@uam.es, doroteo.torre@uam.es

ABSTRACT

In this paper, we describe our multi-resolution mean teacher systems for DCASE 2021 Task 4: Sound event detection and separation in domestic environments. Aiming to take advantage of the different lengths and spectral characteristics of each target category, we follow the multi-resolution feature extraction approach that we introduced for last year's edition. It is found that each one of the proposed Polyphonic Sound Detection Score (PSDS) scenarios benefits from either a higher temporal resolution or a higher frequency resolution. Additionally, the combination of several time-frequency resolutions through model fusion is able to improve the PSDS results in both scenarios. Furthermore, a class-wise analysis of the PSDS metrics is provided, indicating that the detection of each event category is optimized with different resolution points or model combinations.

Index Terms— DCASE 2021, CRNN, Mean Teacher, Multi-resolution, Model fusion, PSDS

1. INTRODUCTION

The development of competitive evaluations such as the DCASE (Detection and Classification of Acoustic Scenes and Events) Challenges, along with the introduction of datasets like Google AudioSet [1] or DESED (Domestic Environment Sound Event Detection) [2, 3], has supported the research in acoustic event detection tasks over the recent years.

DCASE 2021 Challenge Task 4 consists in the detection and classification of 10 different sound events. These sound events belong to domestic environments, and each category shows its own temporal and spectral properties. During the DCASE 2020 Challenge, we explored the idea of employing multiple time-frequency resolution points during the feature extraction process, aiming to exploit these differences, and finding that the combination of different time-frequency resolutions was beneficial for the performance of a system derived from the SED baseline, in terms of both event-based F_1 score and Polyphonic Sound Detection Score (PSDS) [4, 5, 6].

One of the advantages of our multi-resolution approach is that it is, in principle, complementary to other improvements in the model, such as a different topology of the neural network or additional training data. Taking that into account, we have applied multi-resolution to the DCASE 2021 SED baseline system, which features the use of mixup [7] for data augmentation, as well as a larger synthetic subset, as main additions to the Mean Teacher [8] convolutional recurrent neural network (CRNN) system of previous years [9].

Our participation for DCASE 2021 Challenge is based on the provided baseline system and follows the scenario of sound event detection (SED) without source separation pre-processing. We propose a multi-resolution analysis of the audio features (mel-spectrograms) used to train the neural network, in contrast with the single-resolution approach of the baseline.

Work developed under the project DSForSec (RTI2018-098091-B-I00), funded by the Ministry of Science, Innovation and Universities of Spain, and the European Regional Development Fund (ERDF).

2. DATASET

The dataset used for sound event detection in DCASE 2021 Task 4 is DESED, which is composed of real recordings, obtained from Google AudioSet, and synthetic recordings which are generated using the Scaper library [10]. Real recordings include the Weakly-labeled training set (1578 clips), the Unlabeled training set (14412 clips) and the Validation set (1168 clips). Additionally, the Synthetic set contains 12500 strongly-labeled, synthetic clips, generated such that the event distribution is similar to that of the Validation set.

The Weakly-labeled, Unlabeled and Synthetic sets are used to train the neural networks. 10% of the Weakly-labeled set and 20% of the Synthetic set are reserved for validation. The DESED Validation set is used to tune hyper-parameters and perform model selection.

3. PROPOSED SOLUTIONS

3.1. Multi-resolution analysis

The baseline system employs mel-spectrogram features, a two-dimensional representation of audio signals based on the Fast Fourier Transform (FFT) and the Mel scale. Thus, the audio segments are transformed into 2-D images that are processed through the CRNN. The process of mel-spectrogram extraction depends on several parameters: the sampling frequency of the audio (f_s), the number of points of the FFT (N), the number of mel filters (n_{mel}), the analysis window function, and its hop and length (R, L). Given a set of values for these parameters, a time-frequency resolution working point is defined.

A particular time-frequency resolution can be more or less fitted to detect a sound event category depending on its temporal and spec-

Resolution	T_{++}	T_{+}	BS	F_{+}	F_{++}
N	1024	2048	2048	4096	4096
L	1024	1536	2048	3072	4096
R	128	192	256	384	512
$\mathbf{n_{mel}}$	64	96	128	192	256

Table 1: FFT length (N), window length (L), window hop (R) and number of Mel filters (n_{mel}) of the five proposed time-frequency resolution working points. N, L, and R are reported in samples, using a sample rate $f_s = 16000$ Hz.

PSDS	DTC	GTC	α_{ST}	CTTC	α_{CT}	e_{max}
Scenario 1	0.7	0.7	1.0	0.0	-	100
Scenario 2	0.1	0.1	1.0	0.3	0.5	100

Table 2: Parameter configuration for the PSDS scenarios. DTC = Detection Tolerance Criterion. GTC = Ground Truth intersection Criterion. α_{ST} = Cost of instability across classes. CTTC = Cross-Trigger Tolerance Criterion. α_{CT} = Cost of Cross Triggers. e_{max} = Maximum False Positive Rate.

tral characteristics, which vary for each target class. For example, considering the Synthetic training set, some event categories have an average duration shorter than 2 seconds (*Alarm bell/ringing*, *Cat*, *Dishes*, *Dog*, and *Speech*), while other classes are more than 8 seconds long in average (*Electric shaver/toothbrush*, *Frying*, or *Vacuum cleaner*).

Using different mel-spectrogram configurations, we defined five different time-frequency resolution working points. For each one of them, we replicated the baseline, modifying it to handle the corresponding time-frequency resolution. Finally, we combined the frame-level estimation of the class posterior probabilities provided by each resolution, obtaining a multi-resolution system.

The reference for time-frequency resolution is the set of parameters used by the baseline system for the feature extraction process, which will be referred as BS. We maintain the sampling frequency at $f_s = 16000$ Hz and the use of a Hamming window. The rest of the parameters (N, L, R, n_{mel}) are modified to increase time or frequency resolution in each case. The resulting resolution points (T_{++}, T_{+}, BS, F_{+}, and F_{++}) are described in Table 1.

3.2. Model fusion

For each event category i, a binary classification is performed between classes $\theta_{i,0}$, which means "event i not detected", and $\theta_{i,1}$, meaning "event i detected". This classification task is considered independent of other event categories, and we will call it a detection task.

Given an audio clip, a CRNN detector generates a different score sequence for each detection task i, as a time series with a frame rate that is determined by the resolution point employed. The fusion of K different detectors consists in a combination of their sequences $(s_i^{(1)}, ..., s_i^{(K)})$. This combination is performed as a late integration, using the sigmoid outputs of each CRNN as score sequences. By convention, higher scores indicate a stronger support to the presence of event i ($\theta_{i,1}$). The final score sequence is obtained as the frame-wise average of the K score sequences.

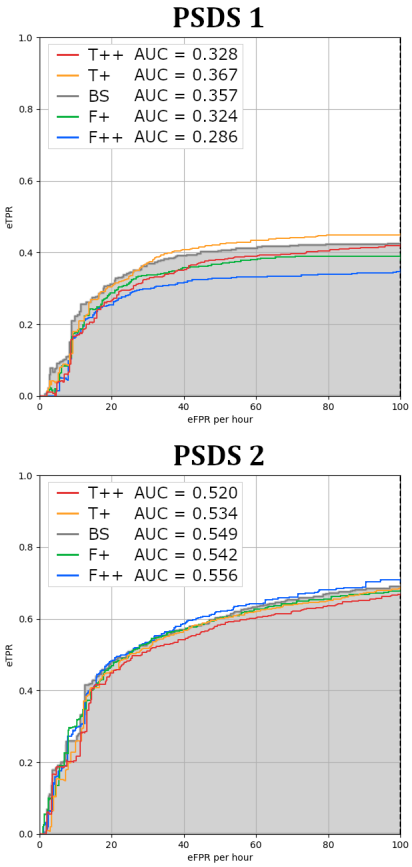


Figure 1: Polyphonic Sound Detection Score (PSDS) curves over the DESED Validation set of the single resolution F_{++}, F_{+}, BS, T_{+}, and T_{++} used to obtain the combined systems submitted to the evaluation.

System	Resolutions	PSDS 1	PSDS 2	$F_1(\%)$
3res	F_{+}, BS, T_{+}	0.380	0.589	45.0
3res-F	F_{++}, F_{+}, BS	0.361	0.589	45.1
3res-T	BS, T_{+}, T_{++}	**0.386**	0.578	**46.4**
4res	F_{++}, F_{+}, BS, T_{+}	0.372	**0.600**	45.1
5res	F_{++}, F_{+}, BS, T_{+}, T_{++}	**0.386**	**0.600**	**46.4**

Table 3: PSDS and F_1 results of multi-resolution systems over the DESED Validation set.

System	Resolutions	PSDS 1	PSDS 2	$F_1(\%)$
3res	F_{+}, BS, T_{+}	0.343	0.571	42.6
3res-T	BS, T_{+}, T_{++}	**0.363**	0.574	**43.1**
4res	F_{++}, F_{+}, BS, T_{+}	0.345	0.571	42.2
5res	F_{++}, F_{+}, BS, T_{+}, T_{++}	0.361	**0.577**	42.7
Challenge Baseline		0.315	0.547	37.3

Table 4: PSDS and F_1 results of multi-resolution systems over the DESED 2021 Evaluation set.

PSDS 1	$\mathbf{F}_{++}$	$\mathbf{F}_{+}$	**BS**	$\mathbf{T}_{+}$	$\mathbf{T}_{++}$
Alarm bell/ringing	0.446±0.009	0.512±0.022	0.556±0.015	0.561±0.012	**0.567**±0.007
Blender	**0.694**±0.021	0.627±0.008	0.677±0.018	0.652±0.029	0.671±0.028
Cat	0.378±0.020	0.414±0.004	0.411±0.011	**0.439**±0.004	0.401±0.024
Dishes	0.107±0.008	0.132±0.010	**0.176**±0.010	0.172±0.039	0.121±0.020
Dog	0.242±0.003	0.272±0.008	0.306±0.010	**0.316**±0.005	0.295±0.012
Electric shaver/toothbrush	0.787±0.027	**0.798**±0.021	0.751±0.057	0.765±0.025	0.687±0.050
Frying	0.582±0.018	0.613±0.013	0.635±0.022	**0.639**±0.021	0.607±0.023
Running water	0.481±0.026	0.510±0.006	0.540±0.014	0.548±0.020	**0.553**±0.013
Speech	0.581±0.006	0.603±0.007	**0.631**±0.004	0.634±0.009	0.620±0.006
Vacuum cleaner	0.732±0.041	0.769±0.085	0.771±0.092	0.770±0.086	**0.790**±0.068
Overall PSDS 1	0.290±0.004	0.319±0.005	0.352±0.005	**0.358**±0.015	0.331±0.005

PSDS 2	$\mathbf{F}_{++}$	$\mathbf{F}_{+}$	**BS**	$\mathbf{T}_{+}$	$\mathbf{T}_{++}$
Alarm bell/ringing	**0.855**±0.003	0.852±0.007	0.836±0.004	0.842±0.004	0.814±0.011
Blender	**0.851**±0.006	0.783±0.016	0.799±0.014	0.782±0.014	0.791±0.016
Cat	**0.717**±0.011	0.705±0.009	0.661±0.015	0.665±0.014	0.622±0.016
Dishes	**0.394**±0.022	0.376±0.019	0.388±0.013	0.374±0.065	0.389±0.021
Dog	0.666±0.014	**0.672**±0.017	0.661±0.007	0.643±0.017	0.604±0.017
Electric shaver/toothbrush	**0.938**±0.020	0.913±0.017	0.885±0.016	0.912±0.015	0.851±0.011
Frying	0.771±0.018	0.780±0.009	**0.795**±0.019	**0.795**±0.012	0.759±0.018
Running water	0.714±0.011	0.714±0.014	0.749±0.012	0.750±0.015	**0.755**±0.015
Speech	0.830±0.007	0.821±0.007	**0.834**±0.007	0.822±0.009	0.813±0.006
Vacuum cleaner	0.892±0.006	**0.902**±0.011	0.886±0.013	0.879±0.014	0.873±0.018
Overall PSDS 2	**0.557**±0.009	0.544±0.013	0.553±0.007	0.544±0.029	0.534±0.012

	$\mathbf{F}_{++}$	$\mathbf{F}_{+}$	**BS**	$\mathbf{T}_{+}$	$\mathbf{T}_{++}$
Macro F_1 (%)	33.94±0.77	38.26±0.77	**42.58**±0.90	42.20±1.19	41.86±0.79

Table 5: PSDS (scenarios 1 and 2) results for each event category and overall PSDS and F_1 scores of single-resolution systems over the DESED Validation set. Mean and standard deviations are computed across 5 trainings of each system with different random initializations.

	PSDS 1					**PSDS 2**				
	3res	**3res-T**	**4res**	**5res**	**5×BS**	**3res**	**3res-T**	**4res**	**5res**	**5×BS**
Alarm bell/ringing	0.572	**0.584**	0.558	0.577	0.576	0.858	0.855	**0.870**	**0.870**	0.852
Blender	0.724	0.744	0.746	**0.768**	0.727	0.840	0.838	0.853	**0.856**	0.841
Cat	0.455	**0.472**	0.435	0.457	0.428	0.701	0.667	**0.727**	0.712	0.681
Dishes	0.202	0.200	0.197	**0.214**	0.206	0.415	0.402	0.435	**0.436**	0.419
Dog	0.319	**0.327**	0.312	0.324	0.326	0.693	0.681	**0.701**	0.700	0.689
Electric shaver/toothb.	0.740	0.695	0.739	0.714	**0.783**	0.902	0.909	**0.918**	0.916	0.917
Frying	0.677	**0.682**	0.668	0.674	0.678	**0.841**	0.836	0.829	0.833	0.832
Running water	0.567	**0.574**	0.562	0.569	0.560	0.775	**0.780**	0.771	0.775	0.772
Speech	0.661	**0.673**	0.659	0.666	0.658	0.851	**0.857**	0.852	0.855	0.850
Vacuum cleaner	**0.893**	0.885	0.877	0.890	0.815	**0.933**	0.923	0.932	0.932	0.921
Global PSDS	0.380	**0.386**	0.372	**0.386**	0.380	0.589	0.578	**0.600**	**0.600**	0.585
Macro F_1 (%)	44.97	**46.42**	45.13	**46.42**	45.84	44.97	**46.42**	45.13	**46.42**	45.84

Table 6: PSDS (scenarios 1 and 2) results of combined systems for each event category and overall PSDS and F_1 scores over the DESED Validation set. *3res*, *3res-T*, *4res* and *5res* are the multi-resolution combinations that were submitted to the challenge, whereas *5×BS* is a single-resolution combination of five models trained with the BS resolution point.

4. EXPERIMENTS AND RESULTS

Our experiments are based upon the 2021 baseline system[1] released by the DCASE Team. The only modification applied to the structure of the CRNN is the adaptation of the max-pooling layers of the convolutional stage to the number of mel-filters employed by each resolution point.

In the first place, we trained the baseline system using each one of the resolution points for feature extraction, leading to five single-resolution systems. Afterwards, following the method described in Section 3.2, several sets of resolution points were combined, obtaining multi-resolution systems.

PSDS scores are computed applying 50 different thresholds (linearly distributed from 0.01 to 0.99) to the combined score sequences, obtaining binary time series which are then smoothed by means of a median filter.

We report the results in terms of PSDS [11] and event-based, macro-averaged F_1-score [12]. In every case, scores are generated employing the Teacher models obtained from the Mean Teacher training.

To allow the evaluation of SED performance in different conditions, the challenge organization proposes two PSDS configurations. While the PSDS scenario 1 (PSDS 1) gives special importance to the precise temporal localization of events, the PSDS scenario 2 (PSDS 2) focuses on the correct detection of the event categories. The parameters that define these scenarios are described in Table 2.

The PSDS curves obtained with each of the feature resolution points described in 3.1 over the DESED Validation set, as well as their AUC (Area Under Curve) metrics, are shown in Figure 1. According to the results, it seems that a higher time resolution is beneficial for PSDS 1, while PSDS 2 is optimized using finer frequency resolutions. This behaviour was expected, taking into account that PSDS 1 is designed to focus on the temporal precision of the systems.

Aiming to include information from different resolution points in the SED system, networks trained with different feature resolutions have been combined as described in Section 3.2, obtaining the PSDS and macro F_1 results shown in Table 3. The model combinations include the Baseline resolution (BS) along with some of the resolution points we have proposed. Combining models trained with different feature resolutions outperforms the baseline and other single-resolution models in both PSDS scenarios, as well as in terms of F_1-score.

The best result for the first PSDS scenario over the Validation set is achieved by the *3res-T* and the *5res* combinations, both of them achieving an area under curve (AUC) of 0.386. On the other hand, the best results for the PSDS 2 scenario are obtained with *4res* and *5res*, both of them reaching AUCs of 0.600. Thus, although each scenario is optimized by combining either higher time resolutions or higher frequency resolutions, the fusion of the five resolution points (*5res*) seems to optimize both of them at the same time.

The *3res*, *3res-T*, *4res*, and *5res* combinations were submitted to the challenge, and their results are presented in Table 4. The best PSDS 1 over the 2021 Evaluation set is achieved by the 3res-T system (0.363), whereas the highest PSDS 2 is obtained by the 5res combination (0.577). Moreover, the performance of the submitted systems over the 2021 Evaluation set is very similar to that observed over the Validation set.

[1] https://github.com/DCASE-REPO/DESED_task

4.1. Class-wise results

In previous editions of the DCASE Challenge Task 4, SED systems were evaluated by means of the event-based macro F_1 score. Such metric is an average of the event-based F_1 scores for each target category, thus the scores for each individual class were usually highlighted in the results of the systems. On the other hand, whereas PSDS overcomes several limitations of the F_1 metric [13], the performance for each category is not usually described when reporting the results. For this reason, and considering that the detection of each event class is an independent task with an impact on the global results, we have computed the class-wise PSDS scores in terms of the Area Under Curve (AUC).

The class-wise PSDS results of the single-resolution systems are presented in Table 5. In each scenario, the best performing system in terms of global PSDS provides the largest AUC for several classes: in the first scenario, resolution T_+ holds the best results for *Cat*, *Dog* and *Frying*, whereas in the second scenario resolution F_{++} obtains the highest scores for *Alarm bell/ringing*, *Blender*, *Cat*, *Dishes* and *Electric shaver/toothbrush*. However, the rest of the event categories obtain better results with other resolutions, indicating that, as expected, the optimal resolution point depends not only on the PSDS settings but also on the characteristics of the target class.

Table 6 shows the PSDS results of combined systems. These systems include the multi-resolution fusions that have been submitted to the challenge (*3res*, *3res-T*, *4res*, and *5res*), as well as a combination of five different instances of the *BS* model (*5×BS*) which aims to contrast the performance of a single-resolution combination against the multi-resolution fusions. In most of the event categories, the largest AUC is obtained with a multi-resolution combination rather than with the single-resolution combination *5×BS*, being *Electric shaver/toothbrush* the exception in PSDS 1. Additionally, the *5×BS* achieves better global performance than the individual models. Therefore, it seems that the average fusion provides an improvement in both PSDS scenarios even when combining systems trained with the same resolution point. However, such improvement is larger when the systems to be combined have been trained with different resolutions.

5. CONCLUSIONS

In this work, we present the results of our participation in DCASE 2021 Challenge Task 4. Built upon the baseline provided by the organization, our proposed system combines different time-frequency resolution points of the mel-spectrogram features by averaging the output sequences of several CRNN detectors.

With the described approach, we have been able to outperform the baseline system in both PSDS scenarios and macro F_1 score over the DESED Validation and 2021 Evaluation sets. Moreover, the results indicate that certain resolutions and their combinations allow to optimize either the PSDS 1 (higher time resolutions) or PSDS 2 scenario (higher frequency resolutions), and that model fusion is more beneficial when different resolutions are combined.

Furthermore, the class-wise analysis of PSDS shows that the adequacy of each resolution point for sound event detection is related not only to the evaluation settings but also to the target category.

6. REFERENCES

[1] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[2] N. Turpault, R. Serizel, A. Parag Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, New York City, United States, October 2019. [Online]. Available: https://hal.inria.fr/hal-02160855

[3] R. Serizel, N. Turpault, A. Shah, and J. Salamon, "Sound event detection in synthetic domestic environments," in *ICASSP 2020 - 45th International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain, 2020. [Online]. Available: https://hal.inria.fr/hal-02355573

[4] D. de Benito-Gorrón, D. Ramos, and D. T. Toledano, "A multi-resolution approach to sound event detection in dcase 2020 task4," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 36–40.

[5] D. de Benito-Gorrón, D. Ramos, and D. T. Toledano, "An analysis of sound event detection under acoustic degradation using multi-resolution systems," in *Proc. IberSPEECH 2021*, 2021, pp. 36–40. [Online]. Available: http://dx.doi.org/10.21437/IberSPEECH.2021-8

[6] D. De Benito-Gorrón, D. Ramos, and D. T. Toledano, "A multi-resolution CRNN-based approach for semi-supervised sound event detection in DCASE 2020 challenge," *IEEE Access*, vol. 9, pp. 89 029–89 042, 2021.

[7] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[8] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in neural information processing systems*, 2017, pp. 1195–1204.

[9] N. Turpault and R. Serizel, "Training sound event detection on a heterogeneous dataset," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 200–204.

[10] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello, "Scaper: A library for soundscape synthesis and augmentation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 344–348.

[11] Bilen, G. Ferroni, F. Tuveri, J. Azcarreta, and S. Krstulović, "A framework for the robust evaluation of sound event detection," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 61–65.

[12] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, May 2016. [Online]. Available: http://dx.doi.org/10.3390/app6060162

[13] G. Ferroni, N. Turpault, J. Azcarreta, F. Tuveri, R. Serizel, Bilen, and S. Krstulović, "Improving sound event detection metrics: Insights from DCASE 2020," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 631–635.

FAIRNESS AND UNDERSPECIFICATION IN ACOUSTIC SCENE CLASSIFICATION: THE CASE FOR DISAGGREGATED EVALUATIONS

Andreas Triantafyllopoulos[1,2], *Manuel Milling*[2], *Konstantinos Drossos*[3], *Björn W. Schuller*[1,2,4]

[1]audEERING GmbH, Gilching, Germany
[2]EIHW – Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany
[3]Audio Research Group, Tampere University, Tampere, Finland
[4]GLAM – Group for Audio, Language, & Music, Imperial College, London, UK
atriant@audeering.com

ABSTRACT

Underspecification and *fairness* in machine learning (ML) applications have recently become two prominent issues in the ML community. Acoustic scene classification (ASC) applications have so far remained unaffected by this discussion, but are now becoming increasingly used in real-world systems where fairness and reliability are critical aspects. In this work, we argue for the need of a more holistic evaluation process for ASC models through *disaggregated evaluations*. This entails taking into account performance differences across several factors, such as city, location, and recording device. Although these factors play a well-understood role in the performance of ASC models, most works report single evaluation metrics taking into account all different strata of a particular dataset. We argue that metrics computed on specific sub-populations of the underlying data contain valuable information about the expected real-world behaviour of proposed systems, and their reporting could improve the transparency and trustability of such systems. We demonstrate the effectiveness of the proposed evaluation process in uncovering underspecification and fairness problems exhibited by several standard ML architectures when trained on two widely-used ASC datasets. Our evaluation shows that all examined architectures exhibit large biases across all factors taken into consideration, and in particular with respect to the recording location. Additionally, different architectures exhibit different biases even though they are trained with the same experimental configurations.

Index Terms— acoustic scene classification, evaluation, fairness, ethics, transparency

1. INTRODUCTION

Acoustic scene classification (ASC) has been established as a central task of artificial auditory intelligence, as exemplified by its prominent place in the DCASE challenge and workshop series [1, 2] and a generally broad accumulation of literature [3, 4, 5, 6, 7]. Overall, model performance has substantially improved through the years, and datasets have accordingly evolved to accommodate new challenges by incorporating factors shown to impact model performance. For example, the exact geographical location of the recordings was identified as an important factor early on, with datasets accordingly adapted by keeping data from the same location in the same partitions [1, 2]. The TUT Urban Acoustic Scenes 2018 Mobile dataset additionally introduced the recording device as a separate factor [8], with the development set consisting of multiple recording devices, and the evaluation set including an extra, unseen device. Finally, the TAU Urban Acoustic Scenes 2019 dataset highlighted the importance that the city of origin has by introducing data from two additional cities in the evaluation set [8].

In general, the community is aware of the influence that recording devices and location have on model performance [9, 10]. Most works approach these factors from the perspective of *domain mismatch* [11]: different cities, locations, and devices, result in slightly different input representations, and the difference needs to be accounted for to improve overall performance. Several approaches have been proposed to mitigate the problem, largely drawing from the wide literature of domain adaptation techniques [11] adapted for the ASC problem [12, 13, 14], or specifically taking steps to mitigate the effects of city and device [15, 16].

In this work, we adopt a different perspective: we argue that those factors deserve a prominent place in the evaluation of ASC systems as they reveal important insights about the behaviour of trained models. To do that, we adopt the language of recent works in the machine learning (ML) *fairness* literature. In particular, we propose *disaggregated evaluations*, a concept highlighted by Mitchell *et al.* [17] as a means to expose the effects that these underlying factors have on system performance. Disaggregation, which corresponds to breaking down an evaluation to more fine-grained levels of analysis, can be done both in a *unitary* (how performance is affected by each factor independently) and in an *intersectional* way (how performance is affected by combination of factors). For the task of ASC, we consider the three aforementioned factors, namely location, city, and device, as warranting a closer investigation. This choice is primarily motivated by availability (the existing metadata is already there) and community awareness (past works take them into account).

The rest of this document is organised as follows. In Section 2, we formulate our research question by discussing fairness and underspecification for ASC. Our methodological approach, including a description of the data and deep learning (DL) architectures used in our experiments, is outlined in Section 3. The results and a discussion of our disaggregated evaluations are presented in Section 4. Finally, we summarise our findings in Section 5.

2. FAIRNESS AND UNDERSPECIFICATION IN ASC

The success and increased usage of ML, and in particular DL, systems in commercial applications has led to rising concerns towards discriminating biases exhibited by ML applications, for instance

based on race [18]. Especially in the case of DL, a lack of interpretability can often be observed [19], thus posing additional challenges to discover and mitigate said biases. Even though ASC models are not widely considered high risk applications, their increasing usage in smart city [20], security [21], elderly monitoring [22], and autonomous driving [23] applications means they may soon (or already) be part of critical decision making systems, thus making fairness a critical consideration for those algorithms.

Of the three factors, the recording device is perhaps the most benign; it is hard to justify why an ASC system that only works for specific devices should raise ethics concerns, although low-income groups could be excluded if data are only collected with high-end equipment. On the other hand, city and location (which corresponds e. g. to specific neighbourhoods) pose potentially bigger problems; a security application should work *equally well* for all citizens irrespective of where they reside, and autonomous driving systems should maintain a standard of performance irrespective of where the vehicle currently is. There is a already a rich body of work in social sciences discussing inequality across different neighbourhoods on income, health, and other socioeconomic factors [24], which an unreliable system may inadvertently exacerbate. This could have adverse effects against people living in those neighbourhoods, and may disproportionately affect minorities in demographically segregated communities. Therefore, we anticipate that explicitly communicating disaggregated performance with respect to all three factors would enhance trustability in ASC systems used in real-life environmental sensing applications.

Disaggregated evaluations can also be viewed under the perspective of recent research on the *underspecification* of ML architectures [25], which corresponds to the fact that several architectures yielding similar in-domain performance nevertheless exhibit different behaviour during system deployment. This undesired property may have negative consequences on the reliability and trustability of ASC systems. For example, if a person using an ASC system observed substantially different performance when visiting different neighbourhoods of the same city, they might eventually lose their trust in system performance and stop using it. As ASC architectures increasingly find their way into more real-life applications, the need to address this issue becomes more pressing. Our evaluation reveals that different architectures yielding almost equivalent performance in standard aggregated evaluations exhibit different behaviour across different sub-populations of the herein examined datasets, thus illustrating that underspecification is also a problem for ASC applications. This shows that disaggregated evaluations can be a useful tool for practitioners that need to select among a pool of candidate models.

3. METHODOLOGICAL APPROACH

Our approach consists of the following steps. First, we train several deep neural network (DNN) models on the training set of each of the datasets examined here. Each model is trained for 60 epochs using stochastic gradient descent (SGD) with a Nesterov momentum of 0.9, a learning rate of 0.001, and a batch size of 64. For all experiments, we use log Mel spectrograms with 64-bins as input features, extracted with a window size of 32 ms and a hop size of 10 ms. These hyper-parameters were fixed a priori for all models and not optimised during our experiments. Each model is trained with 5 random seeds to mitigate the effect of randomness.

Our experiments are conducted on the TUT Urban Acoustic Scenes 2018 and TUT Urban Acoustic Scenes 2018 Mobile data sets [8], which will be henceforth referred to as TUT-Urban and TUT-Mobile for brevity. Both datasets contain data from 10 acoustic scenes recorded across several locations of 6 different European cities. TUT-Urban contains 8640 stereo samples recorded at 48 kHz with a single high-quality recording device (Soundman OKM II Klassik/studio A3), whereas TUT-Mobile additionally contains 720 samples from each of two additional low-quality recording devices (Samsung Galaxy S7 and iPhone SE). In the case of TUT-Mobile all data are stored as mono recordings at 16 kHz.

All models are first evaluated in the standard, aggregated way by computing a single accuracy value, and subsequently assessed using unitary and intersectional evaluations as described below. We begin with unitary evaluations, where each factor is considered in isolation. For city and device, where we have only 6 and 3 different groups, respectively, we simply report the accuracy for each group. The location factor is more complicated, as we have 83 different locations in the test set, thus making it hard to visualise results. Moreover, whereas for each city and device we have all classes available, each location corresponds to exactly one class, thus making accuracy an inappropriate metric for that evaluation. To overcome these problems, we compute the F_1 score, which is the harmonic mean of precision and recall, for the class corresponding to each location and further normalise the per location F_1 score, F_1^l, by the overall F_1 score for that architecture.

Intersectional evaluations are in turn conducted by taking into account two, or more, factors. Due to space limitations, we only consider results for two pairs of factors: the variation of cities across different devices and the variation across locations in different cities. For the first case, we report the accuracy for each combination of factors. For the latter case, we compute the F_1^l score for each location as in the unitary case, but now normalise over the F_1 score for each city, F_1^c.

As DL architectures, we use 5 standard DNN models that belong to different architecture families. **FFNN**: as the most simple architecture we choose a feed-forward neural network with three hidden layers of decreasing sizes, 300, 200, and 100 units with a rectified linear unit (ReLU) activation function. The inputs of of the network are the flattened log Mel spectrograms. **TDNN**: we further employ a time-delay neural network (TDNN) architecture. First introduced by [26] with the aim of learning temporal relationships, TDNNs have recently seen great success in the field of speaker identification [27]. Our TDNN architecture is identical to the DNN architecture described as the *x-vector system* in [27]. **CNN6**, **CNN10**, **CNN14**: the final architectures considered in our experiments are CNN-based and were recently introduced by Kong *et al.* [28] in the context of audio pattern recognition. The three architectures have a total of 6, 10, and 14 layers, respectively, excluding pooling layers after convolutional layers, and take Mel-spetrograms as inputs. The final two layers of each network are fully connected.

4. RESULTS AND DISCUSSION

Our unitary evaluation results for different cities are presented in Table 1, along with the standard aggregated metrics. We show model accuracy for each factor in isolation, and also report the standard deviation over all factors. F_1 results for different locations in TUT-Urban are shown in Figure 1, where we show box-and-whisker plots of the normalised F_1 scores. We omit unitary results for different devices as they can be inferred from the intersectional results in Table 2; as expected, all architectures perform best on the high-quality device A, for which we also have the most data, while doing worse

Table 1: Aggregated and unitary disaggregated evaluations considering different cities in isolation. For the aggregated evaluation, we show accuracy[%] for all test data for TUT-Urban and TUT-Mobile. For the unitary disaggregated evaluations, we show accuracy[%] on different cities for each architecture, as well as its standard deviation (σ) over the different cities. Results are averaged across 5 different runs.

	TUT-Urban					**TUT-Mobile**				
	FFNN	**TDNN**	**CNN6**	**CNN10**	**CNN14**	**FFNN**	**TDNN**	**CNN6**	**CNN10**	**CNN14**
Aggregated	52.8	57.2	**68.7**	67.7	66.3	53.1	54.7	**66.8**	66.3	63.6
City	**Disaggregated evaluations**									
Barcelona	52.9	61.7	**64.8**	60.9	58.9	55.9	56.4	**61.3**	57.9	57.6
Helsinki	56.1	61.3	**70.2**	67.3	63.5	50.8	57.1	**67.9**	66.7	58.3
London	51.1	61.7	71.6	**74.2**	71.5	52.7	59.2	70.1	**72.0**	70.8
Paris	45.5	53.8	62.0	61.0	**62.4**	45.8	54.1	60.1	59.7	**60.8**
Stockholm	53.0	47.4	**73.1**	68.4	68.2	56.2	46.9	**72.5**	68.3	67.3
Vienna	59.8	57.9	69.9	**74.0**	73.0	58.8	54.3	68.0	**72.4**	65.7
σ	4.4	5.2	**3.9**	5.4	5.1	4.3	**3.9**	4.5	5.6	4.9

Table 2: Intersectional evaluations considering recording device and city in combination for the TUT-Mobile dataset. We show accuracy[%] for each combination of city and device. Cites are Barcelona (B), Helsinki (H), London (L), Paris (P), Stockholm (S), and Vienna (V). The best performing architecture value per city and device is marked by boldface. Results are averaged across 5 different runs.

	Device A							**Device B**							**Device C**						
Model	**B**	**H**	**L**	**P**	**S**	**V**	σ	**B**	**H**	**L**	**P**	**S**	**V**	σ	**B**	**H**	**L**	**P**	**S**	**V**	σ
FFNN	56.1	53.1	53.0	46.9	56.5	60.5	**4.2**	54.8	38.1	54.8	29.0	56.7	48.7	10.2	54.1	31.0	45.2	46.5	52.0	48.7	7.5
TDNN	57.9	60.4	62.9	57.0	47.5	56.3	4.8	44.4	38.1	26.5	34.8	53.3	41.3	8.3	46.7	30.3	34.2	32.9	31.3	43.3	**6.2**
CNN6	**61.9**	**70.2**	71.2	**63.1**	**73.4**	69.7	**4.2**	**55.6**	**58.7**	**67.7**	38.1	65.3	**61.3**	9.7	**57.8**	44.5	56.8	39.4	**66.0**	54.7	8.8
CNN10	58.8	68.9	**73.7**	62.2	69.1	**74.2**	5.6	53.3	56.1	64.5	41.9	**68.0**	**61.3**	8.5	49.6	**46.5**	53.5	42.6	56.7	**62.7**	6.6
CNN14	58.7	60.5	71.9	62.4	68.5	67.2	4.7	48.1	47.1	64.5	**43.2**	61.3	58.0	**7.9**	51.9	38.1	**60.0**	**55.5**	55.3	56.0	7.0

on the lower quality and less populous B and C devices. Location results on TUT-Mobile are also omitted due to space limitations but exhibit the same trend as those on TUT-Urban.

Table 1 can be read both horizontally, thus emphasising which model works best for a specific factor, and vertically, where we are interested in how a specific model performs across different factors. Overall, CNN6 is showing the strongest performance, followed by CNN10 and CNN14, with TDNN and FFNN performing substantially worse. Furthermore, CNN6 exhibits relative stability across both cities and devices. However, it is not the best choice for all cities; in both datasets, CNN10 is outperforming it for London and Vienna, and CNN14 for Paris, though the latter only marginally.

Of more interest is the vertical interpretation of Table 1. We observe that different architectures exhibit a different ordering when it comes to performance per city. In TUT-Urban for example, different architectures yield their best performance on different cities: FFNN on Vienna, TDNN on Barcelona and London, CNN6 on Stockholm, CNN10 on London and Vienna, and CNN14 on Vienna. Another interesting case is Stockholm, where CNN6 shows its best performance and TDNN its worst. Conversely, Vienna, where FFNN, CNN10, and CNN14 show (near-)best performance for TUT-Urban, is showing mediocre results for CNN6 and TDNN.

For TUT-Mobile, these results are better visualised in Figure 3 which shows the range of F1 scores per location for the different cities. Notable differences exist; TDNN shows worse performance on Stockholm than Paris, whereas all other architectures show the opposite trend. CNN6 and CNN10, which are almost equivalent in terms of aggregated performance, also exhibit differences, in particular for Stockholm and Vienna. Interestingly, TDNN and FFNN deviate substantially from the other three architectures, which are more closely clustered together, indicating that models from the same family exhibit more similar behaviour. These observations illustrate that the inductive biases introduced by each architecture manifest themselves as different behaviours on different strata of each dataset, which is in line with recent research on inductive biases [29, 30].

Figure 1 additionally shows that location is a very important factor when it comes to system performance, with some locations

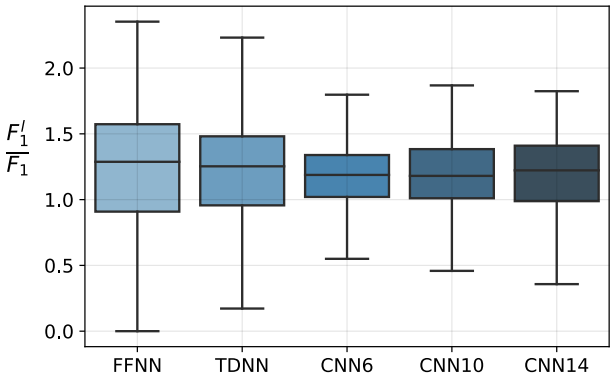


Figure 1: Distribution of relative F_1 score on different locations of TUT-Urban for all architectures. Box plots show median and inter-quartile range of relative F_1 score.

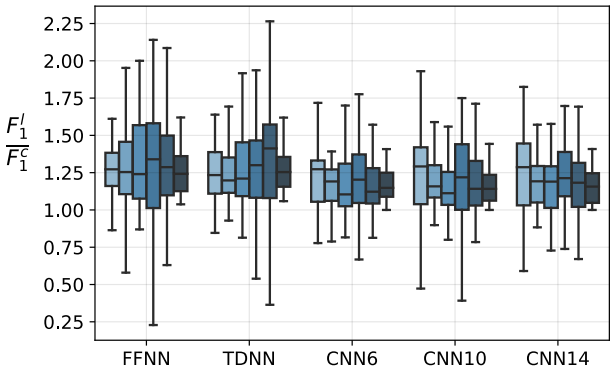


Figure 2: Intersectional analysis of model performance with respect to both city and location for TUT-Urban. For each architecture, cities are (from left to right): Barcelona, Helsinki, London, Paris, Stockholm, Vienna. Box plots show median and inter-quartile range of relative F_1 score with respect to different locations within each city.

exhibiting almost half the aggregated system performance. Such behaviour is highly undesirable because an ASC system deployed across different locations will consistently exhibit subpar performance for some of them, with the risk to equal and fair access to service that this entails. We note that most locations seem to exhibit better than average performance (the F_1 ratio is bigger than 1). This is caused by the fact that the worst performing locations happen to have more samples, thus having a bigger influence on aggregate performance.

Intersectional results are shown in Table 2 for the combination of city and device, and in Figure 2 for the combination of city and location. The differences amongst all cities and all devices were found significant for all architectures using Kruskal-Wallis omnibus H-tests for each factor and architecture, respectively. This shows that, in general, both factors have a large effect on model performance. In addition, Table 2 and Figure 2 both show that different architectures exhibit different behaviour on different strata of the two datasets, even though they were trained on identical settings. Overall, CNN6 is again showing the strongest performance for most, though not all, combinations, followed by CNN10. In terms of individual factors, Paris is showing the biggest drop in performance when switching from device A to device B for all architectures, indicating that the domain shift introduced by different devices is more adversely impacting this city.

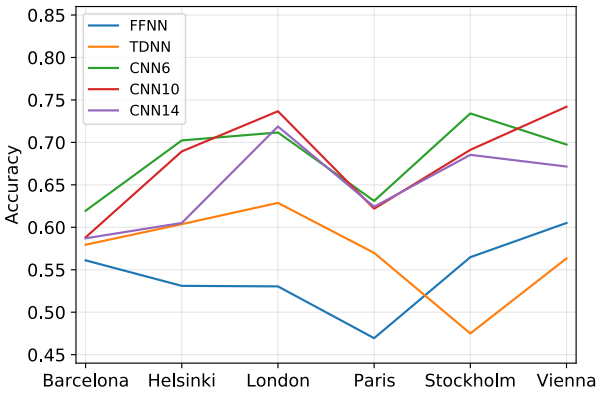


Figure 3: Accuracy for each city and architecture on TUT-Mobile.

The most interesting case is TDNN, which is showing its best and worst performance on London and Stockholm for device A, respectively, but shows the exact opposite for device B, where the best performance is obtained for Stockholm and the worst for London. In fact, the performance of TDNN on Stockholm is far better for device B than for device A, even though the latter has far more samples and should thus lead to better performance.

5. CONCLUSION

In this work, we argue for the need of disaggregated unitary and intersectional evaluations for the task of ASC. Our proposed evaluation methodology reveals that several baseline architectures exhibit different behaviour even though they are trained with similar settings. This illustrates that ASC models trained on the examined datasets suffer from the underspecification problem, which heavily impacts the development of reliable and trustworthy systems. In the future, we intend to further investigate this problem under the perspective of inductive biases introduced by each architecture [30].

Moreover, our work raises interesting questions on the fairness of ASC applications. The architectures examined here exhibit a bias with respect to different cities, locations, and devices. If these architectures were deployed in a real-world setting, this would translate to non-uniform behaviour over these different factors. This poses a risk to fair and equitable use of ML resources. We believe this important point needs to be addressed as ASC models are being increasingly integrated in intelligent decision making systems.

6. ACKNOWLEDGMENT

Part of the work leading to this publication has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 957337, project MARVEL.

7. References

[1] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, Nov. 2017, pp. 85–92.

[2] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 379–393, Feb. 2018, ISSN: 2329-9290. DOI: 10.1109/TASLP.2017.2778423.

[3] Z. Liu, Y. Wang, and T. Chen, "Audio feature extraction and analysis for scene segmentation and classification," *Journal of VLSI Signal Processing*, vol. 20, Apr. 1998. DOI: 10.1023/A:1008066223044.

[4] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.

[5] A. Rakotomamonjy, "Supervised representation learning for audio scene classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1253–1265, 2017. DOI: 10.1109/TASLP.2017.2690561.

[6] K. Qian, Z. Ren, V. Pandit, Z. Yang, Z. Zhang, and B. Schuller, "Wavelets revisited for the classification of acoustic scenes," Nov. 2017.

[7] Z. Ren, K. Qian, Z. Zhang, V. Pandit, A. Baird, and B. Schuller, "Deep scalogram representations for acoustic scene classification," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 3, pp. 662–669, 2018.

[8] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, Nov. 2018, pp. 9–13. [Online]. Available: https://arxiv.org/abs/1807.09840.

[9] ——, *Acoustic scene classification in dcase 2019 challenge: Closed and open set classification and data mismatch setups*, New York University, NY, USA, Oct. 2019.

[10] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in dcase 2020 challenge: Generalization across devices and low complexity solutions," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, 2020, pp. 56–60.

[11] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira, *et al.*, "Analysis of representations for domain adaptation," *Advances in neural information processing systems*, vol. 19, p. 137, 2007.

[12] S. Gharib, K. Drossos, E. Cakir, D. Serdyuk, and T. Virtanen, "Unsupervised adversarial domain adaptation for acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, Nov. 2018, pp. 138–142.

[13] K. Drossos, P. Magron, and T. Virtanen, "Unsupervised adversarial domain adaptation based on the wasserstein distance for acoustic scene classification," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019, pp. 259–263. DOI: 10.1109/WASPAA.2019.8937231.

[14] Z. Ren, Q. Kong, J. Han, M. D. Plumbley, and B. W. Schuller, "Caanet: Conditional atrous cnns with attention for explainable device-robust acoustic scene classification," *IEEE Transactions on Multimedia*, 2020.

[15] H. Chen, Z. Liu, Z. Liu, P. Zhang, and Y. Yan, "Integrating the data augmentation scheme with various classifiers for acoustic scene modeling," DCASE2019 Challenge, Tech. Rep., Jun. 2019.

[16] M. Kośmider, "Calibrating neural networks for secondary recording devices," DCASE2019 Challenge, Tech. Rep., Jun. 2019.

[17] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," in *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 220–229.

[18] M. Wang, W. Deng, J. Hu, X. Tao, and Y. Huang, "Racial faces in the wild: Reducing racial bias by information maximization adaptation network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[19] N. Burkart and M. Huber, "A survey on the explainability of supervised machine learning," *Journal of Artificial Intelligence Research*, vol. 70, Jan. 2021. DOI: 10.1613/jair.1.12228.

[20] J. P. Bello, C. Mydlarz, and J. Salamon, "Sound analysis in smart cities," in *Computational Analysis of Sound Scenes and Events*, Springer, 2018, pp. 373–397.

[21] R. Radhakrishnan, A. Divakaran, and A. Smaragdis, "Audio analysis for surveillance applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.*, IEEE, 2005, pp. 158–161.

[22] R. Mégret, V. Dovgalecs, H. Wannous, S. Karaman, J. Benois-Pineau, E. El Khoury, J. Pinquier, P. Joly, R. André-Obrecht, Y. Gaëstel, *et al.*, "The immed project: Wearable video monitoring of people with age dementia," in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 1299–1302.

[23] M. K. Nandwana and T. Hasan, "Towards smart-cars that can listen: Abnormal acoustic event detection on the road.," in *Interspeech*, 2016, pp. 2968–2971.

[24] M. Wen, C. R. Browning, and K. A. Cagney, "Poverty, affluence, and income inequality: Neighborhood economic structure and its implications for health," *Social science & medicine*, vol. 57, no. 5, pp. 843–860, 2003.

[25] A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, *et al.*, "Underspecification presents challenges for credibility in modern machine learning," *arXiv preprint arXiv:2011.03395*, 2020.

[26] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[27] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 5329–5333.

[28] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[29] G. Ortiz-Jimenez, A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "Neural anisotropy directions," *arXiv preprint arXiv:2006.09717*, 2020.

[30] G. Ortiz-Jimenez, I. F. Salazar-Reque, A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "A neural anisotropic view of underspecification in deep learning," *arXiv preprint arXiv:2104.14372*, 2021.

SEMI-SUPERVISED SOUND EVENT DETECTION USING MULTISCALE CHANNEL ATTENTION AND MULTIPLE CONSISTENCY TRAINING

Yih-Wen Wang[1], *Chia-Ping Chen*[1], *Chung-Li Lu*[2], *Bo-Cheng Chan*[2]

[1]National Sun Yat-Sen University, Taiwan, m083040011@student.nsysu.edu.tw, cpchen@cse.nsysu.edu.tw
[2]Chunghwa Telecom Laboratories, Taiwan, {chungli,cbc}@cht.com.tw

ABSTRACT

We present a neural network-based sound event detection system that outputs sound events and their time boundaries in audio signals. The network can be trained efficiently with an amount of strongly labeled synthetic data and weakly labeled or unlabeled real data. Based on the mean-teacher framework of semi-supervised learning with RNNs and Transformer, the proposed system employs multi-scale CNNs with efficient channel attention, which can capture the various features and pay more attention to the important area of features. The model parameters are learned with multiple consistency criteria, including interpolation consistency, shift consistency, and clip-level consistency, to improve the generalization and representation power. For different evaluation scenarios, we explore different pooling functions and search for the best layer. To further improve the performance, we use data augmentation and posterior-level score fusion. We demonstrate the performance of our proposed method through experimental evaluation using the DCASE2021 Task4 dataset. On the validation set, our ensemble system achieves the PSDS-scenario1 of 40.72% and PSDS-scenario2 of 80.80%, significantly outperforming that of the baseline score of 34.2% and 52.7%, respectively. On the DCASE2021 challenge's evaluation set, our ensemble system is ranking 7 among the 28 teams and ranking 14 among the 80 submissions.

Index Terms— sound event detection, Transformer, channel attention, semi-supervised learning, consistency training

1. INTRODUCTION

Sound event detection (SED) is a useful technique for helping us what is happening in an environment by identifying sounds [1, 2, 3]. SED predicts not only the sound event types in an audio recording but also the corresponding onset and offset times. Recently, Detection and Classification of Acoustic Scenes and Events (DCASE) promotes researches on sound detection and classification by annual workshops and challenges. To learn less from human annotation and more from data, DCASE 2021 Task 4 [4] proposes semi-supervised learning to explore the possibility of learning SED with the data of strongly labeled, weakly labeled, and unlabeled. Furthermore, DCASE proposed two evaluation metrics: PSDS-scenario 1 (PSDS 1) requires that SED system needs to react fast upon an event detection; PSDS-scenario 2 (PSDS 2) requires that SED system must avoid confusion between classes but the reaction time is less crucial than in the previous scenario.

One well-known semi-supervised learning approach is to train CRNN [5] with the mean-teacher framework [6]. CRNN utilizes CNNs to extract the short-term and local information and RNNs to capture the long-term contextual information. The mean-teacher framework exploits consistency regularization to stabilize the classifier output for unlabeled data or weakly-labeled data. Besides, the transformer architecture [7] can extract global information while reducing the high computational cost of RNN and achieve state-of-the-art performance on multiple tasks, such as speech recognition [8], speaker recognition [9], speaker diarization [10], text-to-speech [11], audio tagging [12], and sound event detection [13].

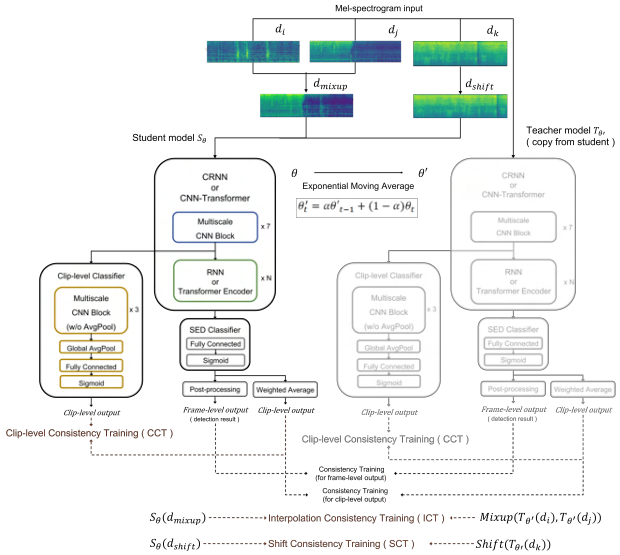


Figure 1: Overview of our proposed system. With the multi-scale CNNs and ECA-net based on RNNs/Transformer network, learning of the mean-teacher framework is enhanced with multiple objectives. ICT/SCT encourages the prediction of interpolated/time-shifted data to be consistent with the interpolated/time-shifted prediction. CCT encourages the origin output consistent with the clip-level classifier output. d_i, d_j, d_k: the original data points; $d_{\texttt{mixup}}$: the mixture of d_i and d_j; $d_{\texttt{shift}}$: time-shift of d_k; $S_\theta, T_{\theta'}$: the student and teacher model.

In this paper, we first explore the performance of RNNs-based and Transformer-based neural networks for two evaluation metrics, PSDS 1 and PSDS 2. Then, since the length of sound events is very different so that we apply the multi-scale CNNs [14] with efficient channel attention (ECA-Net) [15] to capture the more various and important features. Meanwhile, we extend the consistency criteria for model training in mean-teacher framework to include interpolation consistency (ICT) [16], shift consistency (SCT) [17], and clip-level consistency (CCT) [18]. In addition, we apply data augmen-

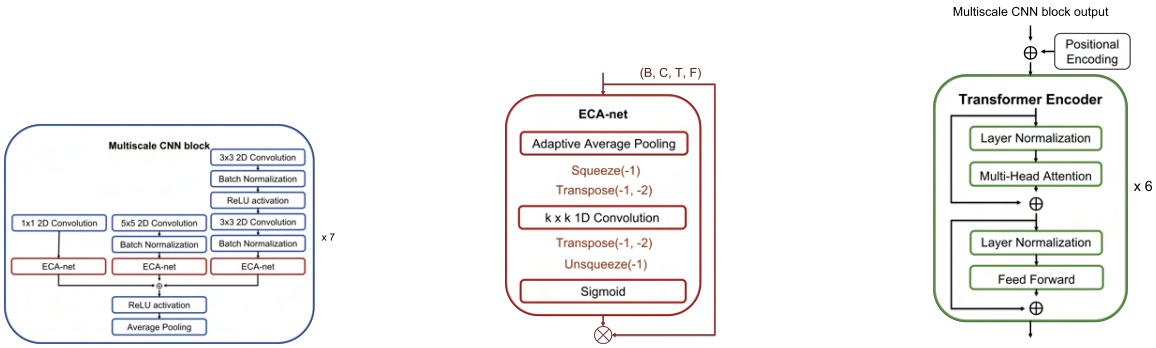


Figure 2: From left to right, the network architecture of multi-scale CNN block, efficient channel attention network (ECA-Net), and Transformer encoder block.

tation and posterior-level score fusion to further improve the model performance. Finally, on the validation set and public evaluation set of DCASE 2021 Task4, our proposed system both outperform the baseline system.

The rest of the paper is organized as follows. In Section 2, we introduce the proposed network architecture, multiple consistency schemes, data augmentation, and posterior-level score fusion to improve the SED system. Section 3 describes the dataset, audio preprocessing, and training setups. Section 4 presents the experimental results and analysis. Finally, we draw conclusion in Section 5.

2. PROPOSED METHODS

2.1. Network architecture

2.1.1. Multi-scale CRNN / CNN-Transformer

From strongly labeled training data, we estimate duration of each sound event as below. 0∼2s: `alarm/bell/ringing`, `cat`, `dishes`, `dog`, and `speech`. 4∼6s: `blender` and `running water`. 7∼10s: `electric shaver/toothbrush`, `frying`, and `vacuum cleaner`. The length of sound events is various and cause the model to work with inconsistent accuracy for the event of different scales. Thus, we refer to [14] to build a multi-scale CNN block to capture the richer features, which contains the kernel size of 1x1, 3x3, 5x5 and uses addition to integrate features of different scales, as shown in the left of Figure 2. In 7 layers of multi-scale CNN block, we also utilize batch normalization and ReLU activation to speed up and stabilize training, each of which attaches an average-pooling layer to calculate the average for each patch of the feature map and downsample feature dimensions along both the time axis and the frequency axis.

To obtain the long-term contextual information, we use the RNNs and transformer encoder to form CRNN [19, 5] and CNN-Transformer [20, 13]. RNNs are applied to two layers of bi-directional gated recurrent unit (GRU) like DCASE 2021 baseline. The network architecture of the transformer encoder is as shown in the right of Figure 2. Positional encoding is used to enhance the output features from the multi-scale CNN blocks with order information before the transformer blocks. A transformer encoder block has layer normalization, multi-head attention, and feed-forward layer. The multi-head attention estimates the similarity between query and key and extracts value as a weighted sum. The mechanism allows the model to jointly pay attention to the information from different positions. The fully connected feed-forward layer with ReLU activation is applied to each position identically. For regularization, we adopt pre-layer normalization (Pre-LN) [21] and residual connection. Finally, the SED classifier consists of a fully connected layer and sigmoid function to discriminate the sound event types.

2.1.2. Efficient Channel Attention

The effect of the acoustic feature extraction largely determines the model ability to predict different sound events and affects the final classification result. However, the attention mechanism can make the model pay more attention to areas which may be important features, and improve the model ability to distinguish features of sound events. We combine the efficient channel attention network (ECA-Net) [15] in multi-scale CNN blocks before adding features of different scales, as shown in the left of Figure 2. ECA-Net is composed of adaptive average pooling (A-AvgPool) layer, 1D convolutional (1D-CNN) layer, and sigmoid function, as shown in the middle of Figure 2. A-Avgpool is applied along the time axis and 1D-CNN calculate the attention of each channel. The kernel size of 1D-CNN is defined by

$$k = \left| \frac{\log_2(C) + b}{\gamma} \right|_{odd} \quad (1)$$

where k and C denote kernel size and channel dimensional, γ and b are set to 2. Clearly, high-dimensional channels have longer range interaction, vice versa.

2.1.3. Pooling Function

Wang et al. [22] compared five different types of pooling functions in the multiple instance learning (MIL) framework for SED, namely attention pooling, max pooling, average pooling, linear softmax, and exponential softmax. The formula of each pooling function is presented in Table 2. The attention pooling estimates the weights for each frame are learned with a dense layer in the network. The max pooling simply take the large probability in all frames. The average pooling assigns an equal weight for all frames. The linear softmax assigns weights equal to the frame-level probability, while the exponential softmax assigns a weight of exponential to the frame-level probability. DCASE 2021 Task4 baseline [5] uses attention pooling to transform frame-level into clip-level. However, with different evaluation scenarios, there should be a relatively appropriate pooling function to replace.

2.2. Semi-Supervised Learning

We employ the mean-teacher framework for its fast convergence, instead of the Π model [23] or temporal ensembling [24], exploiting consistency regularization to stabilize the classifier output for unlabeled data or weakly-labeled data. In this work, we use Mean Square Error (MSE) loss for the consistency cost:

$$\text{MSE}(y, \hat{y}) = (y - \hat{y})^2, \tag{2}$$

where y and $\hat{y}$ denote the target and the prediction, respectively. Next, we propose multiple consistency criteria to regularize how the SED system should learn from unlabeled or weakly-labeled data.

2.2.1. Interpolation Consistency Training

The interpolation consistency training (ICT) [16] has been proposed for semi-supervised learning. ICT encourages the prediction at an interpolation of unlabeled data points to be consistent with the interpolation of the prediction at these data points. Learning from interpolation samples can help the model discriminate ambiguous samples to improve the generalization ability. We define the ICT loss function by

$$\begin{aligned} L_{ICT} = \text{MSE}(&S_\theta(\lambda d_i + (1-\lambda) d_j), \\ &\lambda T_{\theta'}(d_i) + (1-\lambda) T_{\theta'}(d_j)), \end{aligned} \tag{3}$$

where S_θ and $T_{\theta'}$ denote a student model and a teacher model, d_i and d_j denote data points, and λ is randomly sampled from a Beta distribution.

2.2.2. Shift Consistency Training

Inspired by ICT, we consider time-shift as another way to enhance consistency which is similar to proposed by [17], called shift consistency training (SCT). We define the SCT loss function by

$$L_{SCT} = \text{MSE}(S_\theta(\text{shift}(d_k)), \text{shift}(T_{\theta'}(d_k))). \tag{4}$$

SCT encourages the prediction of time-shift input to be consistent with time-shift prediction. In theory, it allows the model to learn shift-invariance and temporal localization of sound events.

2.2.3. Clip-level Consistency Training

In addition to ICT and SCT, we also apply clip-level consistency training (CCT) [18] to enhance the ability to extract the features. We define the CCT loss function by

$$L_{CCT} = \text{MSE}(\text{NN}(d_x), \text{ClipLevel}(f_x)), \tag{5}$$

where $\text{NN}(d_x)$ is the weighted average pooling of the multi-scale CRNN or CNN-Transformer frame-level network output of data d_x, and $\text{ClipLevel}(f_x)$ is obtained by feeding the feature map f_x of the final multi-scale CNN block to a clip-level classifier. As shown in Figure 1, the clip-level classifier consists of 3 extra multi-scale CNN blocks, a global average pooling, and a fully connected layer.

2.2.4. Overall Consistency Training

In summary, the overall loss is

$$L = L_0 + L_{ICT} + L_{SCT} + L_{CCT}, \tag{6}$$

where L_0 denotes the loss without the proposed consistency, namely mean square error for original consistency cost and binary cross-entropy for the supervised cost.

2.3. Data Augmentation

- Mixup [25]. It mixes two randomly selected samples from the original training data and uses λ sampled from Beta distribution to control the strength of interpolation between two samples. The linear interpolation technique can enhance the data diversity and robustness of the network.
- Shift [26]. It shifts a feature sequence on the time axis, and overrun frames are concatenated with the opposite side of the sequence. The usage helps the network learn temporal localization information of the sound event.
- Masks [26]. It creates artificial data by masking a block of consecutive time steps or frequency channels on the mel-spectrogram instead of the raw audio. It can help the network learn the beneficial features to be robust to the partial loss of spectral information or speech segments.

2.4. Posterior-level Score Fusion

To improve generalization performance, we perform score fusion as a model ensemble technique. We utilize different data augmentation methods to build several single systems based on multi-scale CRNN and CNN-Transformer models with different schemes. Then, we average the raw posterior outputs $p(X)$ for inputs X of the multiple models:

$$p_{\text{fusion}}(X) = \frac{1}{N} \sum_{n=1}^{N} p_n(X), \tag{7}$$

where N means the total number of models for our fusion.

3. EXPERIMENTS

3.1. Dataset and Signal Preprocessing

The DESED dataset of DCASE 2021 Task 4 is comprised of 10-sec audio clips and 10 classes of sound events. The data are in two domains: real data (44.1kHz) extracted from AudioSet [27] and synthetic data (16kHz) generated by Scaper [28]. Each audio clip can be strongly labeled with the sound events and their time boundaries annotated, weakly labeled with only the sound events annotated, or unlabeled without any annotation. All dataset is divided into 5 subsets: weakly labeled (1,578 clips), unlabeled (14,412 clips), strongly labeled (10,000 clips), validation set (1,168 clips), public evaluation set (692 clips). Audio signals are resampled to 16kHz sampling rate at first by FFmpeg tool [29]. Then, 128-channel mel-spectrogram from them is extracted with a window size of 2048 and hop size of 256 by Librosa tool [30]. Consequently, the size of the input acoustic features to the deep neural network is 626×128.

3.2. Network Setups

The 7 layers of multi-scale CNN blocks have the number of filters:[16, 32, 64, 128, 128, 128, 128] and pooling size:[[2, 2], [2, 2], [1, 2], [1, 2], [1, 2], [1, 2], [1, 2]]. The 6 layers of transformer encoder blocks have multi-head attention with 256 units and 8 heads and a feed-forward layer with 2048 units. For ICT and mixup augmentation, the parameter λ is sampled from $\text{Beta}(\alpha, \alpha)$ and α from 0.1 to 0.9 in increments of 0.1. For SCT and shift augmentation, we choose the amount of time-shift by sampling from a normal distribution with a zero mean and a standard deviation of 90. For masks augmentation, the size of time-mask and frequency-mask are sampled from a uniform distribution from 0 to 30 and 40, respectively.

4. EVALUATION RESULTS

The evaluation of DCASE 2021 Task4 contains PSDS 1 (react fast) and PSDS 2 (avoid class confusion). From Table 1, we can find that the results of RNNs-based network is better than Transformer-based one, especially on PSDS 2. Then, whatever neural network is CRNN or CNN-Transformer, the incorporation of ICT, SCT, and CCT has significant achievement on two scenarios. The multiple consistency training schemes on CRNN improved PSDS 1 from 34.04% to 37.86%, PSDS 2 from 53.30% to 60.87%, and on CNN-Transformer, PSDS 1 from 33.46% to 37.33%, PSDS 2 from 48.77% to 55.87%. In addition, we observe that multi-scale CNN blocks and ECA-Net can help the model obtain various and important features of sound events so that CRNN can reach 65.54% and CNN-Transformer can reach 61.10% for PSDS 2. From Table 2, both types of neural networks are best when using attention pooling at PSDS 1 and using exponential softmax at PSDS 2. We consider that attention pooling learns weights from the network so that they have a time series relationship. Therefore, it has better performance under stricter evaluation standards with time requirements. Then, exponential softmax uses exponentials as weights to conform to monotonicity so that the higher the prediction probability of the time point, the higher the weight. Thus, it has better performance under the stricter evaluation criteria with category requirements.

We combine CRNN/CNN-Transformer with proposed schemes to build three single systems so that PSDS 1 and PSDS 2 can have the best performance:

- (i) CNN-Transformer + ICT, SCT, CCT, Multiscale
- (ii) CRNN + ICT, SCT, CCT, Multiscale
- (iii) CRNN + ICT, SCT, CCT, Multiscale, ECA, ExpSoftmax

Based on mixup data augmentation following the baseline, we find that (i) and (ii) improve the performance on PSDS 1, and (iii) reach significant achievement on PSDS 2. To ensemble the several systems, we apply several data augmentation methods to build each single system, which includes mixup, time-shift, and time-frequency masks, as shown in Table 3. From Table 4, our fusion systems can achieve 40.72% of PSDS 1 and 80.80% of PSDS 2 on the validation set, 37.42% of PSDS 1 and 69.73% of PSDS 2 on the public evaluation set.

Table 1: Results of different schemes, based on two networks with mixup data augmentation.

Scheme	Model	PSDS 1	PSDS 2
-	CRNN	34.04%	53.30%
	CNN-Transformer	33.46%	48.77%
+ICT	CRNN	36.38%	55.87%
	CNN-Transformer	33.39%	50.07%
+SCT	CRNN	**37.86%**	59.47%
	CNN-Transformer	35.61%	52.01%
+CCT	CRNN	37.64%	60.87%
	CNN-Transformer	**37.33%**	55.87%
+Multiscale	CRNN	37.51%	62.63%
	CNN-Transformer	34.75%	**61.10%**
+ECA-Net	CRNN	34.71%	**65.54%**
	CNN-Transformer	35.13%	60.27%

Table 2: Results of different pooling functions, based on above schemes without ECA. y_i and y means frame-level and clip-level.

Pooling Function	Formula	Model	PSDS 1	PSDS 2
Attention	$y = \frac{\sum_i y_i w_i}{\sum_i w_i}$	CRNN	**37.51%**	62.63%
		CNN-Transformer	**34.75%**	61.10%
Max pooling	$y = \max_i y_i$	CRNN	36.10%	64.59%
		CNN-Transformer	31.73%	59.77%
Average pooling	$y = \frac{1}{n}\sum_i y_i$	CRNN	5.34%	73.95%
		CNN-Transformer	4.53%	60.41%
Linear Softmax	$y = \frac{\sum_i y_i^2}{\sum_i y_i}$	CRNN	26.75%	60.17%
		CNN-Transformer	24.21%	60.57%
Exponential Softmax	$y = \frac{\sum_i y_i \exp(y_i)}{\sum_i \exp(y_i)}$	CRNN	5.82%	**75.35%**
		CNN-Transformer	4.13%	**61.31%**

Table 3: Results of different data augmentations, based on three single systems.

#	Model	Schemes	Data Augmentation	PSDS 1	PSDS 2
0	CRNN	-	Mixup ($\alpha = 0.2$)	34.04%	53.30%
1	CNN-Transformer	ICT, SCT, CCT, Multiscale	Mixup ($\alpha = 0.2$)	**34.75%**	61.10%
2			Shift	31.39%	55.05%
3			Masks	33.24%	59.04%
4			Mixup ($\alpha = 0.2$)+Shift	33.43%	58.68%
5			Mixup ($\alpha = 0.2$)+Masks	34.29%	**61.52%**
6			Shift+Masks	33.64%	55.46%
7	CRNN	ICT, SCT, CCT, Multiscale	Mixup ($\alpha = 0.1$)	37.69%	63.00%
8			Mixup ($\alpha = 0.2$)	37.51%	62.63%
9			Mixup ($\alpha = 0.4$)	36.71%	64.82%
10			Mixup ($\alpha = 0.5$)	36.84%	64.18%
11			Mixup ($\alpha = 0.6$)	36.55%	61.85%
12			Mixup ($\alpha = 0.7$)	36.70%	63.91%
13			Shift	35.71%	61.29%
14			Masks	36.96%	64.84%
15			Mixup ($\alpha = 0.2$)+Shift	37.03%	63.02%
16			Mixup ($\alpha = 0.2$)+Masks	**38.13%**	**65.32%**
17	CRNN	ICT, SCT, CCT, Multiscale, ECA, ExpSoftmax	Mixup ($\alpha = 0.1$)	**6.81%**	75.59%
18			Mixup ($\alpha = 0.2$)	5.71%	76.16%
19			Mixup ($\alpha = 0.7$)	5.37%	**76.29%**
20			Shift	4.46%	72.16%
21			Masks	5.29%	75.07%
22			Mixup ($\alpha = 0.2$)+Shift	5.12%	76.19%
23			Mixup ($\alpha = 0.2$)+Masks	4.82%	75.45%
24			Shift+Masks	4.83%	76.08%

Table 4: Results of the fusion systems on the two testing sets.

#	Model	Schemes	Validation		Public eval	
			PSDS 1	PSDS 2	PSDS 1	PSDS 2
7∼16	CRNN	ICT, SCT, CCT, Multiscale	**40.72%**	70.25%	37.22%	69.47%
17∼24	CRNN	ICT, SCT, CCT, Multiscale, ECA-Net, ExpSoftmax	6.08%	**80.80%**	8.30%	65.39%
1∼16	CRNN CNN-Transformer	ICT, SCT, CCT, Multiscale	38.79%	67.18%	**37.45%**	68.42%
1∼24	CRNN CNN-Transformer	ICT, SCT, CCT, Multiscale, ECA-Net, ExpSoftmax	37.02%	72.42%	33.56%	**69.73%**

5. CONCLUSION

Based on the mean-teacher framework of semi-supervised learning with RNNs and Transformer, we present a multi-scale CNNs with ECA-Net to capture various and important features of sound events. For the multiple consistency criteria, ICT helps the model discriminate the ambiguous samples to enhance the generalization ability, SCT assists the model to learn better temporal information, CCT promotes the model feature representation power. Then, an appropriate pooling function is applied to the specific scenario. The data augmentation and posterior-level score fusion further improve the performance. Finally, on the validation set and challenge's evaluation set, our proposed system significantly outperforms the baseline.

6. REFERENCES

[1] C. Clavel, T. Ehrette, and G. Richard, "Events detection for an audio-based surveillance system," in *2005 IEEE International Conference on Multimedia and Expo*. IEEE, 2005, pp. 1306–1309.

[2] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring activities of daily living in smart homes: Understanding human behavior," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 81–94, 2016.

[3] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[4] http://dcase.community/challenge2021/task-sound-event-detection-and-separation-in-domestic-environments, 2020.

[5] N. Turpault, R. Serizel, J. Salamon, and A. P. Shah, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," 2019.

[6] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *arXiv preprint arXiv:1703.01780*, 2017.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[8] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888.

[9] S. V. Katta, S. Umesh, *et al.*, "S-vectors: Speaker embeddings based on transformer's encoder for text-independent speaker verification," *arXiv preprint arXiv:2008.04659*, 2020.

[10] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with self-attention," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 296–303.

[11] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[12] J. Wang and S. Li, "Self-attention mechanism based system for dcase2018 challenge task1 and task4," *Proc. DCASE Challenge*, pp. 1–5, 2018.

[13] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Weakly-supervised sound event detection with self-attention," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 66–70.

[14] M. Tang, L. Guo, Y. Zhang, W. Yan, and Q. Zhao, "Multi-scale residual crnn with data augmentation for dcase 2020 task 4."

[15] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "Eca-net: Efficient channel attention for deep convolutional neural networks, 2020 ieee," in *CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE*, 2020.

[16] V. Verma, K. Kawaguchi, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz, "Interpolation consistency training for semi-supervised learning," *arXiv preprint arXiv:1903.03825*, 2019.

[17] C.-Y. Koh, Y.-S. Chen, Y.-W. Liu, and M. R. Bai, "Sound event detection by consistency training and pseudo-labeling with feature-pyramid convolutional recurrent neural networks," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 376–380.

[18] L. Yang, J. Hao, Z. Hou, and W. Peng, "Two-stage domain adaptation for sound event detection."

[19] https://github.com/DCASE-REPO/DESED_task/tree/master/recipes/dcase2021_task4_baseline, 2021.

[20] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Conformer-based sound event detection with semi-supervised learning and data augmentation," *dim*, vol. 1, p. 4, 2020.

[21] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 524–10 533.

[22] Y. Wang, J. Li, and F. Metze, "A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 31–35.

[23] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder networks," *arXiv preprint arXiv:1507.02672*, 2015.

[24] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[25] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[26] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[27] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[28] R. Serizel, N. Turpault, A. Shah, and J. Salamon, "Sound event detection in synthetic domestic environments," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 86–90.

[29] https://github.com/FFmpeg/FFmpeg, 2020.

[30] https://github.com/librosa/librosa, 2020.

ACOUSTIC EVENT DETECTION USING SPEAKER RECOGNITION TECHNIQUES: MODEL OPTIMIZATION AND EXPLAINABLE FEATURES

Mattson Ogg *Benjamin Skerritt-Davis*

Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road, Laurel, MD 20723 USA
{mattson.ogg, ben.skerritt-davis}@jhuapl.edu

ABSTRACT

We adapted methods from the speaker recognition literature to acoustic event detection (or audio-tagging) and applied representational similarity analysis, a cognitive neuroscience technique, to gain a deeper understanding of model performance. Experiments with a feed-forward time-delay neural network architecture (TDNN) were carried out using the FSDKaggle2018 dataset. We examined various system optimizations such as speed and reverb augmentation, different input features (spectrograms, mel-filterbanks, MFCCs and cochleagrams), as well as updates to the network architecture (increased or decreased temporal context and model capacity as well as drop-out and batch-normalization). Most system configurations were able to outperform the original published baseline and, primarily using speed augmentation, our system was able to outperform a harder baseline derived from a model pre-trained on many times more data. Additional experiments applying representational similarity analysis to the network embeddings allowed us to understand what acoustic features the different systems used to perform the task.

Index Terms— Audio tagging, acoustic event recognition, speaker recognition, explainable features, acoustic features

1. INTRODUCTION

Objects and events in the environment can be recognized based on the sound patterns they generate [1]. Automated sound identification (also called sound/acoustic event recognition or audio tagging) supports many critical information retrieval [2], hearing assistance [3], urban planning [4], and monitoring [5] applications. Recognition systems can also provide insight into human [6] and animal [7] auditory processing. Despite its obvious importance, sound recognition is not a solved problem. Continued development will help address many remaining gaps in performance and yield new insights for acoustics and machine perception research.

Many modern approaches to sound recognition take the rich deep learning literature on visual object recognition as a starting place [8, 9, 10]. These methods essentially treat the task as an image classification problem based on a spectrogram input. However, such an approach has a number of shortcomings (see [11] for discussion). Most notably it is a sub-optimal treatment of the inherently temporal nature of sound. That is, to comply with the constraints of image recognition methods, an incoming audio example is normally chunked into fixed size spectrogram images that are fed to a 2-dimensional convolutional network architecture. Chunks are then classified individually or the systems' hypotheses are averaged over time points. This results in a slightly awkward treatment of audio examples that frequently vary in duration. Also, if examples are labelled at the file-level, individual chunks run the risk of not containing any information related to the target sound due to pauses or interruptions. Indeed, some results have suggested that chunk averaging may be sub-optimal relative to systems designed to handle variable length input [12].

Other lines of research have emerged for identifying specific classes of sound sources such as human speakers [13, 14]. Speaker recognition in particular has made remarkable progress using deep learning techniques that stem from language and speech modeling [15, 16]. These approaches give special attention to how information unfolds in time [17], for example, by statistically pooling time windows to create an intermediate global representation within the network [18] that is further processed. This approach can also be used to recognize acoustic scenes [19].

With these branches of acoustic research in mind we present a series of experiments where we applied successful techniques from speaker identification to the task of sound event recognition. We also study adaptations of this framework to better fit the sound event recognition task. In the end, we achieved accurate performance on a well-studied recognition dataset [20], beating the initial published baseline as well as another stronger baseline that uses an image-recognition-based system pre-trained on many more hours of data [8]. Finally, we provide insight into the performance differences among these systems by exploring their respective embedding spaces using representational similarity analysis [21].

2. STUDY DESIGN

2.1. Data and Task

Our goal was to apply speaker recognition techniques to sound event recognition. Thus, we aimed to select a dataset that was organized analogously to typical speaker recognition datasets [14, 22]. This involves a large number of discrete target classes with numerous training examples where each example corresponds to a single target class. We chose to conduct our experiments using the popular audio-tagging dataset, FSDKaggle2018 (see [20] for details). Briefly, this is a curated collection of publicly available data that is user generated and user tagged. The data was then binned by the authors of the corpus into 41 discrete classes

according to a well-established acoustic event ontology [2]. Clips vary in duration and the number of training examples is unbalanced across classes. Our experiments were carried out on the full (manual and user-labelled) portion of the dataset. This dataset does not contain a validation partition, so we created one by holding out one third of the manually labelled examples of each class from the training partition. Model selection was done on this validation set. We then report final system performance of the selected model on the test partition. All audio data were down-sampled to a 16kHz sampling rate.

We selected this dataset because, relative to other options, it strikes a good balance in terms of dataset size (up to 18 hours of training data), diversity (41 discrete classes), and label specificity (generally, a single class-label per clip). Some more thoroughly labelled datasets exist but are limited in their number of target classes [23] or training examples [24]. While other larger datasets often sacrifice label purity and usually contain multiple classes per clip [2, 4, 9].

We compare performance against the published baseline for this dataset from [20] (mAP@3 = 0.70). A harder baseline was also generated using Google's YAMNet system trained on their AudioSet corpus [8, 25]. We extracted YAMNet embeddings for each sound token (no augmentation) and then trained a single fully connected layer between those embeddings and 41 output units for the corpus' target classes. The shallow YAMNet embedding network was trained for 100 epochs and we retained the model with the highest performance on the validation partition (validation performance: accuracy = 0.79, mAP@3 = 0.86; test performance: accuracy = 0.78, mAP@3 = 0.85).

2.2. Network Architecture and Training

Our initial TDNN model was an implementation of the x-vector system developed by Snyder and colleagues for speaker recognition [18, 26] that we reproduced in PyTorch (using [27]). Because it is based on feed-forward units, TDNN networks are faster and more efficient to train than recurrent networks, such as LSTMs (c.f., [17]).

A TDNN models temporal context via a hierarchy of layers that progressively see larger windows of time via dilations that occur in higher layers. Variable length audio input is handled by a combination of frame-level and segment-level components within the model. At the frame-level, the TDNN structure slides over frames of the variable length input. The output of these layers is projected to a set of units over which the mean and standard deviation are calculated for a given example (audio file). The mean and standard deviation of these units are concatenated to comprise a statistics pooling layer that begins the segment-level processing. Above the statistics pooling layer are two fully connected layers (which comprise the embedding layers) followed by 41 output units, one for each of the target classes.

Specifically, we re-created the architecture described by [26] with a context of five input time-frames (of the input spectrogram) in the first layer. Layer two received layer one's output with a context of three frames and a dilation of two before sending output to layer three which also has a context of three frames with a dilation of three. Layers four and five both have contexts of one. Thus, layers three and higher operated over a total context of 15 spectrogram frames. All TDNN and embedding layers have 512 units. The layer just prior to the statistics pooling layer projected to 1500 units that were used to calculate a mean and standard deviation which were concatenated before output to the first fully connected layer, followed by the second fully connected layer (each with 512 units).

Audio files were represented to the network via a time-frequency representation. Speaker and sound event recognition studies have used a variety of different inputs, so we tested multiple popular representations to determine an optimal set of features. We explored many Kaldi-style representations using torchaudio all with a 25-ms window and 10-ms hop size. Because we used a higher sample rate (16k) than the original x-vector network implementation (8k), we allowed an increase in the number of frequency bins for each representation: spectrograms (201 frequency bins), mel-filterbanks (80 mel bins) and MFCCs (mel-frequency cepstral coefficients; 40 cepstral features). We also generated a cochleagram representation to approximate the peripheral auditory system of a human listener (which we instantiated via [28]; upper frequency limit = 8k, 4 times overcomplete bandpass filter sampling, output down sampled to 100 Hz). All audio was zero-padded for 5-ms at onset and up to 5 ms at offset before windowing. During training, each input spectrogram was normalized (between 0 and 1), and the durations of input examples were standardized to between 1-second and 30 seconds either via looping spectrograms that were too short (until they exceeded 101 frames), or by truncating them (to 3001 frames if they exceeded that).

Each training run comprised 100 epochs and we retained the model from the epoch with the highest accuracy on the

Table 1: Performance of different model architectures and training configurations on the validation partition. Parentheses indicate change in performance from our baseline TDNN model on the first line.

	Accuracy				mAP@3			
	Cochleagram	Mel-Filterbank	MFCC	Spectrogram	Cochleagram	Mel-Filterbank	MFCC	Spectrogram
Initial Baseline TDNN Model	0.73(Δ0)	0.74(Δ0)	0.24(Δ0)	0.75(Δ0)	0.79(Δ0)	0.8(Δ0)	0.32(Δ0)	0.81(Δ0)
Diff. Maps	0.74(Δ0.016)	0.75(Δ0.012)	0.16(Δ-0.078)	0.75(Δ0.002)	0.8(Δ0.006)	0.81(Δ0.007)	0.24(Δ-0.087)	0.81(Δ-0.005)
Reverb Aug.	0.77(Δ0.043)	0.79(Δ0.043)	0.7(Δ0.464)	0.77(Δ0.022)	0.82(Δ0.03)	0.84(Δ0.035)	0.78(Δ0.453)	0.82(Δ0.011)
Speed Aug.	0.83(Δ0.099)	0.88(Δ0.134)	0.76(Δ0.526)	0.87(Δ0.117)	0.87(Δ0.079)	0.91(Δ0.103)	0.82(Δ0.493)	0.9(Δ0.089)
Smaller Net: 256 Units	0.72(Δ-0.004)	0.76(Δ0.019)	0.45(Δ0.218)	0.74(Δ-0.01)	0.78(Δ-0.012)	0.82(Δ0.013)	0.56(Δ0.234)	0.8(Δ-0.011)
Larger Net: 1024 Units	0.74(Δ0.015)	0.75(Δ0.007)	0.4(Δ0.166)	0.76(Δ0.007)	0.8(Δ0.009)	0.81(Δ0.007)	0.5(Δ0.178)	0.81(Δ0.001)
Reduced Context-Layer	0.72(Δ-0.011)	0.74(Δ-0.002)	0.43(Δ0.194)	0.73(Δ-0.02)	0.78(Δ-0.018)	0.81(Δ0.002)	0.54(Δ0.216)	0.8(Δ-0.015)
Added Context-Layer	0.74(Δ0.007)	0.75(Δ0.011)	0.56(Δ0.326)	0.74(Δ-0.012)	0.79(Δ-0.001)	0.81(Δ0.001)	0.65(Δ0.332)	0.8(Δ-0.015)
Batch-Norm, Drop-Out	0.76(Δ0.031)	0.8(Δ0.055)	0.63(Δ0.39)	0.78(Δ0.033)	0.81(Δ0.016)	0.85(Δ0.044)	0.71(Δ0.392)	0.84(Δ0.026)
Speed+Reverb Aug.	NA	0.87(Δ0.126)	NA	NA	NA	0.9(Δ0.093)	NA	NA
Speed+Reverb, Diff. Maps, Batch-Norm+Drop-Out	NA	0.86(Δ0.113)	NA	NA	NA	0.89(Δ0.084)	NA	NA
Speed+Reverb, Batch-Norm+Drop-Out	NA	0.85(Δ0.109)	NA	NA	NA	0.89(Δ0.084)	NA	NA
Speed+Reverb, Diff. Maps	NA	0.87(Δ0.129)	NA	NA	NA	0.91(Δ0.1)	NA	NA

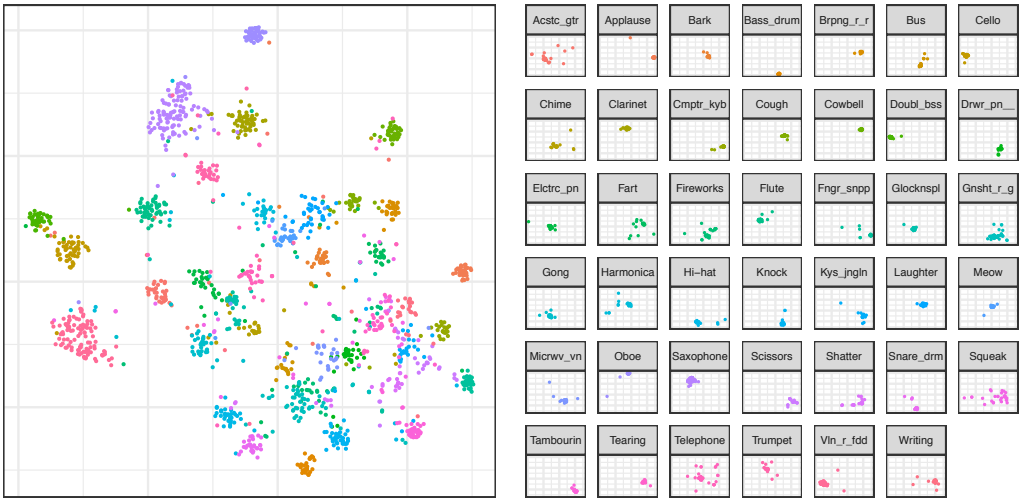


Figure 1: Our top performing TDNN model's embedding space visualized using t-SNE. Points correspond to examples in the test set. Insets in the right panel call out examples for each target class and serves as a color code.

validation partition. Batches were of 16 examples (shuffled between epochs). We used PyTorch's cross-entropy loss function with a stochastic gradient descent optimizer (learning rate: 0.001, momentum: 0.9, weight decay: 0.001). Because there was an imbalance of training data among classes, weights were applied in the loss function that gave more weight to low occurrence classes relative to the class with the highest number of training examples. We report raw accuracy on the validation and test partitions as well as mean Average Precision @ 3 (mAP@3).

2.3. Study of Network Architecture and Training Parameters

We carried out a number of experiments to optimize inputs and model parameters during training. The baseline for these experiments was the performance of a network structured like the original x-vector network configuration [26], albeit with a larger input representation to take advantage of the higher sampling rate (see above). Throughout, we compared the performance of four front-end, time-frequency representations (spectrogram, MFCC, mel-filterbank, or cochleagram input). In terms of input optimizations we also attempted to help the network efficiently learn spectral and temporal variability cues by appending 2 "difference maps" to the input time-frequency representation: 1) the first derivative in each frequency channel over time and, 2) first derivative in each time bin over frequencies.

We then examined the effectiveness of common data augmentation strategies gleaned from work on speech tasks [29, 30] (see also [31]): simple speed augmentation (plus and minus 10%, thus altering any pitch by the same amount) and reverb augmentation (instantiated via [32]), both of which were carried out on the audio files prior to extracting a time-frequency representation. Augmentation by background noise and babble was not examined, since these recordings already contained some. Sounds similar to those in this dataset are also often used in noise augmentation for speech tasks, which risked confusing class labels during training.

Next, we turned our attention to optimizing the network architecture in various ways: varying the number of units in each layer (feed forward layer 256 or 1024 and number of statistics pooling layer units 750 and 3000, respectively), increasing or decreasing temporal context before statistical pooling (by removing or duplicating layer 3), and adding batch norm and 50% dropout. Based on these experiments we studied a final set of models that used combinations of the best performing parameters and optimizations.

3. RESULTS

Model performance is summarized within Table 1. Differences relative to our baseline TDNN configuration are indicated in parentheses. Without any modifications the initial TDNN models outperformed the original published baseline given most of the feature input options, although performance using MFCCs was generally poor. Not every optimization we experimented with improved performance and some optimizations varied in how much they improved performance given different input features.

Table 2: Description of some acoustic features used in the representational similarity analyses.

Feature	Description	Interpretation
Log-Attack-Time[1]	Log of the time difference between attack onset and ending	Lower values = faster onset time
Temporal Centroid[1]	Center of gravity of the energy envelope	Lower values = earlier temporal centroid
Spectral Centroid[2]	Center of gravity of the spectral (ERB) envelope	High values = higher frequency centroid
Spectral Flatness[2]	Ratio of geometric and arithmetic means of the ERB spectra	Measures noise/harmonic content. Higher values are flatter/noisier
Spectral Variability[2]	1 minus the correlation of ERB channel spectra between timepoints	Higher values = more variable envelope
Aperiodicity[3]	Amount of aperiodic energy in the signal	Higher values = more aperiodic
ERB Energy[2]	Amount of energy in the spectral representation at each timestep	Sum of squared amplitudes in the spectral representation.
Raw ERB cochleagram[2]	Raw ERB representation of each of 77 channels: 30 Hz to 16 kHz	Energy in each channel over time.
Mod. Power Spectrum[4]	2D-FFT of Gaussian spectrogram	Degree of joint spectral/temporal modulation rates

Derived based on: 1) Energy envelope or, 2) ERB (cochleagram) representation in the Timbre Toolbox [35, 36], 3) YIN [37], 4) modulation power spectrum [38]. Table adapted from [33].

Speed and, to a lesser degree, reverb augmentation were particularly beneficial, as was batch-norm and drop-out. Changing temporal context and model capacity often hurt performance.

Only with speed augmentation were some of our TDNN models able to beat the performance of the stronger YAMNet-baseline system on the validation partition. To further improve our system we explored combinations of optimizations that were beneficial in the initial experiments. These combination experiments were carried out using mel-filterbank input features because these achieved the highest performance relative to other initial TDNN systems. A model trained with speed and reverb augmentation with the spectrotemporal difference maps using the mel-filterbanks as input features achieved the highest validation performance among these combination experiments. However, no combination experiments out-performed the initial mel-filterbank TDNN model trained with speed-augmented data, so this was selected as our final model to evaluate the test data (accuracy = 0.82, mAP@3 = 0.86), which slightly out-performed the YAMNet-baseline. Class separation within the embedding space of this top-performing model is visualized in Figure 1.

4. REPRESENTATIONAL SIMILARITY ANALYSIS

Performance of our final model and the YAMNet-baseline were both quite high, despite operating over the audio differently. Thus, we were interested in better understanding whether any differences existed in how these systems internally represented audio examples and the influence of different acoustic qualities. To do this, we employed a method called representational similarity analysis [21] which can provide a high-level understanding of complex systems by correlating inter-item distances among different representations of a set of probe examples. We extracted network embeddings (i.e., activations from the layer just prior to the 41-class output layer) from our final, best performing model and from the YAMNet model for each example in the test partition. Then among each model's embeddings, we calculated the cosine distance of the network representations for each pair of test examples, to populate two 1600 by 1600 network-dissimilarity matrices (one matrix for each network). The network-dissimilarity matrices were compared against another set of inter-item, acoustic-dissimilarity matrices (absolute value of feature differences) for a set of well-studied acoustic features derived for each test item (see Table 2 and [33] for detailed description). These acoustic distances were contained within another set of 1600 by 1600 acoustic-dissimilarity matrices, (one matrix per feature). Note, because these dissimilarity matrices are symmetrical across the diagonal, only one unique item pairing was analyzed (e.g., item-1 vs item-2 or item-2 vs item-1).

We carried out rank-order semi-partial Spearman correlations between each network-dissimilarity matrix and the set of acoustic-dissimilarity matrices. In each semi-partial test, a correlation was derived between the network-dissimilarity matrix, and the target acoustic-dissimilarity matrix, while holding the other features constant. Only correlations that were interpretable (i.e., positive) and statistically significant after false-discovery rate correction were retained.

The representational similarity analysis is summarized in Figure 2. We found that despite their high performance, our model and the YAMNet model's embedding spaces were only modestly correlated (r_s = 0.31). Both models' performance was

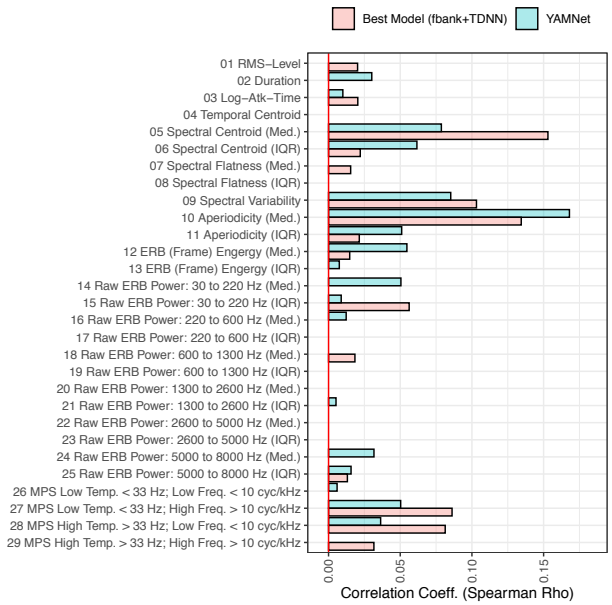


Figure 2: Representational similarity analysis of both the top performing TDNN model and the YAMNet embeddings.

most strongly associated with acoustic cues for aperiodicity, spectral centroid, and spectral variability, albeit with differences in the relative importance of these features. This is similar to features that influence dissimilarity ratings among human listeners [33] and neural representations [34].

5. CONCLUSION

We examined the effectiveness of different speaker recognition methods on an audio-tagging task. We were able to obtain good performance, beating the original baseline for this dataset, and a more challenging YAMNet-baseline derived from a system trained on many hours more data. The TDNN architecture appears to derive great performance benefit from data augmentation (particularly speed augmentation). Representational similarity analyses implicated a set of acoustic features that are also associated with sound recognition in the human auditory system.

6. REFERENCES

[1] F. E. Theunissen, and J. E. Elie, "Neural processing of natural sounds," *Nat. Rev. Neurosci.*, vol. 15, pp. 355–366, 2014.

[2] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, ... and Ritter, M. "Audio set: An ontology and human-labeled dataset for audio events," In *Proc. IEEE ICASSP*, 2017, pp. 776-780.

[3] A. Hüwel, K. Adiloğl, and J. H. Bach, "Hearing aid research data set for acoustic environment recognition." In *Proc. IEEE ICASSP,* 2020, pp. 706-710.

[4] M. Cartwright, A. E. M. Mendez, J. Cramer, V. Lostanlen, G. Dove, H.-H. Wu, J. Salamon, O. Nov, and J. Bello, "SONYC urban sound tagging (SONYC-UST): A multilabel dataset from an urban acoustic sensor network," in *Proc. IEEE DCASE*, 2019, pp. 35–39.

[5] D. Stowell, T. Petrusková, M. Šálek, and P. Linhart, “Automatic acoustic identification of individuals in multiple species: improving identification across recording conditions,” *Journal of The Royal Society Interface*, vol. 16, 20180940, 2019.
[6] A. Kell, D. Yamins, E. N. Shook, S. Norman-haignere, and J. H. McDermott, "A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy," *Neuron*, vol. 98, 2018.
[7] J. E. Elie, and F. E. Theunissen, “Zebra finches identify individuals using vocal signatures unique to each call type,” *Nat. Comm.*, vol. 9, pp. 1-11, 2018.
[8] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J.F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R.A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, “CNN architectures for large-scale audio classification,” in *IEEE ICASSP*, 2017, pp. 131–135.
[9] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, “FSD50K: An open dataset of human-labeled sound events,” *arXiv preprint arXiv:2010.00475*, 2020.
[10] A. Guzhov, F. Raue, J. Hees, and A. Dengel, “ESResNet: Environmental sound classification based on visual domain models,” *arXiv preprint arXiv:2004.07301*, 2020.
[11] E. M. Kaya and M. Elhilali, “Modelling auditory attention,” *Philosophical transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 372, no. 1714, p. 20160101, 2017.
[12] A. Nagrani, J. S. Chung, and A. Zisserman, “Voxceleb: A large-scale speaker identification dataset,” in *Proc. Interspeech*, 2017, pp. 2616–2620.
[13] J.H.L. Hansen and T. Hasan, “Speaker recognition by machines and humans: A tutorial review,” *IEEE Signal Proc. Mag.*, vol. 32, no. 6, pp. 74–99, 2015.
[14] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, “Voxceleb: Large-scale speaker verification in the wild,” *Computer Speech & Language*, vol. 60, 101027, 2020.
[15] V. Peddinti, D. Povey, and S. Khudanpur, “A time delay neural network architecture for efficient modeling of long temporal contexts,” in *Proc. Interspeech*, 2015.
[16] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, “Phoneme recognition using time-delay neural networks,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, Mar. 1989.
[17] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, “End-to-end text-dependent speaker verification,” in *Proc. IEEE ICASSP*, 2016, pp. 5115–5119.
[18] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, “Deep neural network embeddings for text-independent speaker verification,” in *Proc. Interspeech*, pp. 999–1003, 2017.
[19] A. Jati, A. Nadarajan, K. Mundnich, and S. Narayanan “Characterizing dynamically varying acoustic scenes from egocentric audio recordings in workplace setting,” *arXiv preprint arXiv:1911.03843*, 2019.
[20] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, “General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline,” in *Proc. IEEE DCASE*, 2018, pp. 69–73.
[21] N. Kriegeskorte and R. A. Kievit, ‘‘Representational geometry: Integrating cognition, computation, and the brain,’’ *Trends Cognit. Sci.*, vol. 17, no. 8, pp. 401–412, Aug. 2013.
[22] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, “The speakers in the wild (SITW) speaker recognition database,” in *Proc. Interspeech*, 2016, pp. 818–822.
[23] J. Salamon, C. Jacoby, and J. P. Bello, “A dataset and taxonomy for urban sound research,” in *22nd ACM International Conference on Multimedia (ACM-MM’14)*, 2014, pp. 1041–1044.
[24] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *25th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2015, pp. 1–6.
[25] https://github.com/tensorflow/models/tree/master/research/audioset/yamnet
[26] D. Snyder, D. Garcia-Romero, G. Sell, et al., “X-vectors: Robust DNN embeddings for speaker recognition,” in *Proc. IEEE ICASSP*, 2018, pp. 5329–5333.
[27] https://github.com/cvqluu/TDNN
[28] https://pycochleagram.readthedocs.io/en/latest/index.html
[29] T. Ko, V. Peddinti, D. Povey, M. Seltzer, and S. Khudanpur, “A study on data augmentation of reverberant speech for robust speech recognition,” in Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on. IEEE, 2017, pp. 5220–5224.
[30] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, “Audio augmentation for speech recognition,” in *Proc. INTERSPEECH*, 2015, pp. 3586–3589.
[31] J. Salamon and J. P. Bello, “Deep convolutional neural networks and data augmentation for environmental sound classification,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.
[32] https://github.com/mravanelli/pySpeechRev
[33] M. Ogg, and R. L. Slevc, “Acoustic correlates of auditory object and event perception: Speakers, musical timbres, and environmental sounds,” *Front. Psychol.*, vol. 10, 1594, 2019.
[34] M. Ogg, T. A. Carlson, and R. L. Slevc, “The rapid emergence of auditory object representations in cortex reflect central acoustic attributes,” *J. Cogn. Neurosci.*, vol. 32, pp. 111–123.
[35] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, “The timbre toolbox: Extracting audio descriptors from musical signal,” *J. Acoust. Soc. Am.*, vol. 130, no. 5, pp. 2902–2916, 2011.
[36] S. Kazazis, E. Nicholas, P. Depalle, and S. McAdams, “A performance evaluation of the timbre toolbox and the mirtoolbox on calibrated test sounds,” in *Proc. of the Int. Symposium on Musical Acoustics (ISMA)*, 2017, pp. 144–147.
[37] A. De Cheveigné, and H. Kawahara, “YIN, a fundamental frequency estimator for speech and music,” *J. Acoust. Soc. Am.*, vol. 111, pp. 1917–1930, 2002.
[38] T. M. Elliott and F. E. Theunissen, “The modulation transfer function for speech iintelligibility,” *PLoS Comput. Biol.*, vol. 5, no. 3, pp. 1–14, Mar. 2009.

LOW-COMPLEXITY ACOUSTIC SCENE CLASSIFICATION FOR MULTI-DEVICE AUDIO: ANALYSIS OF DCASE 2021 CHALLENGE SYSTEMS

Irene Martín-Morató, Toni Heittola, Annamaria Mesaros, Tuomas Virtanen

Computing Sciences
Tampere University, Finland
{irene.martinmorato, toni.heittola, annamaria.mesaros, tuomas.virtanen}@tuni.fi

ABSTRACT

This paper presents the details of Task 1A Low-Complexity Acoustic Scene Classification with Multiple Devices in the DCASE 2021 Challenge. The task targeted development of low-complexity solutions with good generalization properties. The provided baseline system is based on a CNN architecture and post-training quantization of parameters. The system is trained using all the available training data, without any specific technique for handling device mismatch, and obtains an accuracy of 47.7%, with a log loss of 1.473. The task received 99 submissions from 30 teams, and most of the submitted systems outperformed the baseline. The most used techniques among the submissions were residual networks and weight quantization, with the top systems reaching over 70% accuracy, and log loss under 0.8. The acoustic scene classification task remained a popular task in the challenge, despite the increasing difficulty of the setup.

Index Terms— Acoustic scene classification, multiple devices, low-complexity, DCASE Challenge

1. INTRODUCTION

Acoustic scene classification aims to classify a short audio recording into a set of predefined classes, based on labels that indicate where the audio was recorded [1]. The popularity of the task in DCASE Challenge throughout the years has allowed the development of approaches diverging from the textbook supervised classification, introducing along the years different devices [2], open-set classification, and low-complexity conditions [3], along with the publication of suitable datasets.

The problem of classification of acoustic scenes from different recording devices is illustrated in Fig. 1. Performance and generalization properties of such a system are strongly affected by mismatches between training and testing data, including recording device mismatch. A variety of solutions were proposed for dealing with the mismatch: for example in DCASE 2019 challenge, Kosmider et al. [4] used a spectrum correction method to account for different frequency responses of the devices in the dataset, based on the fact that the provided development data contained temporally aligned recordings from different devices. Other systems used multiple forms of regularization that involves aggressively large value for weight decay, along with mixup and temporal crop augmentation [5]. In DCASE 2020 Challenge, most of the submitted systems used multiple forms of data augmentation including resizing and cropping, spectrum correction, pitch shifting, and SpecAugment, which seems to compensate for the device mismatch [3]. The top system had an accuracy of 76.5% (1.21 log loss), using residual networks for the 10 scene classification with mismatched data [6].

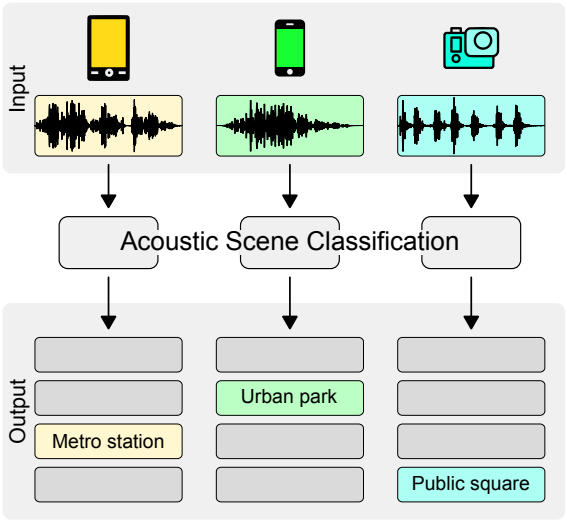


Figure 1: Acoustic scene classification for audio recordings.

In addition to dealing with data collected from devices not available during training, real-world solutions also need to be able to operate on devices with limited computational capacity [7]. For instance, in SED, dilated CNN have been applied to reduce the number of model's parameters [8] whereas in [9], network dimensions have been scaled to obtain smaller and efficient architectures. In DCASE 2020, the low-complexity classification task consisted of a 3-class problem, to which many of the submissions imposed restrictions on the model architectures and their representations, such as using slim models, depth-wise separable CNNs, pruning and post-training quantization of model weights [3]. The top system [10] used a combination of pruning and quantization, using 16-bit float representation for the model weights and having a reported sparsity of 0.28 (ratio of zero-valued parameters), obtaining 96.5% accuracy (0.10 log loss).

In DCASE 2021 Challenge, the two problems are merged, targeting good performance for a 10-class setup, multiple devices, and with model size constraints. The added difficulty of the task comes from imposing more demanding conditions on both studied directions: using 10 classes instead of the three classes like in 2020, and dropping the model size limit from 500KB to 128KB. The choice

This work was supported in part by the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND.

of these conditions is motivated by the good performance demonstrated in DCASE 2020 Challenge, which showed that it was possible to achieve high performance results with a low complexity model.

This paper introduces the results and analysis of the DCASE 2021 Challenge Subtask A: Low-Complexity Acoustic Scene Classification with Multiple Devices. The paper is organized as follows: Section 2 introduces shortly the setup, dataset, and baseline system. Section 3 presents the challenge participation statistics in terms of numbers and use of methods, and Section 4 presents a detailed analysis of the submitted systems. Finally, Section 5 presents conclusions and thoughts for future development of this task.

2. TASK SETUP

The specific feature of this task for acoustic scene classification is generalization across a number of different devices, while enforcing a limit on the model size. The 11 different devices in the dataset include real and simulated devices, and the model limit is 128 KB.

2.1. Dataset and performance evaluation

The task used the **TAU Urban Acoustic Scenes 2020 Mobile** dataset [11, 12]. The dataset is the same as used in DCASE 2020 Challenge Task 1A, comprised of recordings from multiple European cities, ten acoustic scenes [13]: *airport*, *indoor shopping mall*, *metro station*, *pedestrian street*, *public square*, *street with medium level of traffic*, *travelling by a tram*, *travelling by a bus*, *travelling by an underground metro* and *urban park*. Four devices used to record audio simultaneously are denoted as A, B, C, and D (real devices), with an additional 11 devices S1-S11 simulated using the audio from device A. The development/evaluation split, and the training/test split of the development set are the same as in the previous challenge, with 64 hours of audio available in the development set and 22 hours in the evaluation set. For details on the dataset creation, and the amount of data available from each device, we refer the reader to [3].

We evaluate the submitted system using multi-class cross-entropy and accuracy. Accuracy is calculated as macro-average (average of the class-wise performance for each metric), but because the data is balanced, this corresponds to the overall accuracy. We use multi-class cross-entropy (log loss) for ranking the systems, in order to create a ranking independent of the operating point. Accuracy values are provided for comparison with the methods evaluated in previous years.

2.2. System complexity requirements

A model complexity limit of 128 KB was set for the non-zero parameters. This limit allows 32768 in float32 (32-bit float) representation, which is often the default data type (32768 parameter values * 32 bits per parameter / 8 bits per byte= 131072 bytes = 128 KB). Different implementation may consider minimizing the number of non-zero parameters of the network in order to comply with this size limit, or representation of the model parameters with a low number of bits.

The computational complexity of the feature extraction stage is not included in this limit, due to a lack of established methodology for estimating and comparing complexity of different low-level feature extraction implementations. By excluding the feature extraction stage, we keep the complexity estimation straightforward across approaches. Some implementations may use a feature extraction layer as the first layer in the neural network - in this case the limit is applied only to the following layers, in order to exclude the feature calculation as if it were a separate processing block. However, in the special case of using learned features (so-called embeddings, like VGGish [14], OpenL3 [15] or EdgeL3 [16]), the network used to generate them counts in the calculated model size.

System	Log loss	Accuracy	Size
keras	1.473 ($\pm$ 0.051)	47.7% ($\pm$ 0.9)	90.3KB

Table 1: Baseline system size and performance on the development dataset

3. BASELINE SYSTEM

The baseline system is based on a convolutional neural network (CNN) with the addition of model parameters quantization to 16 bits (float16) after training, code available on github[1]. The system uses 40 log mel-band energies, calculated with an analysis frame of 40 ms and 50% hop size, to create an input shape of 40×500 for each 10 second audio file. The neural network consists of three CNN layers and one fully connected layer, followed by the softmax output layer. Learning is performed for 200 epochs with a batch size of 16, using Adam optimizer and a learning rate of 0.001. Model selection and performance calculation are done similar to the baseline system in DCASE 2020 Challenge Subtask A. Quantization of the model is done using Keras backend in TensorFlow 2.0 [17], after training the model, the weights are set to *float16* type. The final model size of the system after quantifying is 90.3 KB.

The classification results on the development dataset training/test split is presented in Table 2. The class-wise log loss is calculated taking into account only the test items belonging to the considered class (splitting the classification task into ten different sub-problems), while overall log loss is calculated taking into account all test items. Given the results shown in this table, *shopping mall* is the class with the lowest log loss, while public square has the highest log loss, being the most difficult to classify. The system behaves similarly to previous year challenge task 1A, the loss only increases 0.108 while the accuracy drops 6.4 points.

[1] https://github.com/marmoi/dcase2021_task1a_baseline

Scene label	Log Loss	Accuracy
Airport	1.43	40.5%
Bus	1.32	47.1%
Metro	1.32	51.9%
Metro station	1.99	28.3%
Park	1.17	69.0%
Public square	2.14	25.3%
Shopping mall	1.09	61.3%
Pedestrian street	1.83	38.7%
Traffic street	1.34	62.0%
Tram	1.10	53.0%
Overall	**1.473**	**47.7%**

Table 2: Class-wise performance of the baseline system on the development dataset.

Rank	System	Logloss ±95% CI	Acc ±95% CI(%)	Size (KB)	Weights	Sparsity	Learning	Architecture
1	Kim_QTI_2	0.72±0.03	76.1±0.94	121.9	int8	✓	KD	BC-ResNet
3	Yang_GT_3	0.74±0.02	73.4±0.97	125.0	int8	✓	KD	Ensemble
9	Koutini_CPJKU_3	0.83±0.03	72.1±0.99	126.2	float16	✓	grouping CNN	CP_ResNet
12	Heo_Clova_4	0.87±0.02	70.1±1.01	124.1	float16	-	KD	ResNet
13	Liu_UESTC_3	0.88±0.02	69.6±1.01	42.5	1-bit	-	-	ResNet
17	Byttebier_IDLab_4	0.91±0.02	68.8±1.02	121.9	int8	✓	grouping CNN	ResNet
19	Verbitskiy_DS_4	0.92±0.02	68.1±1.03	121.8	float16	-	-	EfficientNet
22	Puy_VAI_3	0.94±0.02	66.2±1.04	122.0	float16	-	focal loss	Separable CNN
25	Jeong_ETRI_2	0.95±0.03	67.0±1.04	113.9	float16	-	-	Trident ResNet
28	Kim_KNU_2	1.01±0.03	63.8±1.06	125.1	int8	-	mean-teacher	Shallow inception
85	Baseline	1.73±0.05	45.6±1.10	90.3	float16	-	-	CNN

Table 3: Performance on the evaluation set and complexity management techniques for selected top systems (best system of each team). "KD" refers to Knowledge Distillation and "BC" stands for Broadcasting.

4. CHALLENGE RESULTS

This section presents the challenge results and an analysis of the submitted systems. A total of 99 systems were submitted for this task from 30 teams. The number of participants for the ASC task is steady in the recent years, showing that its popularity does not decrease, but continues to attract attention through different setups.

The highest accuracy obtained for the classification was 76.1%, for the system of Kim_QTI [18], with the same system also having the best log loss of 0.724. Overall, 18 submitted systems had over 70% accuracy. The performance and a few selected characteristics for systems submitted by the top 10 teams (best system of each team) are presented in Table 3. Confidence intervals for log loss were calculated using the jackknife estimation as in [19]. Complete results are available on the task webpage[2].

The ranking of systems is based on log loss; if the systems would be ranked by accuracy, their order would be quite different: while top 3 teams would keep their spots, teams ranked 12th and 27th would move to ranks 4th and 8th. Systems ranked 1st, 2nd, 9th and 10th would keep their place, while the others in between would be shuffled. We calculated the Spearman rank correlation between accuracy and log loss, to investigate the strength of the association between the two. The correlation is 0.73, which, while strong, indicates quite significant changes in the ranking over the entire list of 99 systems.

4.1. Features and augmentation techniques

All top 10 teams make use of mel energies as feature representation, ranging from 40 to 256 mel bands, with three of them adding also delta and delta-delta values of the energies. Overall, only three out of 30 teams do not use log mel as input features; instead, they use gammatone (Naranjo-Alcazar_ITI), deep scattering spectrum (Kek_NU) and embeddings from AemNet (Galindo-Meza_ITESO). Augmentation techniques are also used, with most popular techniques being mixup (used by 20 teams) and specAugment (10 teams). Other augmentation techniques used, known as label-invariant transformations, are pitch shifting, temporal cropping or speed change, and they are commonly used to improve the performance of CNN networks.

[2]http://dcase.community/challenge2021/task-acoustic-scene-classification-results-a

4.2. Architectures

Residual models are the most popular ones; in fact a total of 15 teams use them, among them the top five models, with the exception of the second best team, Yang_GT, which uses ensembles of CNNs. In the literature there are only a handful of models suitable for usage with devices constrained by processing power and/or memory. Some of these models are MobileNet [20] and EfficientNet [21], which are networks based on residual blocks. The most recent one, EfficientNet, also contains squeeze-and-excitation blocks. A total of five teams used some modified version of such models. Finally, the two models that perform below the baseline accuracy make use of fully convolutional models.

4.3. System complexity

Regarding model complexity, the top 10 systems, belonging to three different teams, Kim_QTI [18], Yang_GT [22] and Koutini_CPJKU [23], are close to the allowed model size limit. They range from 110 KB to 126.81 KB, with the system ranked first having a size of 121.9KB. We have to go down to position 77 (1.464 log loss, 47.17% accuracy) to find the smallest model of 29 KB by Singh_IITMandi, which used a filter pruning strategy consisting of 3 steps and one extra for final quantization of the weights to 16-bits.

A notable small model, with size 42.5KB, belongs to a top 5 ranked team, Liu_UESTC [24]. This specific system is ranked 13th, with a 0.878 log loss and 69.60% accuracy. The model compression is performed with 1-bit quantization, similar to the McDonnell_USA system from DCASE2020 Challenge Task 1B [5]. Despite the high performance in DCASE2020, this is the only team using the one-bit quantization approach this year.

There are only two teams that do not use any quantization: Pham_AIT [25] uses channel restriction and decomposed convolution, while Qiao_NCUT does not mention any quantization; however, these are not in the top 10 ranked teams. On the other hand, 11 teams perform pruning with some quantization technique, and two teams perform the Lottery Ticket Hypothesis (LTH) [26] pruning method. One achieved second position, with a model of size 125KB, while the other stayed below the baseline with a model size of 124KB. The main difference between the two is the use of ensemble of CNN with knowledge distillation vs a single CNN model.

Therefore, sparsity used in combination with quantification is a

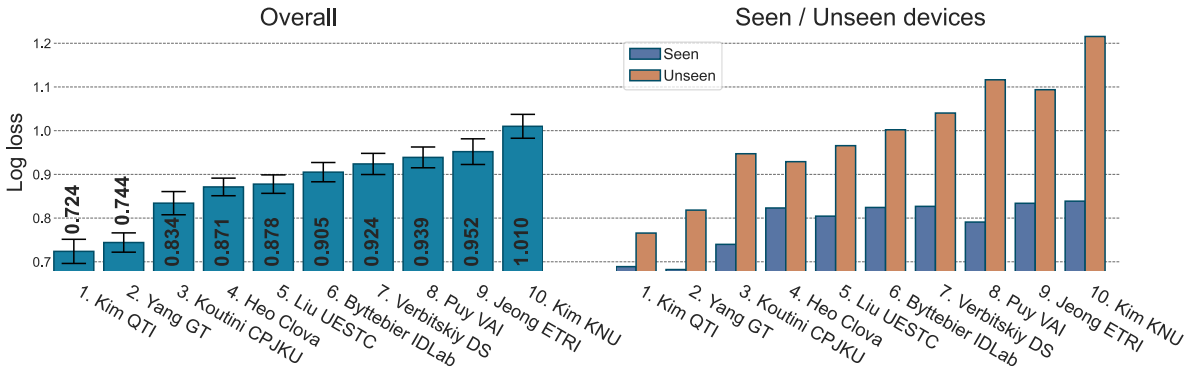


Figure 2: Classification log loss for the 10 top teams (best system per team) on the evaluation dataset.

very popular and efficient way of reducing the model size; however, model architecture and other learning techniques have to be taken into account in order to achieve good classification performance.

4.4. Device and class-wise performance

All systems have higher performance on the devices seen during training (A, B, C, S1, S2, S3) than on the unseen ones (D, S7, S8, S9, S10), with a difference in accuracy of almost 3% (statistically significant) for the system ranked first. As seen in Figure 2, this difference increases as we go down the system ranking, reaching an almost 10% gap when considering accuracy, and 0.37 when considering log loss, for team Kim_KNU. The Spearman's rank correlation between the accuracy on seen and accuracy on unseen devices is 0.92, while between the log loss on the seen devices and the log loss on the unseen devices is 0.91. These values indicate that while they are very highly correlated, the gap between the two does not always preserve the ranking order.

The generalization properties of the systems are worst regarding the unseen devices, while for seen/unseen cities the performance does not vary as much. Some systems get better performance for unseen cities. Indeed, the correlation between performance on seen cities and on unseen cities is 0.95, while device-wise is 0.91. This indicates that data mismatch due to the unseen devices is more challenging than the mismatch created by different cities, due to the different properties of the recorded audio, which are related to the device-specific processing. In particular, the poor performance on the unseen devices is mostly due to device **D**, which is the GoPro, while the other devices are real and simulated mobile phones and tablets, developed with closer attention to the voice/audio transmission quality. Indeed, we can see that accuracy on device **D** is the lowest one on average (48.66%) while device **A** reaches an accuracy of 72.45%.

The most difficult to classify are *airport* and *street pedestrian* classes, while the easiest to classify is *street traffic* with 80% acc. and 0.283 log loss on average for all the systems. Among the techniques used for increasing the generalization capabilities we can find residual normalization [18], domain adversarial training [23], and use of data augmentation techniques as performed in [22, 27].

4.5. Discussion

Residual Networks have been shown to be the most efficient regarding acoustic scene classification for complexity-constrained solutions. Quantization combined with sparsity techniques have kept the model complexity within the required limit. The solutions presented in DCASE2021 Challenge follow the trends from previous year, combining the best characteristics and techniques from both acoustic scene classification subtasks. It is proven that the use of data augmentation improves generalization, compensating device mismatch. However, the reported log loss for seen/unseen devices and cities, shows that there is room for improvement; e.g. the use of domain adaptation techniques, like adversarial training used in [23], is not sufficient to deal with mismatches, since they report the highest mismatch among the 10-best submissions, while the use of mixup techniques prove to be more efficient.

Other mechanisms with less direct impact on the model parameters can be applied during the training step, the so-called learning techniques. These algorithms focus on obtaining a more efficient model by training it differently. Among the submissions, half of the teams have made use of some version of these techniques, the most popular ones being the use of focal loss and knowledge distillation. Focal loss helps the model to pay attention to the more difficult samples during the training step. However, the use of knowledge distillation seems to be the more efficient one, considering the ranking of related solutions.

5. CONCLUSIONS AND FUTURE WORK

This paper presented the results of the DCASE 2021 Challenge Task 1A, Low-Complexity Acoustic Scene Classification with Multiple Devices. The task combines the need for robustness and generalization to multiple devices of such systems with the requirement for a low-complexity solution, bringing the research problem closer to real-world applications. The method for calculating the model complexity includes only the parameters of the network, with exceptions in the case of employing embeddings. However, the strict model size limit has rendered the use of embeddings impossible, as most currently available pretrained models are already too big for the imposed limit. The task has received a large number of submissions that brought into spotlight interesting techniques that combine the best performing methods from the point of view of robustness, like data augmentation, with methods directed towards obtaining light models, e.g., knowledge distillation, weights quantization, and sparsity. The popularity of the task shows that acoustic scene classification is still relevant for the audio community, and in particular, to the development of solutions applicable for real-life devices.

6. REFERENCES

[1] E. Benetos, D. Stowell, and M. D. Plumbley, *Approaches to Complex Sound Scene Analysis*. Cham: Springer International Publishing, 2018, pp. 215–242.

[2] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proc. of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop*, November 2018, pp. 9–13.

[3] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in DCASE 2020 challenge: generalization across devices and low complexity solutions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop*, Tokyo, Japan, November 2020, pp. 56–60.

[4] M. Kośmider, "Calibrating neural networks for secondary recording devices," DCASE2019 Challenge, Tech. Rep., June 2019.

[5] W. Gao and M. McDonnell, "Acoustic scene classification using deep residual networks with late fusion of separated high and low frequency paths," DCASE2019 Challenge, Tech. Rep., June 2019.

[6] S. Suh, S. Park, Y. Jeong, and T. Lee, "Designing acoustic scene classification models with CNN variants," DCASE2020 Challenge, Tech. Rep., June 2020.

[7] S. Sigtia, A. M. Stark, S. Krstulović, and M. D. Plumbley, "Automatic environmental sound recognition: Performance versus computational cost," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 2096–2107, 2016.

[8] Y. Li, M. Liu, K. Drossos, and T. Virtanen, "Sound event detection via dilated convolutional recurrent neural networks," in *ICASSP 2020 - IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 286–290.

[9] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T.-J. Yang, and E. Choi, "Morphnet: Fast simple resource-constrained structure learning of deep networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1586–1595.

[10] K. Koutini, F. Henkel, H. Eghbal-zadeh, and G. Widmer, "CP-JKU submissions to DCASE'20: Low-complexity cross-device acoustic scene classification with rf-regularized CNNs," DCASE2020 Challenge, Tech. Rep., June 2020.

[11] T. Heittola, A. Mesaros, and T. Virtanen, "TAU Urban Acoustic Scenes 2020 Mobile, Development dataset," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3819968

[12] ——, "TAU Urban Acoustic Scenes 2020 Mobile, Evaluation dataset," June 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3685828

[13] A. Mesaros, T. Heittola, and T. Virtanen, "Acoustic scene classification in DCASE 2019 challenge: closed and open set classification and data mismatch setups," in *Proc. of the DCASE 2019 Workshop*, New York, Nov 2019.

[14] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135.

[15] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen and learn more: Design choices for deep audio embeddings," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019, pp. 3852–3856.

[16] S. Kumari, D. Roy, M. Cartwright, J. P. Bello, and A. Arora, "EdgeL3: Compressing L3-net for mote scale urban noise monitoring," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2019, pp. 877–884.

[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[18] B. Kim, Y. Seunghan, K. Jangho, and C. Simyung, "QTI submission to DCASE 2021: Residual normalization for device-imbalanced acoustic scene classification with efficient design," DCASE2021 Challenge, Tech. Rep., June 2021.

[19] A. Mesaros, A. Diment, B. Elizalde, T. Heittola, E. Vincent, B. Raj, and T. Virtanen, "Sound event detection in the dcase 2017 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 6, pp. 992–1006, 2019.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[21] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *36th Int. Conf. on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 6105–6114.

[22] C.-H. H. Yang, H. Hu, S. M. Siniscalchi, Q. Wang, W. Yuyang, X. Xia, Y. Zhao, Y. Wu, Y. Wang, J. Du, and C.-H. Lee, "A lottery ticket hypothesis framework for low-complexity device-robust neural acoustic scene classification," DCASE2021 Challenge, Tech. Rep., June 2021.

[23] K. Koutini, S. Jan, and G. Widmer, "Cpjku submission to dcase21: Cross-device audio scene classification with wide sparse frequency-damped CNNs," DCASE2021 Challenge, Tech. Rep., June 2021.

[24] Y. Liu, J. Liang, L. Zhao, J. Liu, K. Zhao, W. Liu, L. Zhang, T. Xu, and C. Shi, "DCASE 2021 task 1 subtask a: Low-complexity acoustic scene classification," DCASE2021 Challenge, Tech. Rep., June 2021.

[25] L. Pham, A. Schindler, H. Tang, and T. Hoang, "DCASE 2021 task 1A: Technique report," DCASE2021 Challenge, Tech. Rep., June 2021.

[26] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks." in *ICLR*. OpenReview.net, 2019.

[27] H. Hee-Soo, J. Jee-weon, S. Hye-jin, and L. Bong-Jin, "Clova submission for the DCASE 2021 challenge: Acoustic scene classification using light architectures and device augmentation," DCASE2021 Challenge, Tech. Rep., June 2021.

DIVERSITY AND BIAS IN AUDIO CAPTIONING DATASETS

Irene Martín-Morató, Annamaria Mesaros

Computing Sciences
Tampere University, Finland
{irene.martinmorato, annamaria.mesaros}@tuni.fi

ABSTRACT

Describing soundscapes in sentences allows better understanding of the acoustic scene than a single label indicating the acoustic scene class or a set of audio tags indicating the sound events active in the audio clip. In addition, the richness of natural language allows a range of possible descriptions for the same acoustic scene. In this work, we address the diversity obtained when collecting descriptions of soundscapes using crowdsourcing. We study how much the collection of audio captions can be guided by the instructions given in the annotation task, by analysing the possible bias introduced by auxiliary information provided in the annotation process. Our study shows that even when hints are given with the audio content, different annotators describe the same soundscape using different vocabulary. In automatic captioning, hints provided as audio tags represent grounding textual information that facilitates guiding the captioning output towards specific concepts. We also release a new dataset of audio captions and audio tags produced by multiple annotators for a subset of the TAU Urban Acoustic Scenes 2019 dataset, suitable for studying guided captioning.

Index Terms— audio captioning, bias, lexical diversity.

1. INTRODUCTION

Audio captioning is defined as the general audio content description using free-text [1]. As a free-text description of the content in terms of sound events in a soundscape, it is an important step in understanding the dynamics of a sound scene. Most environmental sound datasets (e.g. AudioSet [2], FSD50K [3], TAU Urban Acoustic Scenes [4]) are annotated with one or multiple labels or tags, providing only basic information on the content, and lack information on more intricate relationships e.g., how sounds overlap or follow each other, and other specific attributes. On the other hand, audio captioning (manual or automatic) has the potential to provide rich descriptions of audio content for various needs.

Image captioning has been an active research area for long, and has established certain practices for data collection and for evaluation of automatic methods, that are currently adopted as such in audio captioning. Often, Amazon Mechanical Turk (MTurk) was used to collect large amount of annotated data. Image captioning datasets like PASCAL [5], or Flickr8k [6] also highlight the main problems of using MTurk to collect annotations, such as grammar and spelling mistakes or empty annotations. Nevertheless, MTurk remains the method of choice for efficient and fast data collection.

The amount of audio captioning datasets and related work in audio captioning is very small in comparison to the vast amounts of data and related scientific literature available for image and video captioning. The few existing datasets for audio captioning include AudioCaps [7], a large-scale dataset containing 50K audio files, most files having one human-written description, and Clotho [8], a dataset of 5K clips, each having five human-written descriptions.

We argue that data collection is always prone to bias, being affected by how the annotation task is presented and what kind of instructions, examples, and auxiliary information is provided to the annotator. Moreover, perception of sounds is affected by other co-occurring and overlapping sounds [9]. On one hand, this can lead to a diverse set of free-form descriptions, if the clips to be captioned contain many sounds, because different annotators may choose to describe different sounds. On the other hand, an observation from automatic image captioning is that models do not have the capability of taking into account user interest: when the image to be described is complex, the models produce global descriptions that try to balance the information from the perspective of readability and informativeness [10]. This has lead to studies of diversity of automatic image descriptions [11], and novel methods for guiding the captioning by using a guiding text that refers to either groundable or ungroundable concepts in the image [10].

In this work, we study how human-produced audio captions are affected by bias introduced through auxiliary information during the annotation process. We investigate the lexical diversity of three audio captioning datasets, to determine how the possible bias affects the vocabulary and similarity of the free-text descriptions provided to the same clip by different annotators. The main contributions of this paper are twofold. Firstly, we observe that human annotators can be guided towards describing target content in audio clips without explicit instructions, and without affecting the richness of the language used in the descriptions. Secondly, we release a new crowdsourced dataset of captioned acoustic scene clips and corresponding audio tags, together with the annotator competence estimated based on the tags [12]. The captions provide an extension to the TAU Urban Acoustic Scenes 2018 dataset, and allow using it for automated guided captioning based on the tags as grounding text, while the estimated annotator competence offers a measure of trust in the individual annotations.

The paper is organized as follows: Section 2 describes how the collection of free-form descriptions for acoustic scene audio clips was set up and post-processed. Section 3 explains how we measure the vocabulary bias, the lexical diversity and the similarity of the captions. Section 4 shows the results of the analysis; finally, Section 5 presents conclusions and future work.

2. DATASETS FOR AUDIO CAPTIONING

We collected captions for a subset of TAU Urban Acoustic Scenes 2019 [4], through a process designed such that human annotators

This paper has received funding from Academy of Finland grant 332063 "Teaching machines to listen".

were given hints on the audio content. The comparative analysis of three datasets allows understanding how the diversity and the vocabulary of the captions is influenced by the annotation setup.

2.1. MACS: Multi-Annotator Captioned Soundscapes

The data to be annotated consists of recordings from three acoustic scenes (airport, public square and park) of the TAU Urban Acoustic Scenes 2019 development dataset. A number of 3930 files were annotated, each file being 10-seconds long. The 133 annotators, students taking an audio signal processing course, were randomly assigned a maximum of 131 files each. Annotators were assigned into 30 groups, aiming that each group will provide annotations to the same set of files.

The annotation procedure used a web-based interface, and annotators were given examples of correct annotations before they started. The annotation process consisted in two tasks. The annotator was provided with a list of ten classes and an audio clip that could be played back multiple times, and was required to first select the sounds present in the audio clip from the given list. Afterwards, the annotator was required to write a free-form one sentence description of the clip, using a minimum of 5 words. The sound labels provided were: *birds singing, dog barking, adults talking, children voices, traffic noise, music, footsteps, siren, announcement speech* and *announcement jingle*. The instructions neutrally mentioned that using these sounds in the free-form description is fine. We hypothesize that by giving annotators a tagging task and a preselected list of sounds, we bring to their attention certain content, and therefore influence the produced caption without explicitly mentioning on what content to focus on. The produced captions were then processed by removing punctuation (!., :; ?()−), replacing symbols and numbers by their non-numerical form (e.g. "1″ to "one″, "+″ to "and″) and correcting minor grammar mistakes (using Ginger Software through *gingerit*).

We publish the complete dataset, which we call MACS[1], consisting of the captions and tags assigned by each annotator to each of the files, and the estimated competence for each annotator. Annotator competence is calculated using multi-annotator competence estimation (MACE) [13] as described in [12] as a measure of trustworthiness of the individual annotations.

2.2. Other audio captioning datasets

AudioCaps [7], is a collection of sentence-long descriptions for a subset of AudioSet [2], focused on the audio input. The video was provided to be played if necessary, and the AudioSet tags were presented to the annotator as hints. The dataset contains over 46k files of 10 seconds each, and one caption per file, collected using MTurk. We consider that the tags given as hints and the video, if played, introduced some bias to the content described by the captions. Clotho [1] was also collected using MTurk using a three-step framework composed of captioning, grammar correction, and rating of the captions [8]. It contains five captions per clip, for audio clips 15 to 30 seconds long that were collected from Freesound [14]. We consider this dataset as having no bias, since the captions are based solely on the audio clip provided, and no additional information regarding the possible active sounds or clip content was available to annotators.

MACS contains audio recorded in the wild, which compared to Clotho may have more complex acoustic content. Freesound samples are typically highly representative of the tagged sound and often contain only the indicated sound without much background [15, p.51]. On the other hand, the clips in MACS and AudioCaps may contain uncontrolled sequences or co-occurrence of multiple sounds, as they happened naturally in the recorded environment.

[1]MACS dataset: https://zenodo.org/record/5114771

whistling footsteps and adults talking	5 words
adults talking and someone whistling	5 words
adults talk and whistle outside	5 words
people talking followed by footsteps and whistling	7 words
adults chattering and whistling nearby	5 words
total number of words: 27; unique words: 14	**TTR = 0.51**

Table 1: TTR, which represents the ratio of unique words with respect to the total number of words

3. DIVERSITY, BIAS AND SIMILARITY

This section presents an overview of the metrics we use for assessing diversity and evaluating similarity of the captions. There is no clear consensus on metrics regarding similarity of text; however, we employ a few metrics inspired from machine translation, automatic captioning, and natural language processing, which are most often used to benchmark certain vocabulary characteristics.

3.1. Lexical diversity

One simple measure that represents the variety in vocabulary, or lexical diversity, is the type-token ratio (TTR). TTR is often used in measuring language acquisition in infants or learners of a second language, to assess if the learner uses the same words over and over, or uses a variety of different words to communicate [16].

TTR is defined as the number of distinct words (tokens), divided by the total number of words. Therefore, it ranges from a theoretical 0 (infinite repetition of a single word) and 1 (no repetition at all). In practice, the value is influenced by the length of the analyzed text: the longer the analyzed text, the lower the calculated TTR, because of using more of the same words. Moving-Average-TTR (MATTR) [17] was proposed to remove text length dependency; however it is dependent on the window length, being equivalent to calculating TTR for a smaller fixed window size. An example for calculating TTR is presented in Table 1, using five descriptions assigned to the same audio file. We use TTR to have a simple understanding of the use of different words in the datasets under study.

3.2. Vocabulary bias

We propose to measure the vocabulary bias as the proportion of hinted sounds with respect to the number of sounds mentioned in the caption. For identifying sounds in the caption, we use the AudioSet taxonomy, consisting of approximately 600 classes, considering that it provides a comprehensive list of the most common sounds encountered in our everyday environments.

We analyze only AudioCaps and MACS for bias, because they were provided with hints during the annotation process. For AudioCaps, the hints are the tags associated to the clip in AudioSet (possibly incorrect). For MACS, the hints are the ten tags among which the annotator was asked to mark the sounds present in the clip. For example, "whistling footsteps and adults talking" contains three sounds (whistling, footsteps and talking) of which two (foot-

Captions	Jaccard	BLEU-4	BERTscore	sBERT
a person whistling and singing *people are talking and singing*	0.38	0.00	0.92	0.91

Table 2: Example of similarity metrics calculated for two captions of the same audio clip.

steps, talking) were given as hint, therefore the bias is 0.66, while for "adults talking and someone whistling" the bias is 0.5.

3.3. Similarity

Automatic captioning methods are evaluated using metrics from machine translation, to compare the machine-generated captions with human produced free descriptions of the same items. In this study, we are interested to evaluate the similarity of descriptions produced by different annotators for the same audio example. One basic approach to calculate similarity is the Jaccard similarity coefficient, or intersection-over-union, for two sets that are compared. For two sentences a and b, Jaccard index is defined as

$$J(a,b) = \frac{|S_a \cap S_b|}{|S_a \cup S_b|}, \tag{1}$$

where $S_{a/b}$ is the tokenized version of the sentence. $J(a,b) = 0$ means that sentences a and b do not have any token in common, while $J(a,b) = 1$ means that they contain the exact same tokens. Jaccard index is a fast low-cost metric for measuring similarity [8].

BLEU [18] is a commonly-used metric for comparing machine translated text to human-translated references. It does so by calculating the overlap between n-grams from the reference and candidate sentences. BLEU is defined as the geometric mean of the n-gram precision up to a certain length of n:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \tag{2}$$

where p_n is the modified n-gram precision multiplied by positive weights w_n, and BP is a brevity penalty applied when the generated text is too short. Most commonly reported is BLEU-4 (or cumulative 4-gram BLEU score), that incorporates 1-, 2-, 3-, and 4-grams, with a weight of 0.25 each. Because it measures overlap of n-grams, BLEU cannot handle synonyms and paraphrasing. Despite this, it is the most widely used automatic evaluation score in machine translation, and commonly reported in automatic captioning. Recent methods for calculating similarity in natural language processing use BERT [19], a model pretrained on large amounts of unlabeled data that can be fine-tuned with smaller amounts of labeled data. Building on BERT, BERTScore [20] calculates contextual embeddings to represent the tokens and computes matching using cosine similarity, optionally weighted with inverse document frequency scores. The BERT contextual embeddings can handle paraphrasing and different ordering, capturing distant dependencies in sentences. A slightly different approach is given by sentence-BERT (sBERT) [21], a modification of the BERT model using siamese networks [22]; s-BERT encodes an entire sentence into an embedding, instead of going token by token, then uses the cosine measure between the embedding vectors of two sentences.

The three selected measures represent similarity at different granularity: Jaccard index treats the tokens as a set, disregarding the order of words; BLEU looks at n-gram overlaps, therefore very specifically focuses on the ordering of words, while BERTscore and sBERT are state-of-the-art similarity measures that give a holistic view of the semantic content. Table 2 presents an example of metrics values for two captions corresponding to the same audio file in MACS. The scores produced by the BERT-based models (0.91 and 0.92) reflect the fact that there is a high similarity in the content of the two sentences; the Jaccard score of 0.38 shows the proportion of identical words within the vocabulary, while BLEU-4 has difficulties in matching n-grams, returning a score of 0.0.

Dataset	Audio clips	Vocab. size	Unique sentences	Sentence length (std)
AudioCaps	57188	5218	52198	9.17 (4.27)
Clotho	5929	4373	29611	11.34 (2.78)
MACS	3930	2775	16262	9.46 (3.89)

Table 3: Statistics of the studied datasets.

4. EXPERIMENTAL RESULTS AND ANALYSIS

The statistics of the three studied datasets are presented in Table 3, with the vocabulary calculated without lemmatization. Note that, these numbers correspond to the current version of the downloaded datasets, and may differ from the ones reported in the original paper. The most used words in the MACS dataset (after lemmatization and stop word removal) are *talk*, *people* and *adult*, *noise*, and *bird*, of which the first three are parts of the provided tags. *Speak* is used in all its forms, but in a considerable less amount, since the given tag was *adults talking*. In AudioCaps the most used five words are: *man*, *speak*, *follow*, *talk*, and *engine*. In contrast, in Clotho the clips were selected specifically to not include speech [1], and the most used 5 words are: *bird*, *water*, *background*, *chirp*, and *someone*.

4.1. Lexical diversity

Lexical diversity is calculated in three different versions: (1) without any processing of the text; (2) with lemmatization; and (3) with lemmatization and removal of stopwords. Overall lexical diversity, calculated as TTR using all captions in each dataset, is presented in Table 4. TTR is lower when lemmatization is performed than without any processing because lemmatization merges some forms into the same unique word, decreasing the number of types. When stopwords removal is added, TTR is slightly higher because a significant amount of repetitive words is removed from the overall text. The overall lexical diversity is very low for all datasets, implying that, for all of them, a small set of words is used repeatedly to describe the audio. While the vocabulary of AudioCaps is larger than the other datasets, the total amount of text in it is also larger, resulting in a small TTR. If MATTR is calculated instead, overall diversity values increase when using a small window. AudioCaps has the highest diversity when MATTR is calculated using a relatively small window (10-1000 tokens), while for larger windows (5000-10000), Clotho is more diverse. In all cases, MACS has a smaller MATTR diversity, showing a high repetition of the vocabulary.

We also calculate local lexical diversity, i.e., TTR for the set of

S	L	AudioCaps	Clotho		MACS	
		overall	overall	local	overall	local
-	-	1.09%	1.30%	56.52%	1.80%	69.37%
-	✓	0.79%	0.91%	52.08%	1.38%	66.06%
✓	✓	1.27%	1.66%	60.43%	2.17%	71.02%

Table 4: Global and local lexical diversity of captions. S: removal of stopwords; L: lemmatization. AudioCaps has only a single caption per clip, thus we do not calculate local lexical diversity for it.

	Tag bias (std)	Word bias (std)
AudioCaps	0.33 (0.35)	0.35 (0.35)
MACS	0.38 (0.36)	0.49 (0.38)

Table 5: Calculated vocabulary bias.

descriptions assigned to the same clip. Here the types and tokens are counted based on the 5 captions of each clip, and the resulting clip-wise TTR values are averaged over the dataset. The results are presented in Table 4 for the datasets with multiple captions per clip. The comparison of local lexical diversity between Clotho and MACS shows that while Clotho has a larger vocabulary and slightly larger overall lexical diversity, MACS has a higher proportion of different words used to describe individual clips. The reason for this could be the source of audio clips: even though diverse in terms of sound categories, many Freesound clips often contain only the indicated sound, while the clips in MACS, being recorded in the wild, allow description of different details and sounds.

4.2. Vocabulary bias

We identify sound events present in the captions using the AudioSet vocabulary. We have merged our tags into the AudioSet vocabulary to deal with synonyms, e.g we added "talk" and "adult talk" to the vocabulary, because Audioset contains only the synonym "speech". We use a total of 722 labels to identify sounds in the captions. Table 5 shows the calculated bias for the two datasets with given hints. We also calculate word bias to account for the hints that do not match exact categories in AudioSet. About one third of the sounds mentioned in the captions are found in the given hints for both AudioCaps and MACS. On the other hand, for individual words, MACS has a much higher bias. Considering that we added our tags to the vocabulary, and that "adults talking" was the most frequently annotated tag in MACS, this confirms that the choice of words in the free-text description is influenced by the given hints. In addition, for MACS, the ratio of sounds selected by the annotators as tags but not mentioned in the caption is 29%. This means that, on average, over one fourth of the tags indicating sounds being active in a clip were not included in the free-form description. This can be explained by the complexity of the scenes, for which the caption is only a partial description of a complex acoustic content. The calculated bias for the guided annotation tasks is not considerably high, and it is interesting to note the tags missing from the caption. We hypothesize that the complexity of the acoustic scene can affect the diversity of the vocabulary more than it affects the observed bias. Indeed, a closer look at scene-wise lexical diversity shows that *airport* class has a local lexical diversity approximately 3 percent points higher than the park and street scenes, indicating that airport clips have a higher scene complexity than the other scenes in terms of events happening.

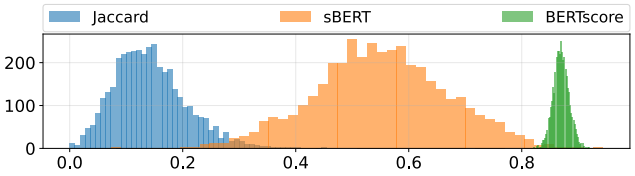


Figure 1: Similarity metrics for MACS dataset.

Dataset	BLEU-4	Jaccard	sBERT	BERTscore
Clotho	0.06 (0.04)	0.22 (0.09)	0.61 (0.13)	0.88 (0.01)
MACS	0.01 (0.02)	0.16 (0.08)	0.55 (0.12)	0.87 (0.01)

Table 6: Average similarity of the captions, using multiple metrics.

4.3. Similarity

We calculate similarity of the captions produced by different annotators for the same audio clip. The metrics are calculated for every pair of captions (10 pairs for a clip with 5 captions), and then averaged. Even though BLEU is generally meant to be used at corpus level, we use it at sentence level for comparison with the other metrics. The calculated values are presented in Table 6, and histogram plots of the clip-wise values for MACS are presented in Fig. 1.

We observe that BLEU provides very low similarity values, which implies diversity at least through ordering or paraphrasing. BLEU is higher for Clotho, and so is the Jaccard similarity index, indicating that descriptions of the same clip have more words in common for the captions in Clotho. This is in agreement with previously calculated local diversity that indicates MACS has more distinct words per clip. On the other hand, metrics based on BERT embeddings indicate high similarity for the descriptions of the same content. While sBERT is higher for Clotho, BERTscore is equally high. This effect may be due to the highly variable caption lengths in MACS, as sBERT groups sentences of same length for reducing computational load, and pads them to the longest one in each batch; according to the sentence length variance, this padding takes place more often in MACS than in Clotho. Both Clotho and MACS exhibit a high degree of caption similarity at clip level, irrespective of the difference in the characteristics of their audio content. On the other hand, the datasets do not have the same degree of diversity in terms of language used, showing its dependence on the nature of the complexity of the acoustic content.

5. CONCLUSIONS

This paper presented a study of the lexical diversity, bias, and similarity of captions from three audio captioning datasets. A new set of captions was collected for everyday soundscapes, with provided sound event hints. However, these hints turned out to not be a significant source of bias; instead, the free-text descriptions are more affected by the complexity of the soundscape. Despite the hints, the captions in the studied datasets have a high lexical diversity, and while token and n-gram based similarities are relatively low, the semantic similarity between captions assigned to the same clips by different annotators was found to be high. The new captions are freely available, along with the tags provided by the same annotators. This dataset brings novel elements to audio captioning; for example the tag-caption pairs allow guided captioning, and the estimated annotator reliability provides a measure of trustworthiness for each caption, which can be used in the learning process.

6. REFERENCES

[1] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: an audio captioning dataset," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 736–740.

[2] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[3] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: an open dataset of human-labeled sound events," *ArXiv*, vol. abs/2010.00475, 2020.

[4] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 9–13.

[5] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier, "Collecting image annotations using Amazon's Mechanical Turk," in *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Los Angeles: Association for Computational Linguistics, Jun. 2010, pp. 139–147.

[6] M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *J. Artif. Int. Res.*, vol. 47, no. 1, p. 853–899, May 2013.

[7] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the NAACL HLT, Vol. 1*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 119–132.

[8] S. Lipping, K. Drossos, and T. Virtanen, "Crowdsourcing a dataset of audio captions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, Oct. 2019.

[9] N. J. Vanderveeer, "Ecological acoustics: Human perception of environmental sounds," 1979.

[10] E. G. Ng, B. Pang, P. Sharma, and R. Soricut, "Understanding guided image captioning performance across domains," 2020, arXiv:2012.02339.

[11] E. van Miltenburg, D. Elliott, and P. Vossen, "Measuring the diversity of automatic image descriptions," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1730–1741.

[12] I. Martín-Morató and A. Mesaros, "What is the ground truth? reliability of multi-annotator data for audio tagging," in *29th European Signal Processing Conference 2021 (EUSIPCO 2021)*, Dublin, Ireland, Aug 2021.

[13] D. Hovy, T. Berg-Kirkpatrick, A. Vaswani, and E. Hovy, "Learning whom to trust with MACE," in *Proceedings of the 2013 Conference of the NAACL HLT*. Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 1120–1130.

[14] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *ACM International Conference on Multimedia (MM'13)*, ACM. Barcelona, Spain: ACM, Oct. 2013, pp. 411–412.

[15] N. Turpault, "Analysis of the scientific challenges in ambient sound recognition in real environment," Ph.D. dissertation, Universite de Lorraine, Villers-lès-Nancy, France, May 2021.

[16] G. Youmans, "Measuring lexical style and competence: The type-token vocabulary curve," *Style*, vol. 24, no. 4, pp. 584–599, 1990.

[17] M. A. Covington and J. D. McFall, "Cutting the gordian knot: The moving-average type–token ratio (mattr)," *Journal of Quantitative Linguistics*, vol. 17, no. 2, pp. 94–100, 2010.

[18] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 311–318.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the NAACL HLT, Vol. 1*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[20] T. Zhang*, V. Kishore*, F. Wu*, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," in *International Conference on Learning Representations*, 2020.

[21] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese bert-networks," in *EMNLP/IJCNLP (1)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 3980–3990.

[22] T. Ranasinghe, C. Orasan, and R. Mitkov, "Semantic textual similarity with Siamese neural networks," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd., Sep. 2019, pp. 1004–1011.

A MULTI-MODAL FUSION APPROACH FOR AUDIO-VISUAL SCENE CLASSIFICATION ENHANCED BY CLIP VARIANTS

Soichiro Okazaki, Quan Kong, Tomoaki Yoshinaga

Lumada Data Science Lab., Hitachi, Ltd.
{soichiro.okazaki.xs, quan.kong.xz, tomoaki.yoshinaga.xc}@hitachi.com

ABSTRACT

In this paper, we propose a system for audio-visual scene classification with a multi-modal ensemble way consisting of three features: (1) Log-mel spectrogram audio features extracted by CNN variants from audio modality. (2) Frame-wise image features extracted by CNN variants from video modality. (3) Another frame-wise image features extracted by OpenAI CLIP models which are trained with a large-scale web crawling text and paired image dataset under contrastive learning framework. We trained the above three models respectively and made an ensemble weighted by class-wise confidences of each model's semantic outputs. As a result, our ensemble system reached 0.149 log-loss (official baseline: 0.658 log-loss) and 96.1% accuracy (official baseline: 77.0% accuracy) on TAU Audio-Visual Urban Scenes 2021 dataset which are used in DCASE2021 Challenge Task1B.

Index Terms— Audio-visual Scene Classification, Multi-modal, CLIP, Convolutional Neural Network, Vision Transformer, Log-mel Spectrogram, SpecAugment, Random Erasing

1. INTRODUCTION

Audio-visual scene classification is one of the classification problems which uses both audio and video modalities for classifying the defined scene. Like human perception, we can expect to create a better model by exploiting complementary information from different modalities.

As recent research, several works tackle audio-visual joint learning. In [1, 2, 3, 4], the recognition performance of the audio-visual models is enhanced with a self-supervised manner using multi-modal information in each way. From action recognition perspectives, [5] proposed the Audiovisual SlowFast Networks, which utilize the SlowFast Networks [6] mainly used in video recognition tasks. In this research, audio features are concatenated to the image features through the model's internal pathway for multi-modal fusion. Other general multi-modal architectures are also used for audio-visual recognition tasks. [7] proposed the general perception architecture called Perceiver, which can treat image and audio features in the same way through concatenating the over 50 thousand dimension inputs. For leveraging audio-visual recognition performance using multi-modality information, various research has been proposed vigorously.

This year, Detection and Classification of Acoustic Scenes and Events (DCASE) challenge 2021 [8] holds the audio-visual scene classification task as Task1B [9] with a large-scale dataset called TAU Audio-Visual Urban Scenes 2021. This dataset provided by the organizer contains synchronized audio and video recordings from 12 European cities in 10 different scenes [10].

This paper describes the details of our team's (team name: LD-SLVision) solution for Task1B of DCASE2021. For this task, we developed various audio classification models and video classification models, and created final submissions by fusing those models using an ensemble method and a post-processing technique.

The features of our system can be concluded as three folds:

1) Instead of learning raw audio waves directly, we only used log-mel spectrogram features extracted from audio files as inputs, and leveraged those features with strong CNN variants which are used vigorously in the recent computer vision community.

2) We developed CLIP Late Fusion Network, which uses extracted features from various CLIP image encoders [11] as inputs for a multi-branch network. As far as we know, this is the first approach which uses CLIP models for audio-visual scene classification task.

3) We applied a post-processing technique to suppress the value of log-loss, which is defined as the competition's metric.

2. PROPOSED SOLUTION

In this section, we describe the details of our solution for DCASE2021 challenge Task1B. For tackling the audio-visual scene classification task, we created various audio classification models and video classification models respectively in each modality. After created various models, we integrated these models with ensemble method and applied post-processing technique to suppress the log-loss value for final submissions. The overview of our multi-modal fusion approach is shown in Fig. 1.

2.1. Audio Classification Models by Log-mel CNN Variants

For utilizing the audio modality of the provided dataset, we created various audio classification models with 1 second split audio files. The test dataset is provided as 1 second audio files in this competition. Therefore, we divided each 10 seconds audio files provided as development dataset for DCASE2021 Task1B into ten 1 second audio files.

As inputs for audio classification models, we extracted log-mel spectrograms with delta/delta-delta features, which are also used in DCASE2020 Task1A winner's solution [12]. We used librosa library [13] for creating log-mel spectrograms. The parameters of log-mel spectrogram transformation are as follows: sampling rate (sr) is 48kHz, the number of mel bins (n_mels) is 256, the length of FFT window (n_fft) is 4096, and the number of samples between successive frames (hop_length) is 512.

About the choice of inputs type (e.g. log-mel spectrogram, raw audio waveform, etc.), we referred to the PANNs paper [14] which proposed a widely used audio classification model. In the

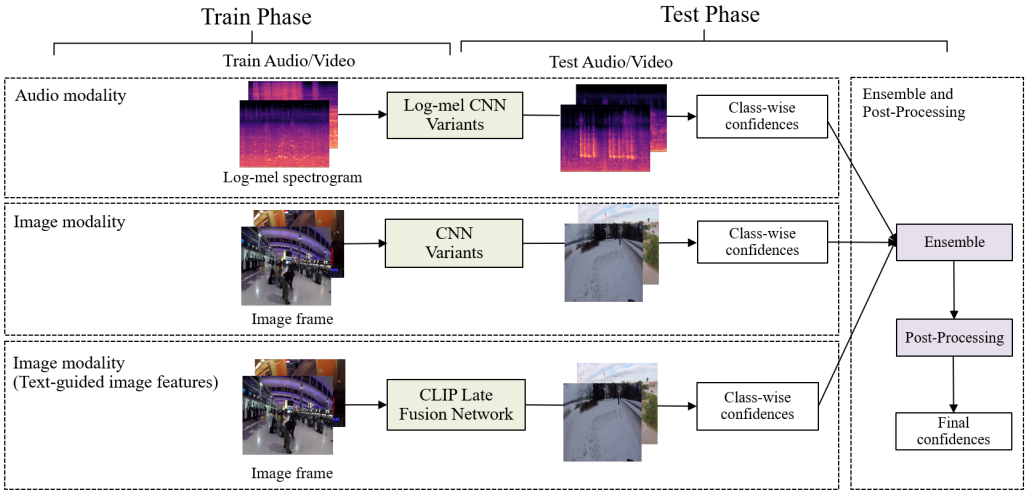


Figure 1: The overview of our system for DCASE2021 challenge Task1B. This picture shows the case of S02 in Table 3.

experiments of PANNs paper, the recognition performance of log-mel spectrogram classification with ResNet-38 [15] is competitive with that of wavegram log-mel CNN which is proposed as state-of-the-art architecture in the paper. Therefore, we only selected log-mel spectrogram features as inputs, and uses these features with strong CNN backbones (EfficientNet [16] with Noisy Student [17], ResNeSt [18], RegNet [19]) which are better than ResNet-38 in the recent computer vision community.

2.2. Video Classification Models by CNN Variants

For creating video classification models, we firstly extracted 12 image frames from 10 seconds video file with equal interval. After extracted all image frames, we created standard image classification models with strong CNN variants. We selected three backbones (ResNeSt [18], RegNet [19], HRNet [20]) for expecting different characteristics. These models have better recognition performances than ResNet in ImageNet classification task, and the combination of these models have achieved top accuracy in multi-label multi-class disaster scene classification task (LADI-only section) [21].

2.3. Video Classification Models by CLIP Late Fusion Network

For leveraging text modality and large amounts of publicly available data, we used CLIP image encoders which are trained with various web image and text caption pairs using contrastive learning method [11] [22]. With CLIP image/text encoders, we first conducted a zero-shot prediction on the provided TAU dataset. As a result, even without training, CLIP models achieved strong recognition performances which are competitive with our trained audio classification models as shown in Table 2.

For boosting the CLIP-based approach, we utilized CLIP models by adding a learnable multi-branch network, which we call CLIP Late Fusion Network. The architecture of CLIP Late Fusion Network is shown in Table 1. For this multi-branch network, we extracted image features from three types CLIP image encoders and feed these features as inputs to the network. In the SI-Score paper [23] which compares various CNN/ViT/CLIP models, the authors show that each CNN/ViT/CLIP models have different characteristics. Therefore, we selected to use not a single image encoder but multiple image encoders (i.e. ResNet50x4, ResNet101, Vit-B/32) for creating more diversity in input features.

Table 1: The architecture of CLIP Late Fusion Network.

RN50x4 (dim:640)	RN101 (dim:512)	ViT-B/32 (dim:512)
Linear(640, 512)	Linear(512, 512)	Linear(512, 512)
BatchNorm1d(512)	BatchNorm1d(512)	BatchNorm1d(512)
ReLU()	ReLU()	ReLU()
Dropout(p=0.2)	Dropout(p=0.2)	Dropout(p=0.2)
Linear(512, 256)	Linear(512, 256)	Linear(512, 256)
concatenation of 256*3 dimension		
Linear(256*3, 128)		
Linear(128, 10)		

2.4. Ensemble and Post-Processing

After created audio and video classification models, we used these models to output the confidences for each defined 10 scene classes. For the validation of the official fold1 split, we firstly inferred confidences for each 10 split audio files and 12 image frame files from each 10 seconds synchronized files. For each 10 seconds file, we equally made an ensemble of the output confidences of each split audio file and image frame file. In the ensemble process, audio classification models are generally worse than video classification models (Table 2). Due to the reason, we used the good accuracy's class score (e.g. In fold1 validation, the recognition performance of A04 for tram class is competitive with that of C04/V04.) Therefore, we used only bus/park/tram classes' confidence scores and discard the other classes in ensemble. In addition, for each sample, we replaced the confidences of video models with those of audio models when the maximum confidence of 10 classes from video models is 0.20 lower than that of audio models. This method improves the recognition performance on night scenes, to which video models have low confidences due to visual difficulty, but audio models can correctly classify the class (Fig. 2).

Table 2: Summary of our created models. In the baseline system, the Audio-only model is trained with 10 sec. audio files, but we trained our audio classification models with 1 sec. audio files as test audio files are provided as 1 sec. audio files. When we train our audio classification models with 10 sec. audio files, the recognition performances are more boosted than that of 1 sec. models. About CLIP models indexed as C01-C03, we provided original labels as sentences to the CLIP text encoders and evaluated the models with the CLIP image features.

Index	Architecture	Audio	Video	Notes	Logloss	Accuracy
B01	OpenL3's model	log-mel CNN	-	Baseline model of Audio-only	1.048	65.1
A01	RegNet-6.4F	log-mel CNN	-	Training with 1 sec. audio files	0.711	76.6
A02	ResNeSt-50d	log-mel CNN	-	Training with 1 sec. audio files	0.732	76.9
A03	TF-Efficientnet-B1-NS	log-mel CNN	-	Training with 1 sec. audio files	0.821	77.2
A04	A01-A03's models	log-mel CNN	-	Ensemble of A01-A03	0.721	78.1
B02	OpenL3's model	-	CNN	Baseline model of Visual-only	1.648	64.9
V01	RegNet-6.4F	-	CNN	-	0.328	90.0
V02	ResNeSt-50d	-	CNN	-	0.367	91.7
V03	HRNet-W18	-	CNN	-	0.336	90.9
V04	V01-V03's models	-	CNN	Ensemble of V01-V03	0.316	92.4
C01	ResNet-101	-	CLIP CNN	No Training	0.671	76.7
C02	ResNet-50x4	-	CLIP CNN	No Training	0.668	74.5
C03	ViT-B/32	-	CLIP ViT	No Training	0.725	72.5
C04	C01-C03's models	-	CLIP CNN&ViT	Late Fusion of C01-C03	0.273	90.9
B03	OpenL3's model	log-mel CNN	CNN	Baseline model of Audio-Visual	0.658	77.0
E01	A04/V04/C04's models	log-mel CNN	CNN / CLIP CNN&ViT	Ensemble of A04/V04/C04	0.238	95.8
E02	**A04/V04/C04's models**	**log-mel CNN**	**CNN / CLIP CNN&ViT**	**E01 with Post-Processing**	**0.149**	**96.1**

About post-processing, we applied the below (1) for replacing each models' output confidences. The idea behind this equation is as follows: For example, in the log-loss metric, when a sample belongs to class "tram" and the output confidence of the sample for class "tram" is a too small value (e.g. 0.00001), the log-loss value for this sample becomes large (i.e. -log(0.00001) = 11.51) and it will have a large negative impact on the calculation of whole log-loss value even with a few misrecognition. Therefore, to mitigate the whole log-loss error, we avoided the extreme confidence value (e.g. x = 0 $\sim$ 0.001 and 0.99 $\sim$ 1.0) by clamping with the small offset. Also, the samples which have low or high confidence scores (e.g. x = 0.001 $\sim$ 0.06, 0.06 $\sim$ 0.20, and 0.70 $\sim$ 0.99) almost always belong to the correct class as the confidence shows, however, the complete correction of the confidence values is difficult by only model learning processes. Therefore, to suppress the whole log-loss error, we introduced this confidence calibration approach to our system as post-processing. This approach is heuristic, but it significantly improved the log-loss results on the validation dataset and test dataset provided in DCASE2021 Task1B (Table 2 and 3).

$$f(x) = \begin{cases} 0.001, & when\ \ 0 < x \leq 0.06 \\ 0.06, & when\ \ 0.06 < x \leq 0.20 \\ x, & when\ \ 0.20 < x \leq 0.70 \\ 0.99, & when\ \ 0.70 < x \leq 1.0 \end{cases} \quad (1)$$

3. EXPERIMENTS

In this section, we present our experimental setting and results for both audio and video classification models.

Experimental setting for 2.1: We created audio classification models by log-mel CNN variants under the following setting: (1) Data augmentation: Resized to 256 × 100 × 3, Random Gain, Frequency Masking [24]. We did not use Mixup [25] and Time Warping/Masking [24] for our final submissions, as these augmentations did not work in our experimental setting. (2) Train batch size: 24 (3) Epoch: 20 (Best models' epoch are around 3-5 epoch.)

Experimental setting for 2.2: We created video classification models by CNN variants under the following setting: (1) Data augmentation: Resized to 448 × 448 × 3, RandomAffine, ColorJitter, GaussianBlur, Random Erasing [26]. In RandomAffine augmentation, we set degrees as [-10, 10], translate as (0.1, 0.1), and scale as (0.5, 1.5). In GaussianBlur augmentation, we set the kernel size as (11, 11). Other augmentations' parameters are the default ones of PyTorch [27]. (2) Train batch size: 20 (3) Epoch: 20 (Best models' epoch are around 15-20 epoch.)

Experimental setting for 2.3: We created video classification models by CLIP Late Fusion Network under the following setting: (1) Data augmentation: We used extracted features from CLIP image encoders and applied no data augmentation to these features in the late fusion network. (2) Train batch size: 48 (3) Epoch: 20 (Best models' epoch are around 3-5 epoch.)

Overall setting: In the above experiments, we used SGD with Momentum method [28] as the optimizer. The learning rate is divided by 10 when the training model reached 5 epoch, 10 epoch and 15 epoch. Other hyper-parameters are the same as used in IBN-Net [29] GitHub repository[*1]. We trained these models with Focal Loss [30] which γ parameter is 2.0. About pre-trained models, we used ImageNet pre-trained models from timm GitHub repository[*2] and CLIP pre-trained models from official CLIP GitHub repository[*3].

For ensemble as noted in Table 3, we used best epoch models in the validation accuracy of each model. Instead of using Test-Time Augmentation, we extracted five images from each test video by ffmpeg, and ensembled the confidences of the five images equally for each test video. In addition, all our models are trained and tested on 1 GPU (GeForce RTX 2080Ti).

Results: Table 2 shows the results for all of our models on

*1: https://github.com/XingangPan/IBN-Net
*2: https://github.com/rwightman/pytorch-image-models
*3: https://github.com/openai/CLIP

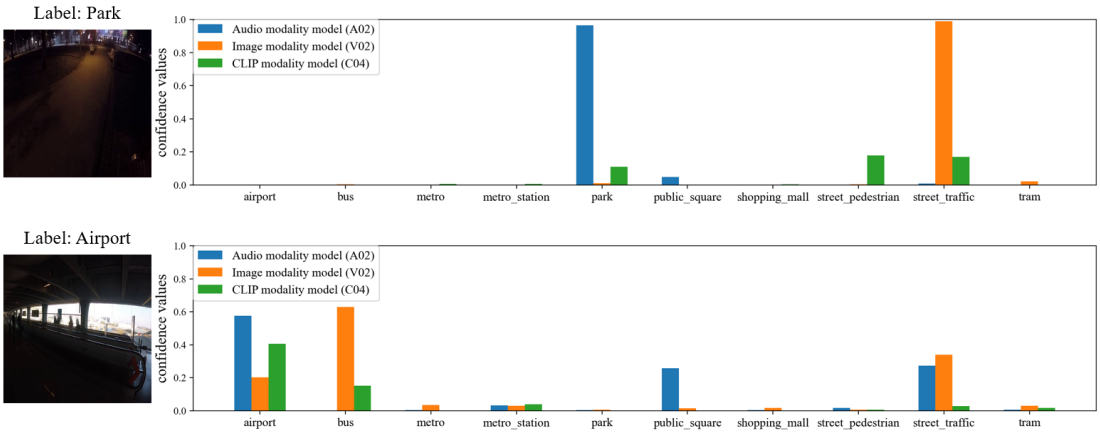


Figure 2: The sample inference results for dim scenes from validation dataset. For each scene, audio modality model (A02) can correctly classify the two dim scenes, however, image modality model (V02) failed to predict the true class. In addition, we can see that image modality model (V02) and CLIP modality model (C04) have different characteristics even trained with same image dataset of DCASE2021 Task1B.

the validation dataset. In this task, we found that CLIP models (C01-03) are competitive with the baseline models (B01-03) without training. In addition, our video classification models are much stronger than our audio classification models and we can classify 10-class scenes well only with our video classification models. In Table 3, we present the submission results for test dataset. The details are as follows: S01 consists of V04 and C04. Though C04 uses extracted features from C01-03, C04 model is constructed from one multi-branch network. Therefore, we counted the number of models in C04 as 1. S02 is the same as E02. S01-02 are trained and tested with official train/val split respectively. S03 consists of five E01 models. S04 consists of five E02 models. In Table 3, the number of models for each submission is denoted as "Models".

Table 3: Summary of our final submissions and the logloss scores for test dataset. p.p. in the description means post-processing which method is explained in 2.4 section. We created five train/val label files for creating S03-04 submissions. In the process of creating those five label files, we splitted the whole dataset into train and validation with keeping no overlapping about the location id.

Index	Description	Models	Logloss (Test)
S01	only-visual, 1fold	4	0.312
S02	audio-visual, 1fold, p.p.	7	0.320
S03	audio-visual, 5folds	35	0.303
S04	audio-visual, 5folds, p.p.	35	0.257

The effect of CLIP Late Fusion Network: Table 4 shows the effectiveness of our developed CLIP Late Fusion Network. Comparing the results for the case with and without CLIP, we can see that adding CLIP models can greatly improve the recognition performance of our audio-visual scene classification system, as CLIP models are created from different approach like leveraging text modality and large dataset. From model's complexity perspective, Table 5 compares the space/time complexity of CLIP Late fusion Network with other models. CLIP models have highly discriminative features, and the extracted image features from CLIP image encoders can be used directly as inputs for training shallow networks.

Table 4: The effect of CLIP Late Fusion Network (C04). With adding CLIP Late Fusion Network, the recognition performance are boosted in both logloss and accuracy metric for validation dataset.

Description	CLIP	Logloss	Accuracy
A04/V04 Fusion	no	0.293	92.4
A04/V04/C04 Fusion	**yes**	**0.238**	**95.8**
A04/V04 Fusion with p.p.	no	0.205	93.0
A04/V04/C04 Fusion with p.p.	**yes**	**0.149**	**96.1**

Table 5: The space/time complexity of CLIP Late Fusion Network (C04). Epoch means the each model's convergence epoch and Parameters means the number of learned parameters. Comparing with other models, CLIP Late Fusion Network is lightweight and can be quickly trained. The feature extraction time is not included in C04's epoch, as it can be ignored compared with training time.

Description	Epoch	Parameters	Logloss	Accuracy
V01	20	24.60M	0.328	90.0
V02	20	25.45M	0.367	91.7
V03	20	19.27M	0.336	90.9
V04	60	69.32M	0.316	92.4
C04	**5**	**1.35M**	**0.273**	**90.9**

4. CONCLUSION

In this paper, we described our approach for tackling the Task1B of the DCASE2021 challenge. We showed that by utilizing the features of CLIP variants with each audio classification models and video classification models, we can improve the recognition performance of the audio-visual scene classification task. In addition, we applied the post-processing method to the ensembled confidences, and our model achieved 0.149 log-loss (official baseline: 0.658 log-loss) and 96.1% accuracy (official baseline: 77.0% accuracy) on the officially provided fold1 validation dataset of Task1B.

5. REFERENCES

[1] A. Owens and A. A. Efros, "Audio-visual scene analysis with self-supervised multisensory features," in *IEEE European Conference on Computer Vision (ECCV)*, 2018.

[2] B. Korbar, D. Tran, and L. Torresani, "Cooperative learning of audio and video models from self-supervised synchronization," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[3] M. Patrick, Y. M. Asano, R. Fong, J. F. Henriques, G. Zweig, and A. Vedaldi, "Multi-modal self-supervision from generalized data transformations," in *arXiv preprint arXiv:2003.04298*, 2020.

[4] P. Morgado, N. Vasconcelos, and I. Misra, "Audio-visual instance discrimination with cross-modal agreement," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[5] F. Xiao, Y. J. Lee, K. Grauman, J. Malik, and C. Feichtenhofer, "Audiovisual slowfast networks for video recognition," in *arXiv preprint arXiv:2001.08740*, 2020.

[6] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[7] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General perception with iterative attention," in *International Conference on Machine Learning (ICML)*, 2021.

[8] http://dcase.community/challenge2021/.

[9] S. Wang, T. Heittola, A. Mesaros, and T. Virtanen, "Audio-visual scene classification: analysis of dcase 2021 challenge submissions," in *arXiv preprint arXiv:2105.13675*, 2021.

[10] S. Wang, A. Mesaros, T. Heittola, and T. Virtanen, "A curated dataset of urban scenes for audio-visual scene analysis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[11] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *arXiv preprint arXiv:2103.00020*, 2021.

[12] S. Suh, S. Park, Y. Jeong, and T. Lee, "Designing acoustic scene classification models with CNN variants," in *DCASE2020 Challenge Technical Paper*, 2020.

[13] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[14] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28. IEEE, 2020, pp. 2880–2894.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning (ICML)*, 2020.

[17] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[18] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Muller, R. Manmatha, M. Li, and A. Smola, "Resnest: Split-attention networks," in *arXiv preprint arXiv:2004.08955*, 2020.

[19] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-resolution representations for labeling pixels and regions," in *arXiv preprint arXiv:1904.04514*, 2019.

[21] S. Okazaki, Q. Kong, M. Klinkigt, and T. Yoshinaga, "Hitachi at trecvid dsdi 2020," in *TRECVID Notebook Paper*, 2020.

[22] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[23] J. Yung, R. Romijnders, A. Kolesnikov, L. Beyer, J. Djolonga, N. Houlsby, S. Gelly, M. Lucic, and X. Zhai, "Si-score: An image dataset for fine-grained analysis of robustness to object location, rotation and size," in *arXiv preprint arXiv:2104.04191*, 2021.

[24] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019.

[25] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations (ICLR)*, 2018.

[26] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Neural Information Processing Systems (NeurIPS)*, 2019.

[28] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *International Conference on Machine Learning (ICML)*, 2013.

[29] X. Pan, P. Luo, J. Shi, and X. Tang, "Two at once: Enhancing learning and generalization capacities via ibn-net," in *IEEE European Conference on Computer Vision (ECCV)*, 2018.

[30] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

ASSESSMENT OF SELF-ATTENTION ON LEARNED FEATURES FOR SOUND EVENT LOCALIZATION AND DETECTION

Parthasaarathy Sudarsanam, Archontis Politis, Konstantinos Drossos

Audio Research Group, Tampere University, Finland
{parthasaarathy.ariyakulamsudarsanam, archontis.politis, konstantinos.drossos}@tuni.fi

ABSTRACT

Joint sound event localization and detection (SELD) is an emerging audio signal processing task adding spatial dimensions to acoustic scene analysis and sound event detection. A popular approach to modeling SELD jointly is using convolutional recurrent neural network (CRNN) models, where CNNs learn high-level features from multi-channel audio input and the RNNs learn temporal relationships from these high-level features. However, RNNs have some drawbacks, such as a limited capability to model long temporal dependencies and slow training and inference times due to their sequential processing nature. Recently, a few SELD studies used multi-head self-attention (MHSA), among other innovations in their models. MHSA and the related transformer networks have shown state-of-the-art performance in various domains. While they can model long temporal dependencies, they can also be parallelized efficiently. In this paper, we study in detail the effect of MHSA on the SELD task. Specifically, we examined the effects of replacing the RNN blocks with self-attention layers. We studied the influence of stacking multiple self-attention blocks, using multiple attention heads in each self-attention block, and the effect of position embeddings and layer normalization. Evaluation on the DCASE 2021 SELD (task 3) development data set shows a significant improvement in all employed metrics compared to the baseline CRNN accompanying the task.

Index Terms— Sound event localization and detection, Self-attenion, acoustic scene analysis

1. INTRODUCTION

Sound event localization and detection (SELD) is a research problem associated with spatiotemporal analysis of acoustic scenes, providing temporal activity information of target sound classes along with their spatial directions or locations while they are active. The problem has seen increased research activity recently [1, 2], which culminated into the introduction of a new SELD task in the *Detection and Classification of Acoustic Scenes and Events* (DCASE) challenge in 2019, currently on its third iteration[1]. The task brings together two long-standing problems in acoustical signal processing: sound event detection (SED) aiming at only a temporal description of target sound classes in the scene, and sound source localization (SSL) aiming at detecting localized sound sources without regard to the type of the emitted sound events. Formulating and addressing the joint problem brings new possibilities in machine listening, robot audition, acoustical monitoring, human-machine interaction, and spatially informed deployment of services, among other applications.

[1] http://dcase.community/challenge2021/

The SELD task has been addressed in literature predominantly with deep learning models, with a few exceptions combining deep-learning SED classifiers with model-based localization [3, 4]. The seminal work of [1] proposed SELDnet, a model performing both SED and SSL tasks jointly, based on a convolutional and recurrent neural network (CRNN) architecture. SELDnet used a series of convolutional layers as feature extractors, operating on multi-channel spectrograms, followed by layers of gated recurrent unit (GRU) layers modeling longer temporal context. Such a CRNN architecture had proved successful in the SED task [5], and was extended in [1] with a localization inference output branch, predicting the frame-wise direction of arrival (DOA) of each detected class, in a regression manner. While alternative architectures have been explored (e.g. ResNets [6], TrellisNets [7], the R3Dnet of [8]), the CRNN architecture has remained the most popular through the submissions in DCASE2019 and DCASE2020. On the other hand, many innovations were network-independent, focusing on improved input features [9], separate modeling of SED and SSL tasks and fusion [9, 4], and improved SELD representations and loss functions [10, 8].

Recently, the Transformer [11] architecture has shown state-of-the-art performance in a variety of tasks ranging from NLP [11], to image classification [12] and video object tracking [13], among others, and has been proposed as a replacement for both CNNs and RNNs, or combined with convolutional layers in a Conformer [14] architecture. Transformers base their representational power on self-attention (SA) layers that can model longer temporal or spatial dependencies than typical convolutional layers, while, in contrast to RNNs, they can be efficiently parallelized making them significantly faster during inference. Recently transformers have shown strong state-of-the-art performance in SED tasks [15], while their use in SSL and SELD proposals has remained limited. Regarding source localization, Schymura et al. integrated self-attention into the outputs of the RNN layers in a CRNN model [16] showing performance gains over the standard CRNN. In subsequent work [17], RNNs are dropped for transformer layers including linear positional encoding, bringing further performance improvements. With regard to SELD, the first work using SA seems to be the DCASE2020 challenge submission of [10] which follows a SELDnet-like CRNN architecture, augmented with SA layers following the bidirectional RNN layers. The best performing team in DCASE2020 also seems to employ attention in the form of conformer blocks, as detailed in a later report [18]. Following DCASE2020, Cao et al. [19] proposed their Event Independent Network V2 (EINV2), realizing a track-based output format instead of the class-based one of standard SELDnet, using multi-head self-attention (MHSA) layers following convolutional feature extractors. Sinusoidal positional encoding is used before the MHSA as in [11]. Since the above SELD proposals

include various other improvements and modifications over the basic SELDnet CRNN, such as modified loss functions [10], partially independent models for SED and SSL with parameter sharing [19], or various data augmentation strategies [18], the effect of adding self-attention in isolation to the result is not clear.

In this work we exclusively investigate the effects of self-attention in a SELD setting. The rest of this paper is organized as follows. Section 2 presents our baseline method and the multi-head self-attention mechanism. In section 3, we describe in detail our experimental set up used to analyze the effect of self-attention. In section 4, we discuss the results of all our experiments. Finally, in section 5, we present our conclusion of this study.

2. METHOD

For our study, we employ a widely used SELD method that is based on a learnable feature extraction and a learnable temporal pattern identification, that operate in a serial fashion. We call this commonly used SELD method as our baseline. We replace the temporal pattern identification with a self-attention mechanism, that attends to the output of the learnable feature extraction layers.

The input to both the baseline and the version with the self-attention, is a tensor of K sequences of features from different audio channels, each sequence having T feature vectors with F features, $\mathbf{X} \in \mathbb{R}^{K \times T \times F}$. $\mathbf{X}$ is given as an input to the learnable feature extractor. For the baseline, the output of this feature extractor is used as an input to a function that performs temporal pattern identification, and the output of the temporal pattern identification is given as an input to a regressor. In the case of the method used for our study, the output of the learned feature extraction is given as an input to self-attention blocks, and then the output of the latter is given as an input to a regressor. The regressor in both cases predicts the directions-of-arrival for all classes and at each time step, represented by the directions of the output Cartesian vectors. Using the ACCDOA [8] representation, the detection activity is also integrated into the same vector representation, with the length of the vectors encoding the probability of each class being active. The output of the regressor and the targets are $\hat{\mathbf{Y}} \in \mathbb{R}^{T \times C \times 3}$ and $\mathbf{Y} \in \mathbb{R}^{T \times C \times 3}$ respectively, where C is the number of classes and 3 represents the Cartesian localization co-ordinates.

2.1. Baseline

As the baseline, we use the CRNN architecture proposed in [20], with ACCDOA representation for the output. The baseline has three convolutional neural network (CNN) blocks, CNNBlock_n with $n = 1, 2, 3$. CNNBlock_n acts as the learnable feature extractor, extracting high level representations from $\mathbf{X}$ as,

$$\mathbf{H}_n = \text{CNNBlock}_n(\mathbf{H}_{n-1}) \tag{1}$$

where $\mathbf{H}_n$ is the output of the n-th CNN block and $\mathbf{H}_0 = \mathbf{X}$. Each CNN block consists of a 2D convolution layer, a batch normalization process (BN), a rectified linear unit (ReLU), and a max pooling operation, and process its input as

$$\mathbf{H}_n = (\text{MP}_n \circ \text{ReLU} \circ \text{BN}_n \circ \text{2DCNN}_n)(\mathbf{H}_{n-1}) \tag{2}$$

where $\circ$ indicates function composition. BN_n and MP_n are the batch normalization and max-pooling processes of the n-th CNN block, and 2DCNN_n is the 2D convolution layer of the n-th CNN block. The output of the last CNN block is $\mathbf{H}_3 \in \mathbb{R}^{T' \times F'}$, where T' is the time resolution of the annotations and F' is the feature dimension down sampled from input dimension F in the CNNBlocks.

$\mathbf{H}_3$ is used as an input to a series of m recurrent neural networks (RNNs), with $m = 1, 2$ as

$$\mathbf{H}'_m = \text{RNN}_m(\mathbf{H}'_{m-1}) \tag{3}$$

where $\mathbf{H}'_m \in \mathbb{R}^{T' \times F''}$ is the output of the m-th RNN, where F'' is the hidden size of the RNN and $\mathbf{H}'_0 = \mathbf{H}_3$

The output of the RNN blocks is fed to a fully connected layer. The fully connected layer combines the learnt temporal relationships and it is followed by the regressor layer which predicts the detection and direction of arrival for all the classes for each time step in ACCDOA format.

$$\mathbf{y}' = \text{FC1}(\mathbf{H}'_2) \tag{4}$$

$$\hat{\mathbf{Y}} = \text{FC2}(\mathbf{y}') \tag{5}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{T' \times C \times 3}$ is the predicted ouput from the model.

2.2. ACCDOA representation

The annotations in the dataset for detections are of the form $\mathbf{Y}_{det} \in \mathbb{R}^{T' \times C}$, where T' is the number of time frames and C is the number of classes. For each time frame, the value is 1 for a class which is active, 0 otherwise. For localization, the labels are $\mathbf{Y}_{loc} \in \mathbb{R}^{T' \times C \times 3}$, which gives the 3 Cartesian localization co-ordinates for the classes in each time step that the classes are actrive.

The ACCDOA output representation simplifies these two labels into a single label $\mathbf{Y} \in \mathbb{R}^{T' \times C \times 3}$. In this representation, the detection probalility score is the magnitude of the predicted localization vector. This value is thresholded to predict the detection activity for each class. Thus the need for two different output branches to predict detection and localization separately becomes unnecessary.

2.3. Multi-head Self-Attention in SELD

The motivation of this study is to quantify the effect of replacing the RNN blocks in the baseline with self-attention blocks to capture the temporal relationships. In our experiments, the convolutional feature extractor is kept exactly the same as in the baseline architecture. The output $\mathbf{H_3}$ from the convolutional feature extractor is passed through a series of N self-attention blocks, with $N = 1, 2, ..$ as,

$$\mathbf{H}'_N = \text{SABlock}_N\{M, P, LN\}(\mathbf{H}'_{N-1}) \tag{6}$$

where $\mathbf{H}'_N \in \mathbb{R}^{T' \times F''}$ is the output of the N-th self-attention block, where F'' is the attention size and $\mathbf{H}'_0 = \mathbf{H}_3$.

In particular, we systematically study the effects of number of self-attention blocks (*N*), number of attention heads (*M*) in each self-attention block, positional embeddings (*P*)) for each time step and the effect of layer normalization (*LN*) on the detection and localization metrics.

The self-attention layer calculates the scaled dot-product attention [11] of each time step in the input with itself. For any input $\mathbf{H} \in \mathbb{R}^{T \times I}$, where T is the number of time steps and I is the input dimension, its self-attention is calculated as,

$$\text{SA}(\mathbf{H}) = \text{softmax}(\mathbf{H}\mathbf{W_q}\mathbf{W_k^T}\mathbf{H^T})\mathbf{H}\mathbf{W_v} \tag{7}$$

Here, $\mathbf{W_q}, \mathbf{W_k} \in \mathbb{R}^{I \times K}$ and $\mathbf{W_v} \in \mathbb{R}^{I \times O}$ are learnable query, key and value matrices respectively. K is the key dimension in the attention layer and O is the output dimension.

Table 1: Detection and localization results for different configurations of self-attention block on DCASE 2021 Development set. (* - Size of self-attention head in each layer)

N	M	P	LN	# params	ER_{20}	F_{20}	LE_{CD}	LR_{CD}
Baseline-CRNN				0.5 M	0.69	33.9	24.1	43.9
1	4	No	No	0.3 M	0.65 ± 0.01	38.11 ± 1.44	23.17 ± 0.85	46.73 ± 1.44
1	8	No	No	0.6 M	0.65 ± 0.01	39.12 ± 1.48	22.78 ± 0.73	46.71 ± 1.25
1	12	No	No	0.9 M	0.65 ± 0.01	38.96 ± 1.06	22.96 ± 0.88	46.74 ± 1.94
2	8	No	No	1.1 M	0.67 ± 0.01	36.95 ± 1.16	23.44 ± 1.27	44.66 ± 1.53
3	8	No	No	1.6 M	0.78 ± 0.02	19.57 ± 3.63	27.05 ± 0.90	22.96 ± 4.83
2	8	No	Yes	1.1 M	0.62 ± 0.01	44.62 ± 1.34	22.03 ± 0.66	55.04 ± 1.34
3	8	No	Yes	1.6 M	0.62 ± 0.01	44.11 ± 0.74	22.04 ± 0.53	54.61 ± 1.07
2	12	No	Yes	1.6 M	0.63 ± 0.01	43.95 ± 0.69	22.13 ± 0.36	54.23 ± 0.90
3	12	No	Yes	2.4 M	0.64 ± 0.01	43.10 ± 0.70	22.38 ± 0.54	54.00 ± 1.49
3 (128-256-128)*	8	No	Yes	2.2 M	0.63 ± 0.01	44.65 ± 1.88	21.98 ± 0.51	55.15 ± 1.47
3 (128-64-128)*	8	No	Yes	1.4 M	0.63 ± 0.01	43.64 ± 1.23	22.06 ± 0.46	54.24 ± 1.11
2	8	Yes	Yes	1.1 M	$\mathbf{0.61 \pm 0.01}$	$\mathbf{45.84 \pm 1.06}$	$\mathbf{21.51 \pm 0.74}$	54.99 ± 1.87
3	8	Yes	Yes	1.6 M	0.62 ± 0.01	44.63 ± 1.14	21.56 ± 0.46	54.46 ± 0.94
3 (128-256-128)*	8	Yes	Yes	2.2 M	0.62 ± 0.01	45.14 ± 1.03	21.67 ± 0.41	$\mathbf{55.29 \pm 1.23}$

First, we ran experiments to determine the optimal number of attention heads for the task. A single attention head allows each time step to attend only to one other time step in the input. For SELD task, it is useful to attend to more than one timestep to establish semantic relationships in the input audio scene. A multi-head self-attention (MHSA) layer is described as,

$$\text{MHSA}(\mathbf{H}) = \underset{m=1,2,..,M}{Concat}[\text{SA}_m(\mathbf{H})]\mathbf{W_p} \tag{8}$$

where M is the number of heads. The output from all the heads are concatenated and $\mathbf{W_p} \in \mathbb{R}^{MO \times O}$, a learnt projection matrix projects it into the desired output dimension.

Next, we studied the effect of stacking multi-head self-attention blocks. It enables the model to learn high level temporal features of different time scales. We also experimented with different ways to stack these MHSA blocks. Specifically, we compared the effect of having layer normalization (LN) and residual connections between successive blocks and not having both. The first multi-head self-attention layer takes as input the features from the CNN. The inputs to the successive layers of MHSA are given by,

$$\mathbf{H}_N = \text{LN}(\text{MHSA}_{(N-1)}(\mathbf{H}_{N-1}) + \mathbf{H}_{N-1}) \tag{9}$$

At last, we assessed the effect of having position embeddings in the self-attention block. Position embeddings are helpful in keeping track of the position and order of features that occur in an audio scene. This helps the model to learn temporal dependencies based on order of the sound events. Instead of using a sinusoidal position vector originally proposed in [11], since the data is split into chunks and the number of time steps is always fixed in our case, we used a fixed size learnable embedding table. If $P \in \mathbb{R}^{T \times I}$ is the position embedding, then the self-attention of input H with position embedding is calculated as $\text{SA}(H + P)$ in equation (7).

3. EVALUATION

3.1. Dataset

We trained and evaluated our models using the dataset provided for the DCASE 2021 sound event localization and detection challenge [21]. The development set contains 600 one-minute audio recordings with corresponding detections belonging to 12 different classes (alarm, crying baby, crash, barking dog, female scream, female speech, footsteps, knocking on door, male scream, male speech, ringing phone, piano) and their localization labels.

The multi-channel audio data is available in two recording formats, 4-channel first-order ambisonics (FOA) format and 4-channel tetrahedral microphone recordings (MIC) format. We used the 4-channel FOA recordings with a sampling rate of 24kHz. The audio recordings also contain realistic spatialization and reverberation effects from multiple multi-channel room impulse responses measured in 13 different rooms. The data is split into 6 folds of 100 recordings each. Folds 1-4 are used for training while 5 and 6 are used for validation and evaluation respectively.

3.2. Network Training

As described in section 2.3, we analysed the effect of different settings for the self-attention block. First, we replaced the two GRU layers in the baseline, with a single self-attention layer with 4 heads and an attention size of 128. This early result already suggested that using self-attention layers were beneficial compared to RNN layers. With the single layer self-attention, we then set the number of heads to 8 and 12 to evaluate the best hyper-parameter for the number of heads.

Next, we studied the effect of number of self-attention blocks. Specifically, we modified the architecture to have 2 and 3 attention blocks. For each of these configurations, we also varied the number of heads to be 8 and 12. The self-attention dimension was kept at 128 for all these experiments. When stacking self-attention blocks, we studied the effect of having and not having layer normalization and residual connections between sucessive blocks. In architectures having three self-attention blocks, we also studied the effect of the attention dimension in the multi-head self-attention blocks. In particular, we used 128-128-128, 128-256-128 and 128-64-128 configurations. Finally, we studied the effect of adding positional embedding vectors to the input of the first self-attention layer. We added learnable position embedding of vector size 128 to each time step

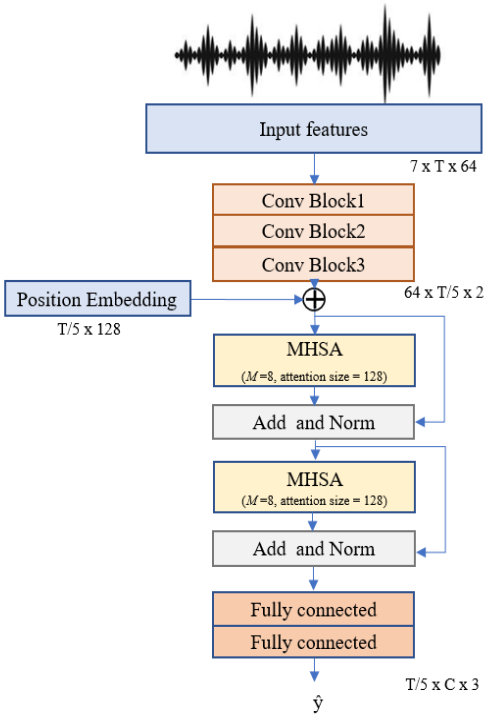


Figure 1: MHSA model configuration for SELD task.

in the input sequence to the self-attention.

For all our experiments, as input features, we extracted log mel spectrograms with 64 mel bins for each channel in the multi-channel audio. For the spectrogram extraction, we used short-time Fourier transform (STFT) with a Hann window, 50% overlap between frames and a hop length of 0.02 seconds. Further, we also calculated the intensity vectors [22] of the multi-channel audio signal from its linear spectra. The log mel spectrograms and the intensity vectors are concatenated along the channel dimension and fed as input to our model. The model is trained for 100 epochs using Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 0.001. We employed mean squared error as our objective function for this regression task and the model with the best validation score was chosen for evaluation.

The detection metrics are F score and error rate, they are also location-dependent, using a spatial threshold for true positives as detailed in [2]. Similar to DCASE2020, true positives occur only if events are localized within 20° from the ground truth of the same class. The localization metrics are localization error and localization recall and they are class dependent. For each setting, we train the model 10 times and report the average scores along with the standard deviation for each metric.

4. RESULTS

The results of all our experiments are summarized in Table 1. Our results from the first set of experiments for determining the appropriate number of attention heads showed that using 8 attention heads was marginally better than 12 heads when the number of attention blocks is fixed to one. Compared to the baseline, the detection error rate decreased from 0.69 to 0.65 and the F score increased from 33.9 to 39.12. There was also a decrease in the localization error from 24.1 to 22.78 and increase in the recall score from 43.9 to 46.71.

Our next set of analysis was to find the optimal number of self-attention blocks. Experimental results clearly demonstrate that serially connecting more self-attention blocks without layer normalization drastically reduces the performance of the model. Adding residual connections and layer normalization between the self-attention blocks significantly improves the performance of the model. We also verified that with multiple self-attention blocks, 8 attention heads was still the best performing configuration. With two self-attention blocks and 8 heads each, there was a steep increase in the F score to 44.62 and the localization recall jumped to 55.04.

Finally, we examined the importance of position embeddings to the first self-attention block and it proved to further increase the performance of our SELD system. From all our experiments, the best model configuration had two self-attention blocks with eight attention heads each with an attention dimension of 128, a learnt fixed size position embedding and residual connections with layer normalization between successive self-attention blocks. For this configuration, the detection error rate ER_{20} (lower the better), decreased by 11.6% and F-score F_{20} (higher the better), increased by 35.2% compared to the baseline. Similarly, the localization error rate LE_{CD}(lower the better) reduced by 10.7% and the localization recall LR_{CD} (higher the better) improved by 25.2% from the baseline. This model configuration is shown in Figure 1.

The best model configuration has close to twice the number of parameters as the baseline. However, due to the parallelization achieved by the self-attention blocks, it is also 2.5x faster than the baseline model during inference, based on our experiments on a V100 GPU. Hence, MHSA based models can be useful over RNN based models for real-time SELD tasks.

5. CONCLUSIONS

In this study, we systematically assessed the effect of self-attention layers for the joint task of sound event detection and localization. To account only for the impact of self-attention on this task, we employed the common SELDnet model using CRNN architecture and studied the effects of replacing the temporal pattern recognition RNN blocks with self-attention blocks. We experimented with various hyper parameter settings for the self-attention block such as number of blocks, number of attention heads in each self-attention block, size of the attention, layer normalization and residual connections between sucessive self-attention blocks and adding positional embedding to the input of self-attention block. Our experiments showed that, multi-head self-attention blocks with layer normalization and position embeddings significantly improve the F_{20} score and LR_{CD} score compared to the baseline. There is also a considerable decrease in the detection and localization error metrics compared to the baseline. The self-attention blocks also reduced the time required for training and inference compared to RNN blocks by exploiting parallel computations.

6. ACKNOWLEDGMENT

The authors wish to acknowledge CSC-IT Center for Science, Finland, for computational resources. K. Drossos has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, project MARVEL.

7. REFERENCES

[1] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 1, pp. 34–48, 2018.

[2] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen, "Overview and evaluation of sound event localization and detection in dcase 2019," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 684–698, 2020.

[3] A. Pérez-López and R. Ibánez-Usach, "Papafil: A low complexity sound event localization and detection method with parametric particle filtering and gradient boosting," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020), Tokyo, Japan*, 2020, pp. 155–159.

[4] T. N. T. Nguyen, D. L. Jones, and W.-S. Gan, "A sequence matching network for polyphonic sound event localization and detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 71–75.

[5] E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[6] Y. Cao, T. Iqbal, Q. Kong, Y. Zhong, W. Wang, and M. D. Plumbley, "Event-independent network for polyphonic sound event localization and detection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 11–15.

[7] S. Park, "Trellisnet-based architecture for sound event localization and detection with reassembly learning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, October 2019, pp. 179–183.

[8] K. Shimada, Y. Koyama, N. Takahashi, S. Takahashi, and Y. Mitsufuji, "Accdoa: Activity-coupled cartesian direction of arrival representation for sound event localization and detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, Ontario, Canada, June 2021.

[9] Y. Cao, Q. Kong, T. Iqbal, F. An, W. Wang, and M. Plumbley, "Polyphonic sound event detection and localization using a two-stage strategy," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, October 2019, pp. 30–34.

[10] H. Phan, L. Pham, P. Koch, N. Q. K. Duong, I. McLoughlin, and A. Mertins, "On multitask loss function for audio event detection and localization," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, 2020, pp. 160–164.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[13] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," *arXiv preprint arXiv:2101.02702*, 2021.

[14] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[15] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Convolution-augmented transformer for semisupervised sound event detection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, pp. 100–104.

[16] C. Schymura, T. Ochiai, M. Delcroix, K. Kinoshita, T. Nakatani, S. Araki, and D. Kolossa, "Exploiting attention-based sequence-to-sequence architectures for sound event localization," in *28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 231–235.

[17] C. Schymura, B. Bönninghoff, T. Ochiai, M. Delcroix, K. Kinoshita, T. Nakatani, S. Araki, and D. Kolossa, "Pilot: Introducing transformers for probabilistic sound event localization," *arXiv preprint arXiv:2106.03903*, 2021.

[18] Q. Wang, J. Du, H.-X. Wu, J. Pan, F. Ma, and C.-H. Lee, "A four-stage data augmentation approach to resnet-conformer based acoustic modeling for sound event localization and detection," *arXiv preprint arXiv:2101.02919*, 2021.

[19] Y. Cao, T. Iqbal, Q. Kong, F. An, W. Wang, and M. D. Plumbley, "An improved event-independent network for polyphonic sound event localization and detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 885–889.

[20] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 1, pp. 34–48, 2018.

[21] A. Politis, S. Adavanne, D. Krause, A. Deleforge, P. Srivastava, and T. Virtanen, "A dataset of dynamic reverberant sound scenes with directional interferers for sound event localization and detection," *arXiv preprint arXiv:2106.06999*, 2021. [Online]. Available: https://arxiv.org/abs/2106.06999

[22] V. Pulkki, A. Politis, M.-V. Laitinen, J. Vilkamo, and J. Ahonen, *First-Order Directional Audio Coding (DirAC)*, 2017, pp. 89–140.

MANY-TO-MANY AUDIO SPECTROGRAM TRANSFORMER: TRANSFORMER FOR SOUND EVENT LOCALIZATION AND DETECTION

Sooyoung Park, Youngho Jeong, Taejin Lee

Electronics and Telecommunications Research Institute, Media Coding Research Section, Daejeon, Republic of Korea, {sooyoung, yhcheong, tjlee}@etri.re.kr

ABSTRACT

Over the past few years, convolutional neural networks (CNNs) have been established as the core architecture for audio classification and detection. In particular, a hybrid model that combines a recurrent neural network or a self-attention mechanism with CNNs to deal with longer-range contexts has been widely used. Recently, Transformers, which are pure attention-based architectures, have achieved excellent performance in various fields, showing that CNNs are not essential. In this paper, we investigate the reliance on CNNs for sound event localization and detection by introducing the *Many-to-Many Audio Spectrogram Transformer* (M2M-AST), a pure attention-based architecture. We adopt multiple classification tokens in the Transformer architecture to easily handle various output resolutions. We empirically show that the proposed M2M-AST outperforms the conventional hybrid model on TAU-NIGENS Spatial Sound Events 2021 dataset.

Index Terms— Sound event localization and detection, self-attention, Transformer

1. INTRODUCTION

Convolutional neural networks (CNNs) have become essential for designing deep neural networks for image understanding tasks. The translation equivariance and locality of CNNs are known to be effective for image understanding. Due to the success of CNNs in image understanding, CNNs have also been used in other pattern recognition fields [1, 2]. Especially in audio understanding, CNNs have been applied to spectrogram images which are extracted from audio recordings by applying short-time Fourier transform to recognize image patterns. However, it is necessary to understand the longer context as well as the local context of the spectrogram in audio understanding fields. To understand this longer context, networks combining recurrent neural networks (RNNs) or self-attention with CNNs have been widely used [3, 4].

Self-attention mechanisms [5], especially Transformers, have become the new standard for natural language processing (NLP). The main approach of NLP is to fine-tune large pre-trained networks on small task-specific datasets. Transformers are well known for their computing efficiency and scalability. Using these Transformers, large models trained on large-scale text corpus datasets have been released [6]. These large models are known to extract generality from large amounts of training data. With the success of Transformers in NLP, Transformers are starting to be utilized in other fields [7, 8, 9, 10]. However, architectures combined with CNN rather than pure transformer architectures are mainly used.

Recently, *Vision Transformer* (ViT) [7, 8] using only pure Transformers for image understanding has been introduced. The outstanding performance of ViT is starting to question whether CNNs are still essential in many applications. Since then, research on Transformers replacing CNNs has become a trend in various fields. The *Keyword Transformer* (KWT) [9] and *Audio Spectrogram Transformer* (AST) [10] have been introduced as the first attempts to replace CNNs with Transformers in audio understanding. These studies demonstrate the potential of a pure Transformer to lower the dependence on CNNs in audio understanding. Inspired by the strength of the simple Transformer model in computer vision and audio classification, we propose an adaptation of this architecture to sound event localization and detection (SELD) [11].

In this paper, we propose a pure transformer architecture, *Many-to-Many Audio Spectrogram Transformer* (M2M-AST), for sound event localization and detection (SELD). M2M-AST enables efficient training of large models through transfer learning from large pre-trained models. AST provides one audio classification output for a single channel audio input (one-to-one). M2M-AST can have different resolution output sequences for multi-channel audio inputs (many-to-many).

2. RELATED WORK

2.1. Sound Event Localization and Detection

SELD [11] is the task of classifying multiple sound events with temporal activity into specific classes and detecting their directions. Therefore, SELD can be separated into two small tasks: sound event detection (SED) and direction of arrival estimation (DOAE). Specifically, SED is the task of classifying sound events into specified target classes to identify their onsets and offsets when sound events occur. DOAE is the task of detecting directions in which sound events occur in every frame. The DCASE challenge has published datasets for SELD since 2018. The TAU-NIGENS Spatial Sound Events 2020 dataset [12] consists of data that allow up to two simultaneous occurrences of sound events with directional activities. Additionally, up to three target sound events can occur simultaneously in the TAU-NIGENS Spatial Sound Events 2021 dataset [13]. Also, TAU-NIGENS Spatial Sound Events 2021 dataset is more difficult than TAU-NIGENS Spatial Sound Events 2020 dataset because there is background noise from unknown spatial acoustic events.

In SELD, the two-stage approach [14] and the joint modeling approach are dominant. The two-stage approach splits SELD into two models, SED and DOAE, and trains each separately. In the joint modeling approach, SED and DOAE are co-trained or integrated into a single system. Both methods commonly use convolutional recurrent neural networks (CRNNs) [4, 15, 16] or hybrid networks [17] that combine CNNs with self-attention layers. For CNNs, the output resolution depends on the pooling size. Therefore, SELD models using CNNs have limited output resolution by pooling size

and cannot freely construct output resolution depending on the application.

2.2. Self-Attention and Transformers

As pure self-attention-based networks, Transformers became the standard for NLP. Then, with the advent of ViT [7], the pure Transformer model expanded to the field of image understanding. ViT outperforms CNNs in image classification with self-attention computations between different image patches. However, ViT requires a significant amount of training data. To improve this, DeiT [8], which uses data augmentations and a knowledge distillation token to improve data efficiency, has been proposed. With the success of understanding images without CNNs, other research fields are also studying the reliance on CNNs.

KWT [9] and AST [10] are the first studies using a pure transformer in the field of audio understanding. These studies show that the pure Transformer models can replace CNNs in audio classification. In particular, KWT is a model that has adjusted the structure of the DeiT model for audio classification. AST is a model for efficiently training large-scale Transformer networks using ImageNet pre-trained models. The above studies are about a one-to-one structure that performs one classification on one audio recording. We propose an M2M-AST architecture that outputs sequences of varying resolutions from multi-channel audio recordings.

3. MANY-TO-MANY AUDIO SPECTROGRAM TRANSFORMER

3.1. Features

We use logmel and intensity vectors as input features [13] for SELD. The proposed SELD system is based on two-stage approach. The proposed SED network and DOAE network take different input features. The SED network uses logmel energy extracted from the microphone array data segmented into a single channel as input features. The DOAE network uses 7-channel inputs by extracting logmel and intensity vectors from Ambisonic data. This is summarized in Table 1. Table 2 shows the pre-processing parameters to extract input features

	Format	Feature	# Channels (C)	Label
SED	Microphone array	Logmel	1	Multi label binarization
DOAE	Ambisonic	Logmel, intensity vector	7	Cartesian coordinate (xyz)

Table 1: Feature and label configuration for SED and DOAE

Pre-processing	
Time window length	20 ms
Time window stride	10 ms
Frame length (T)	300 (3 sec)
# Mel-bins (M)	128

Table 2: Pre-processing parameters

3.2. Model Architecture

As shown in Figure 1, the proposed M2M-AST uses a Transformer encoder in the same way as AST [10]. M2M-AST uses only the encoder layer of the Transformer for classification and regression. Compared to AST, M2M-AST has differences in input feature and classification token configuration. Neural Networks for SELD extract multi-channel feature images from 4-channel audio recordings and use them as input features. Therefore, M2M-AST uses multi-channel feature images extracted from 4-channel audio recordings. Then we segment the extracted multi-channel feature images into the 16x16 patch sequence. At this point, we split the feature images by applying the same stride to the time and frequency dimensions. Afterwards, patch tokens are extracted through a linear projection for each patch. Like ViT [7, 8], the learnable classification token for classification is appended at the beginning of the patch token sequence. However, since SELD performs SED and DOAE every 100 ms, M2M-AST should output a series of outputs rather than a single output like AST. Therefore, patch embedding consists of appending a classification token sequence of equal length to the length of the output sequence at the beginning of the patch token sequence. The length of the classification token sequence determines the output resolution. For example, configuring an output resolution of 100 ms for 3 seconds input data would use a sequence of 30 classification tokens. On the other hand, configuring an output resolution of 20 ms for 3 seconds of input data would use a sequence of 150 classification tokens. The Transformer has no convolution or recurrence, so it cannot leverage the relative spatial information of the patch tokens in the 2D feature images. To take advantage of the position information of the patch tokens, we add a learnable positional embedding $\mathbf{p}_i (\in \mathbb{R}^d)$ to the patch embedding. The Transformer encoder's outputs of the classification token sequence learn the audio spectrogram representation by computing the self-attention between each patch token. Then, we use a dense layer with an activation layer for SED and DOAE from the Transformer encoder's output of classification tokens. M2M-AST's model parameters are the same as AST and are summarized in Table 3.

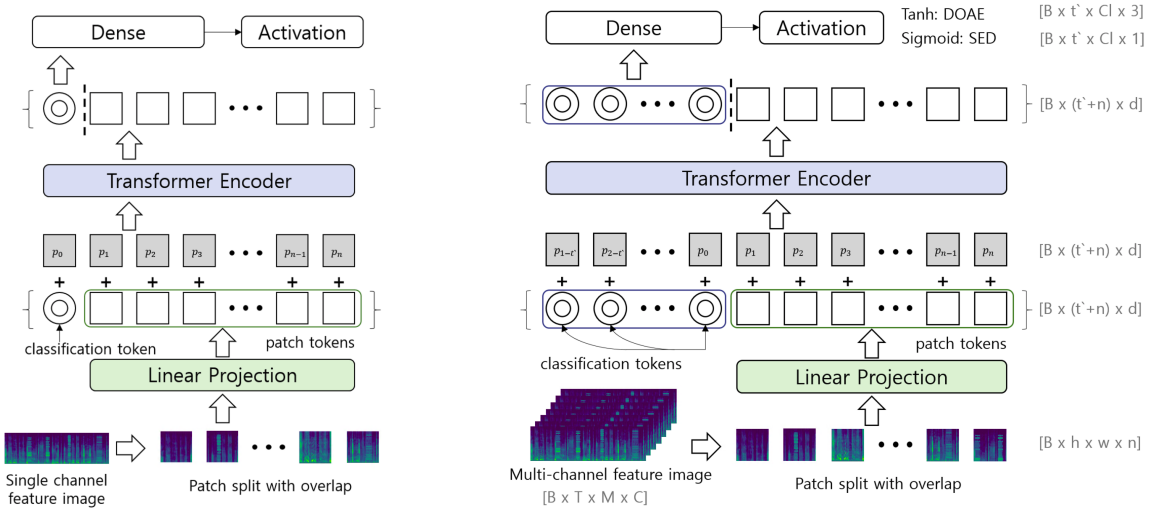
Model parameter	
Patch shape (h x w)	16x16
Patch overlap	6
# Patches (n)	348
Patch dimension (d)	768
# Encoder layer (L)	12
# Attention head	12
Output size (t')	20
Dropout	0.1

Table 3: Model parameters

3.3. Transfer Learning

M2M-AST uses only a Transformer encoder like AST [10]. M2M-AST uses the same Transformer encoders as ViT [7, 8]. ViT requires a large dataset for sufficient performance. To overcome this problem, DeiT [8] uses knowledge distillation. Unlike image datasets, audio datasets contain relatively small amounts of data. Therefore, AST uses transfer learning to distill knowledge from the ImageNet pre-trained model. M2M-AST uses transfer learning in the same way as AST. The weight can be easily transferred because the same Transformer encoder is used. However, the layer learning patch embeddings vary in size and require some adjustments.

DeiT uses 3-channel input images, while M2M-AST uses variable multi-channel input images. In M2M-AST, the weight corresponding to each channel in the linear projection layer uses the



(a) AST (b) Many-to-Many AST

Figure 1: Architecture of AST and Many-to-Many AST; B: batch size, T: time, M: # mel-bins, C: channel, t': output size, (h x w): patch shape, n: # patchs, d: patch dimension, Cl: # class

average weight of the three channels in DeiT. While DeiT has a fixed length of patch embedding sequence, M2M-AST has not fixed length of patch embedding sequence because the classification token sequence that determines the output resolution is variable. Therefore, the positional embeddings for the patch tokens in M2M-AST $[\mathbf{p}_1, \cdots, \mathbf{p}_n]$ are transferred as scaled values through cut and bilinear interpolation to map the relative positions of the positional embeddings in DeiT to the input feature. Individual positional embeddings of classification token sequence $[\mathbf{p}_{1-t'}, \cdots, \mathbf{p}_0]$ are equally initialized by the average value of classification token and distillation token in DeiT. This transfer learning method makes it easy to extract pre-trained network knowledge for ImageNet into the audio domain.

3.4. Post-processing

The input time window for our system is 3 seconds. We slide this window with a small hop size to create many overlapped results and average these results during the inference [15]. Additionally, we apply median filtering and tuning the threshold for each class during SED inference. Finally, we apply a 16-way rotation augmentation to infer the test data and average the values obtained by rotating the results in reverse [15, 16].

4. EXPERIMENTS

We provide experimental results on TAU-NIGENS Spatial Sound Events 2021 development dataset [13]. The development dataset consists of 600 1-minute wave files. We use 400 minutes of data for training, 100 minutes for validation, and the remaining 100 minutes for testing. Our system is trained using the hyper-parameters in Table 4. We use transfer learning with the pre-trained model. The pre-trained model used in our system is shown in Table 5. We fine-tune the SED model with 85M parameters and the DOAE model with 86M parameters for 50 epochs independently. We use the Adam optimizer. For the development dataset, the training time consumed by the M2M-AST is 4 hours for SED and 2 hours for DOAE at 4-TITAN Xp. The model mentioned in Table 5 is used for the experiment. The models mentioned in Table 5 are for each SED and DOAE task. Because of the large model size of M2M-AST, we use a two-stage approach rather than joint training for SELD.

Training	
Epoch	50
# Batch (B)	24
Learning rate	0.0001
Optimizer	Adam

Table 4: Hyper-parameters for proposed system

	Task	Pre-trained model	Loss
M2M-AST1	SED	DeiT	BCE
M2M-AST2	SED	M2M-AST1	soft f-loss [18, 19]
M2M-AST3	DOAE	DeiT	MSE
M2M-AST4	DOAE	M2M-AST3	masked MSE

Table 5: Model configuration

4.1. Results

Table 6 reports the results on the TAU-NIGENS Spatial Sound Events 2021 dataset [13]. All results are based on logmel energy and intensity vectors as input features. Baseline-Large is a model in which the filter size of the baseline is increased to be similar to the model size of M2M-AST. Using M2M-AST with the two-stage approach significantly improves the performance of all metrics over the baseline. In addition, the proposed pure Transformer model, M2M-AST, outperforms the CRNN-based models listed in Table 6, demonstrating that it is a sufficient replacement for CRNNs. Therefore, we show that self-attention computing within the Transformer on SELD, SED, and DOAE can reduce the reliance on CNNs.

	# Params	ER_{20°	F_{20°	LE_{CD}	LR_{CD}
CRNN (Baseline FOA)	0.5M	0.69	33.9 %	24.1°	43.9 %
CRNN (Baseline-Large)	184M	0.65	45.6 %	22.6°	55.0 %
CRNN [20]	14M	0.65	48.3 %	22.0°	62.6 %
M2M-AST1&3	172M	0.55	62.6 %	17.5°	74.0 %
M2M-AST1&4	172M	0.52	64.4 %	**16.0°**	74.0 %
M2M-AST2&3	172M	0.52	64.0 %	17.7°	**74.7 %**
M2M-AST2&4	172M	**0.50**	**65.7 %**	**16.3°**	**74.7 %**

Table 6: Experimental results for development dataset

4.2. Ablation Study

We perform a series of ablation studies to explain M2M-AST design choices. We conduct ablation studies based on M2M-AST1 and M2M-AST3 initialized with ImageNet pre-training models while using loss functions commonly used in SELD.

4.2.1. Batch size and frame length

We compare the performance of M2M-AST with different batch sizes and frame lengths of input features through grid search. Table 7 shows the results of this comparison. Performance for SED under the ideal DOAE condition is evaluated through the F_1 score and LR_{CD}(F_∞ score). F_1 score represents the balanced score of precision and recall. Besides LR_{CD} represents the recall dominant score. On the other hand, longer input frames improve both precision and calls. This is because longer input frames make M2M-AST use more patches for training. For DOAE, smaller batch sizes and longer input frame lengths improve performance.

	SED (F_1, LR_{CD})			DOAE (LE_{CD})		
# Batch	1 sec	2 sec	3 sec (Used)	1 sec	2 sec	3 sec (Used)
24 (Used)	(68.3, 66.3)	(75.0, 73.2)	(74.0, **74.0**)	26.3°	22.2°	**21.8°**
48	(69.5, 70.9)	(75.7, 72.1)	(75.2, 73.6)	27.9°	23.1°	23.0°
96	(70.7, 70.3)	(**75.8**, 68.7)	-	27.0°	24.4°	-

Table 7: Experimental results with different batch sizes and input frame lengths

4.2.2. Patch split with overlap

Table 8 shows a performance comparison with patch splits of 16x16 sizes using various sizes of strides. Configuring dense patch segmentation with large overlap helps both SED and DOAE improve performance. However, for SED, performance improvements converge on overlap size 6. Thus, exploiting patch splits with a larger overlap size than 6 leads to the burden of memory and computation cost.

	# Patches	SED (F_1, LR_{CD})	DOAE (LE_{CD})
No Overlap	144	(71.6, 60.2)	27.3°
Overap-2	189	(73.8, 68.6)	24.6°
Overap-4	240	(74.1, 70.6)	24.1°
Overap-6 (Used)	348	(74.0, **74.0**)	**21.8°**
Overap-8	540	(**74.9**, 72.5)	**21.0°**

Table 8: Experimental results with different lengths of patch overlap

4.2.3. Output resolution

M2M-AST can adjust the number of classification tokens to have a variety of output resolutions. Table 9 shows a performance comparison of M2M-AST with output resolution from 100 ms to 25 ms. Since the resolution of the ground truth data is 100 ms, we use the nearest-neighbor interpolation to construct labels with high resolution and use them for training. Then we apply a median filter to construct an output of 100 ms. For SED, smaller resolution results in slight performance gains due to median filtering. On the other hand, for DOAE, the results do not vary significantly with changes in output resolution.

Output resolution	Output size (t')	SED (F_1, LR_{CD})	DOAE (LE_{CD})
25 ms	120	(75.3, 73.8)	22.2°
33.3 ms	90	(76.5, 75.1)	22.1°
50 ms	60	(74.4, 72.8)	22.7°
100 ms (Used)	30	(74.0, 74.0)	21.8°

Table 9: Experimental results with different output resolutions

4.2.4. Pre-training and loss function

We compare the performance of randomly initialized M2M-AST and M2M-AST transferred from pre-trained models. As shown in Table 10, the weight transferred model from ImageNet pre-trained model outperforms the randomly initialized model in SED. On the other hand, transfer learning from ImageNet pre-trained model improves DOAE performance slightly. In addition, we compare M2M-AST using different loss functions while using a pre-trained model. In SED, soft f-loss [18, 19] is slightly better than binary cross-entropy (BCE), but there is no significant difference. On the other hand, with the DOAE pre-trained model, masked MSE improves performance by 2.7 degrees over BCE.

	Pre-trained model	Loss	SED (F_1, LR_{CD})	DOAE (LE_{CD})
No pre-train (SED)	-	BCE	(60.4, 54.5)	-
ImageNet pre-train (M2M-AST1)	DeiT	BCE	(74.0, 74.0)	-
SELD pre-train (M2M-AST2)	M2M-AST1	soft f-loss	(**75.8**, **74.7**)	-
No pre-train (DOAE)	-	MSE	-	22.5
ImageNet pre-train (M2M-AST3)	DeiT	MSE	-	21.8
SELD pre-train (M2M-AST4)	M2M-AST3	masked MSE	-	**19.1**

Table 10: Experimental results with different loss functions and pre-trained models

5. CONCLUSIONS

In this paper, we describe how to apply the standard Transformer architecture to SELD. As a consequence, we introduce M2M-AST, a pure Transformer model for SELD. Existing SELD networks have commonly used hybrid architectures that combine CNNs with RNNs or self-attention layers. We empirically show that M2M-AST can replace these hybrid networks in SELD, SED, and DOAE. The experimental results represent the potential of a pure Transformer to lower the reliance on CNNs in SELD. Traditional neural networks use pooling layers to change the output shape. However, due to the pooling size of this pooling layer, the output resolution cannot be configured freely. On the other hand, M2M-AST has the advantage of being able to easily design to have a variety of output resolutions.

6. ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00050, Development of Human Enhancement Technology for auditory and muscle support).

7. REFERENCES

[1] S. Suh, S. Park, Y. Jeong, and T. Lee, "Designing acoustic scene classification models with CNN variants," DCASE2020 Challenge, Tech. Rep., June 2020.

[2] R. Giri, S. V. Tenneti, K. Helwani, F. Cheng, U. Isik, and A. Krishnaswamy, "Unsupervised anomalous sound detection using self-supervised classification and group masked autoencoder for density estimation," DCASE2020 Challenge, Tech. Rep., July 2020.

[3] Q. Wang, H. Wu, Z. Jing, F. Ma, Y. Fang, Y. Wang, T. Chen, J. Pan, J. Du, and C.-H. Lee, "The ustc-iflytek system for sound event localization and detection of dcase2020 challenge," DCASE2020 Challenge, Tech. Rep., July 2020.

[4] Y. Cao, T. Iqbal, Q. Kong, Y. Zhong, W. Wang, and M. D. Plumbley, "Event-independent network for polyphonic sound event localization and detection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 11–15.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 6000–6010.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[8] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers amp; distillation through attention," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, 18–24 Jul 2021, pp. 10 347–10 357.

[9] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword transformer: A self-attention model for keyword spotting," *arXiv preprint arXiv:2104.00769*, 2021.

[10] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.

[11] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen, "Overview and evaluation of sound event localization and detection in dcase 2019," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 684–698, 2020.

[12] A. Politis, S. Adavanne, and T. Virtanen, "A dataset of reverberant spatial sound scenes with moving sources for sound event localization and detection," in *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, November 2020.

[13] A. Politis, S. Adavanne, D. Krause, A. Deleforge, P. Srivastava, and T. Virtanen, "A dataset of dynamic reverberant sound scenes with directional interferers for sound event localization and detection," *arXiv preprint arXiv:2106.06999*, 2021.

[14] Y. Cao, Q. Kong, T. Iqbal, F. An, W. Wang, and M. Plumbley, "Polyphonic sound event detection and localization using a two-stage strategy," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, October 2019, pp. 30–34.

[15] K. Shimada, N. Takahashi, S. Takahashi, and Y. Mitsufuji, "Sound event localization and detection using activity-coupled cartesian doa vector and rd3net," DCASE2020 Challenge, Tech. Rep., July 2020.

[16] L. Mazzon, M. Yasuda, Y. Koizumi, and N. Harada, "Sound event localization and detection using foa domain spatial augmentation," DCASE2019 Challenge, Tech. Rep., June 2019.

[17] Q. Wang, J. Du, H. Wu, J. Pan, F. Ma, and C.-H. Lee, "A four-stage data augmentation approach to resnet-conformer based acoustic modeling for sound event localization and detection," *arXiv preprint arXiv:2101.02919*, 2021.

[18] S. Park, Y. Jeong, and T. Lee, "Metric optimization for sound event localization and detection," in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, 2020, pp. 1–4.

[19] T. Tanaka and T. Shinozaki, "F-measure based end-to-end optimization of neural network keyword detectors," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2018, pp. 1456–1461.

[20] T. N. T. Nguyen, K. Watcharasupat, N. K. Nguyen, D. L. Jones, and W. S. Gan, "Dcase 2021 task 3: Spectrotemporally-aligned features for polyphonic sound event localization and detection," DCASE2021 Challenge, Tech. Rep., November 2021.

AN ENSEMBLE APPROACH TO ANOMALOUS SOUND DETECTION BASED ON CONFORMER-BASED AUTOENCODER AND BINARY CLASSIFIER INCORPORATED WITH METRIC LEARNING

Ibuki Kuroyanagi[1,2]*, Tomoki Hayashi*[1,2]*, Yusuke Adachi*[1,2]*,*
Takenori Yoshimura[1]*, Kazuya Takeda*[2]*, Tomoki Toda*[2]

[1] Human Dataware Lab. Co., Ltd., Nagoya, Japan
tomoki.hayashi@hdwlab.co.jp
[2] Nagoya University, Nagoya, Japan
kuroyanagi.ibuki@g.sp.m.is.nagoya-u.ac.jp

ABSTRACT

This paper presents an ensemble approach based on two unsupervised anomalous sound detection (ASD) methods for machine condition monitoring under domain-shifted conditions in DCASE 2021 challenge Task 2. The first ASD method is based on a conformer-based sequence-level autoencoder with section ID regression and a self-attention architecture. We utilize the data augmentation techniques such as SpecAugment to boost the performance and combine a simple scorer module for each section and each domain to address the domain shift problem. The second ASD method is based on a binary classification model using metric learning that uses task-irrelevant outliers as pseudo-anomalous data while controlling centroids of normal and outlier data in a feature space. As a countermeasure against the domain shift problem, we perform data augmentation based on Mixup with data from the target domain, resulting in a stable performance for each section. An ensemble approach is applied to each method, and the resulting two ensembled methods are further ensembled to maximize the ASD performance. The results of DCASE 2021 challenge Task 2 have demonstrated that our proposed method achieves a harmonic mean of 63.745% of area under the curve (AUC) and partial AUC ($p = 0.1$) over all machines, sections, and domains.

Index Terms— Anomalous sound detection, autoencoder, binary classification, metric learning

1. INTRODUCTION

Anomalous sound detection (ASD) is the task of detecting anomalous sounds caused by atypical events, such as the malfunction or breakdown of a machine. The detection of anomalous sounds can be used to improve the efficiency of maintenance work on manufacturing equipment and infrastructure, and to monitor equipment installed in locations which are difficult for people to enter. The use of ASD technology is expected to become widespread during the coming fourth industrial revolution, in applications such as factory automation utilizing artificial intelligence [1].

When training ASD models, it would be difficult to collect data representing every possible anomalous sound that could occur, because these sounds rarely occur during the normal operation of factory equipment, and the types of anomalous sounds which are possible are very diverse. Therefore, it is desirable to train ASD models without using anomalous data. In addition, real-world environments are often complicated and different conditions than those foreseen during the training of the ASD models may be encountered. Therefore, it is desirable to develop models that can detect anomalous sounds even when the normal state distribution is changed (i.e., after domain shift).

The two main approaches that have been proposed for performing ASD are generative methods and classification methods. Generative methods use only the normal data of a target sound to model its probability distribution and detect data that does not correspond to the model, categorized as anomalous. As a result of advances in deep learning technology, typical generative methods now involve the training of autoencoders (AE) to reconstruct normal data and calculate reconstruction error, or the use of autoregressive models with recursive neural networks to calculate the model likelihood, which is then used as an anomaly score [2, 3, 4, 5]. On the other hand, more recently developed classification methods distinguish between normal and outlier data by calculating anomaly scores based on distance from a decision boundary [6, 7, 8, 9], which have attracted much attention. Normal data from the operation of different machines is often used as outlier data during training. This method assumes that anomalous data is distributed outside the normal data, and that the outlier data is distributed even further outside the normal data. Based on this assumption, a binary classifier is trained using the normal data as positive examples, and the outlier data as pseudo-negative examples. Although generative and classification methods are both able to achieve good performance, they are unable to resolve the domain shift problem because the training and test data are recorded in the same environment.

Therefore, in this paper we propose an ASD method which is an ensemble of an autoencoder and a binary classification model, allowing it to function well even under domain shift conditions. The first component is a conformer-based, sequence-level autoencoder with section ID regression and a self-attention architecture [3]. We then utilize data augmentation techniques such as SpecAugment [10] to boost performance, and add a simple scorer module to each section and each domain to address the domain shift problem. The second component of our ASD method is a binary classification model employing metric learning, that uses task-irrelevant outliers as pseudo-anomalous data while controlling the centroids of normal and outlier data in a feature space [9]. As a countermeasure against the domain shift problem, we also perform data augmentation based on Mixup [11] using data from the target domain, resulting in stable performance for each data section. An ensemble approach is

applied to each of the two components of our method, and the resulting ensemble methods are then ensembled to maximize ASD performance. We conduct our experimental evaluation using the DCASE 2021 Challenge Task 2 dataset.

2. BASELINE METHODS

This section provides an overview of methods that achieved high ASD performance when using the DCASE 2020 Task 2 data [12]. The Task 2 datasets contain recordings of six types of machines, and each set of audio data for each type of machine consists of seven or eight different machines of that type. ID information is provided to indicate which machine the audio data belonged to.

2.1. Conformer-based autoencoder [3]

The autoencoder method [3] assumes that an autoencoder would not be able to accurately reconstruct anomalous data, i.e., data other than normal data used to train the autoencoder. We assume that the input of an autoencoder at frame t is $\mathbf{x}_t$, and that the corresponding output is $\hat{\mathbf{x}}_t$. Reconstruction error e_t can be computed as follows:

$$e_t = \mathrm{abs}(\hat{\mathbf{x}}_t - \mathbf{x}_t), \tag{1}$$

where $\mathrm{abs}(\cdot)$ denotes an element-wise absolute operator. If $\mathbf{x}_t$ contains anomalous data, the norm of e_t should be large. Thus, anomaly detection can be performed by simple thresholding.

In this paper, we use a conformer [13] as an autoencoder. ID regression is used to accurately detect anomalous sounds combined with the sound of the target machine. We concatenate the integer machine ID to the input features, and the autoencoder then reconstructs the input acoustic features and the machine ID. The autoencoder tends to misidentify the machine ID when the audio clip includes anomalous sound, even if we provide the correct machine ID as an input. Therefore, we can detect whether the audio clip includes anomalous sound from the estimated machine ID. In addition, we modify $e_{1:T}$ based on the distribution of the reconstruction error to improve detection accuracy. Specifically, frame-level anomaly score, a_t, represent the negative likelihood of a Gaussian mixture model (GMM) consisting of K-mixture Gaussians for e_t:

$$a_t = -\sum_{k=1}^{K} w_k \mathcal{N}(e_t \mid \mu_k, \mathbf{\Sigma}_k), \tag{2}$$

where w_k, μ_k, and $\mathbf{\Sigma}_k$ are the weight, mean vector, and covariance matrix of the k th mixture component, respectively. For training the parameters, we use the reconstruction error calculated from the validation set, which is not used for training the autoencoder, but is randomly selected $10\,\%$ from the development data set. The frame-by-frame anomaly scores obtained by the GMM are aggregated into the final anomaly scores by removing some outlier data and using the softmax weighted average, since some of the lowest or highest negative likelihood may have adversely affected the anomaly scores. The aggregated anomaly score $\hat{a}$ is given by:

$$\hat{a} = \frac{1}{T'} \sum_{i=1}^{T'} a_i^{(\mathcal{M})} \frac{\exp(\alpha a_i^{(\mathcal{M})})}{\sum_{i=1}^{T'} \exp(\alpha a_i^{(\mathcal{M})})}, \tag{3}$$

where $a_i^{(\mathcal{M})}$ represents the frame-by-frame anomaly scores selected from a_i, T' is the number of selected frames, and α is a scalar hyperparameter.

2.2. Binary classifier with metric learning [9]

A binary classifier is trained using normal data as positive examples and with outlier data as pseudo-negative examples. We perform learning for each particular machine ID. The normal data for a particular target machine ID is used as the normal data, and the normal data of other machine IDs of the same machine type, as well as the normal data of all of the other machines in the same dataset, are used as outlier data.

Consider a set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ that has N samples of normal and outlier data. Normal and outlier data sets are assigned labels $y_i \in \{+1, -1\}$ $(i = 1, 2, ..., N)$ for each data sample. When performing ASD using a method based on binary classification, the network is trained to minimize the following binary cross-entropy (BCE) loss function:

$$\mathcal{L}_{\mathrm{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} \Big\{ u(y_i) \log(p_i) + (1 - u(y_i)) \log(1 - p_i) \Big\}, \tag{4}$$

where p represents the posterior probabilities output of network ϕ_p (e.g., $p = \phi_p(\mathbf{x})$), which minimizes (4) when $\mathbf{x}$ is used as input, and where $u(y)$ is a binary function that takes 1 for $y > 0$ and 0 for $y \leq 0$. To further improve binary classification, we use the Deep Double-Centroids Semi-supervised Anomaly Detection (DDCSAD) loss function proposed in [9], which considers the centroids of both the normal and outlier data. The objective of the DDCSAD loss function is to minimize intra-class variance and maximize inter-class variance. The DDCSAD loss function is calculated as follows:

$$\mathcal{L}_{\mathrm{DDCSAD}} = \frac{1}{N} \sum_{i=1}^{N} \Big\{ \parallel \mathbf{z}_i - \mathbf{c}_p \parallel^{2y_i} + \parallel \mathbf{z}_i - \mathbf{c}_n \parallel^{-2y_i} \Big\}, \tag{5}$$

where $\mathbf{z}$ is the embedding vector output by encoder network ϕ_z (e.g., $\mathbf{z} = \phi_z(\mathbf{x})$), which minimizes (5) when $\mathbf{x}$ is used as the input, and where $\mathbf{c}_p \in \mathbb{R}^D$ and $\mathbf{c}_n \in \mathbb{R}^D$ represent the centroid of the normal and outlier data, respectively. Note that the initial values of centroids $\mathbf{c}_p$ and $\mathbf{c}_n$ are calculated using randomly initialized parameters, and are then recalculated at each epoch using the entire training data set. They are updated at each epoch by recalculating the centroids using the entire training data set. The following equation expresses the final loss function:

$$\mathcal{L} = \mathcal{L}_{\mathrm{BCE}} + \lambda \mathcal{L}_{\mathrm{DDCSAD}}, \tag{6}$$

where $\lambda > 0$ is a hyperparameter that controls the balance between the loss functions. Multi-task learning using both the cross-entropy of the posterior probability and the DDCSAD loss function increases accuracy when learning the decision boundaries, resulting in more accurate ASD.

During inference, posterior probability p, and distance $d = \parallel \mathbf{z} - \mathbf{c}_p \parallel^2$ between embedding vector $\mathbf{z}$ and centroid $\mathbf{c}_p$ of the normal class, are used to obtain the anomaly score. First, we compute distance d across the entire set of evaluation data, and then calculate standardized distance d' across the entire dataset. Finally, anomaly score s is calculated using the following equation:

$$s = \gamma \times (1 - p) + (1 - \gamma) \times d', \tag{7}$$

where, γ is a hyperparameter that determines the proportion of anomaly scores using posterior probability p.

3. PROPOSED METHODS

This section describes our proposed ASD methods working under domain shift conditions. We use section ID instead of machine ID due to the dataset change, but they have almost the same meaning.

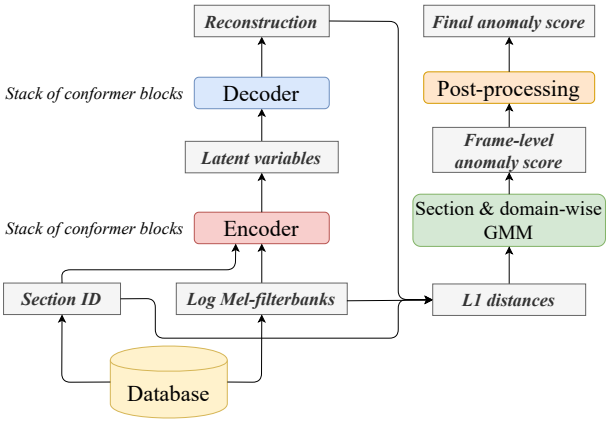


Figure 1: *Overview of proposed autoencoder method.*

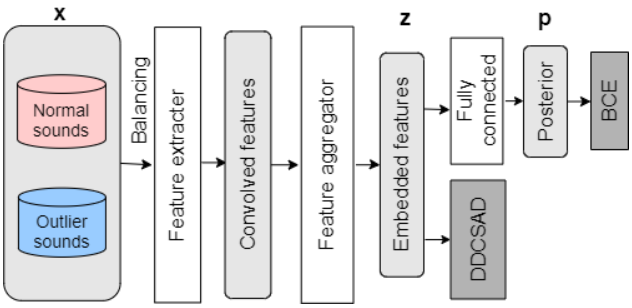


Figure 2: *Overview of proposed binary classification method.*

3.1. Ensembled, comformer-based autoencoder

The first component of our method is a sequence-level autoencoder with ID regression, which is based on the method described in Section 2.1. An overview is shown in Fig. 1. To boost the autoencoder's performance, we utilize SpecAugment [10] and dropout for the input feature sequence. Inspired by the interpolation deep neural network approach proposed in [14], we apply SpecAugment and dropout for the input feature sequence not only during training but also during inference, where we replicate the input sequence and apply different masking for each sequence. We then calculate the reconstruction error for each sequence and integrate the results using a pooling operation (e.g., average or max). This allows us to obtain the gain by using an ensemble model, even when using a single model. To address the domain shift problem, we build separate reconstruction error scoring modules for each section and each domain. This method is enabling us to capture differences in the anomaly score range between the sections and domains.

To further improve performance, we ensemble the model by selecting the N-best models when using the development data for each machine and each domain, and then integrate the outputs of these models to obtain the final score. We normalize the outputs and combine the normalized scores using the following four methods: *average*, *median*, *maximum*, and *ranking* which converts the scores into a rank and then calculates the average of the rankings.

3.2. Ensembled, binary classifier with metric learning

The second component of our proposed method is a binary classification model using metric learning, based on the method described in Section 2.2. An overview is shown in Fig. 2. We build the model for the source domain, and then perform fine-tuning for the target domain. When training for the source domain, only data from the source domain is used. On the other hand, when fine-tuning the model for the target domain, we create pseudo-target domain data using Mixup [11], using source and target domain data. It is expected that the use of Mixup will increase variation within each class and create data with an intermediate representation between the positive and negative examples.

The performance of the binary classification method is less stable than that of the autoencoder method, since the binary classification method builds a different model for each section of data. Therefore, we create many models in order to improve the performance of the model ensemble; for example, we change the method of pseudo-anomalous example selection, introduce additional data augmentation using methods such as Gaussian Noise and Volume control, change the architecture of the feature extraction module, and use an additional loss function (ArcFace [15]). ArcFace is a loss function used to achieve a clear geometric interpretation within a feature space, and the combination of ArcFace with DDCSAD results in further improvement in ASD performance. During pseudo-anomalous example selection, we also select samples from within the same dataset, resulting in more stable performance across the data sections. Finally, we use various different models as our feature extraction module, including ResNet34 [16], ResNeXt50 [17] and EfficientNet b3 [18] in PyTorch Image Models [19]. We average $N \times S$ models for each machine and each domain, where S represents the number of sections in the validation set.

3.3. Ensemble of the autoencoder and binary classifier models

We normalize the anomaly score of each of the two ensembled models to mean = 0 and variance = 1, and average the normalized scores.

4. EXPERIMENTAL EVALUATION

4.1. Experimental conditions

We conducted our experimental evaluation using the DCASE 2021 Challenge Task 2 dataset [20, 21]. The dataset consists of the normal and anomalous operating sounds of seven types of real machines: *ToyCar*, *ToyTrain*, *fan*, *gearbox*, *pump*, *slider*, and *valve*. Data for each type of machine includes six sections. Each section is further divided into two domains, containing source and target data, respectively. Each recording is a single-channel, 10 second segment of audio sampled at 16 kHz. The training data includes 1,000 samples in the source domain and only 3 samples in the target domain. The development and evaluation data consists of around 200 samples in each domain. The training data includes only normal sounds, but the development data includes both normal and anomalous sounds to allow the evaluation of anomaly detection performance. To evaluate the performance of our proposed method, we included the following models in our experiment for comparison:

Baseline (AE): The official autoencoder-based baseline method [22], trained with normal training data, which minimizes reconstruction error.

Baseline (MNV2): The official classification-based baseline method, which uses MobileNetV2 [22] trained using section ID classification.

Our baseline (AE): Our baseline, sequence-level autoencoder

Table 1: *Evaluation results. Values represent the harmonic mean of AUC* [%] *and pAUC (*$p = 0.1$*)* [%] *for each section of each domain. "All/har-mean" column values represent the harmonic mean of AUC and pAUC over all machines, sections and domains.*

		ToyCar		ToyTrain		fan		gearbox		pump		silder		valve		All
	Method	source	target	source	target	source	target	source	target	source	target	source	target	source	target	har-mean
dev	Baseline (AE)	59.44	54.74	64.31	51.99	59.14	56.72	56.42	61.04	63.85	53.01	67.09	55.71	52.43	51.45	57.28
	Baseline (MNV2)	57.19	55.89	58.81	50.77	63.31	61.58	65.54	60.72	62.20	57.36	65.43	52.17	53.99	55.17	58.22
	Our baseline (AE)	80.41	63.05	80.50	61.46	71.82	66.35	62.69	70.01	72.61	62.41	86.04	62.01	80.60	64.30	69.05
	Our baseline (BC)	57.91	58.68	76.23	49.04	67.36	59.36	**74.85**	74.59	72.12	59.86	80.64	57.24	86.18	70.10	69.08
	AE ens	**83.29**	68.70	**81.06**	**62.83**	74.37	69.75	64.14	72.32	74.60	**65.68**	86.12	65.41	82.94	67.81	71.67
	BC ens	60.93	64.55	76.16	55.40	**82.29**	66.62	74.49	70.58	75.20	60.70	**89.28**	57.69	92.70	79.35	73.29
	AE+BC ens (mix)	79.90	**70.08**	80.23	59.85	82.25	**71.58**	72.95	**76.25**	**77.29**	64.03	89.06	**68.49**	**93.03**	**80.15**	**76.59**
	AE+BC ens (max)	**83.29**	68.70	**81.16**	**62.83**	**82.29**	69.75	74.49	72.32	75.20	**65.68**	**89.28**	65.41	92.70	79.35	75.68
eval	Baseline (AE)	61.33	55.63	61.86	**63.26**	57.99	52.54	61.17	**60.58**	56.38	52.55	56.76	51.78	51.22	50.69	56.38
	Baseline (MNV2)	41.81	**57.59**	49.76	43.50	63.65	59.24	53.31	49.55	63.79	64.00	66.17	66.34	53.86	50.86	54.77
	AE ens	54.94	54.95	**65.84**	54.82	62.84	59.00	**66.18**	60.31	58.13	62.14	71.45	62.49	63.30	49.52	59.92
	BC ens	**64.39**	55.07	54.86	52.90	65.97	**63.70**	55.91	50.44	**81.40**	**79.86**	**84.22**	75.69	66.23	**58.49**	63.21
	AE+BC ens (mix)	60.83	56.30	64.64	54.84	**70.01**	63.12	60.93	56.21	70.46	64.84	82.26	**77.36**	**68.78**	55.10	**63.75**
	AE+BC ens (max)	54.94	54.95	**65.84**	54.82	65.97	59.00	55.91	60.31	**81.40**	62.14	**84.22**	62.49	66.23	**58.49**	62.26

model, trained for 50,000 steps using the Adam optimizer [23] with Warmup scheduler [24]. The batch size was set to 64 and the number of warmup steps was 8,000. The hyperparameters were optimized for each machine and each domain, including Mel-spectrogram extraction condition (e.g., shift size and Mel basis), model architecture (e.g., the number of blocks, units and kernel size) and post-processing. In SpecAugment, the number of time masks was set to 50, with a width range from one to five, while the number of frequency masks was set to five, with a width range from zero to ten. The dropout rate for the input sequence was set to 0.2.

Our baseline (BC): Our baseline, binary classification-based model was ResNet34 [16]. The size of the spectrogram was 256×256. The model was trained for 8,000 steps using the Adam optimizer, with a learning rate for the fully-connected layer of 1.0e-3, and a learning rate for the convolution layer of 5.0e-4. The OneCycleLR scheduler [25] was used, and the batch size was 64. The ratio of normal to outlier data was set to 1:1 in the mini-batch. When fine-tuning for the target domain, we trained the pre-trained model created using source domain data for 800 steps. Sampling was performed during fine-tuning so that the mini-batch always contained 16 samples of target domain or pseudo-target domain data, which was obtained by mixing up data from the target and source domains.

AE ens: An ensemble of the proposed autoencoder models. The value of N was selected from among 3, 5, 10 and 20, and the ensemble methods were optimized for each machine and each domain.

BC ens: An ensemble (average) of the proposed binary-classification models. The value of N was set to two.

AE+BC ens (mix): The average of AE ens and BC ens.

AE+BC ens (max): The ensemble of AE ens and BC ens. We took the maximum output value between AE ens and BC ens for each machine and each domain.

The hyperparameters and post-processing parameters of each model were optimized for each section and each domain.

4.2. Experimental results

Our experimental results are shown in Table 1. First, we focus on the results when using the development data. When comparing the performance of our baseline (AE) and AE ens, and our baseline (BC) and BC ens, we can see that performance of harmonic mean generally improved. When comparing the results when using our two baseline methods, we can see that AE outperformed BC for machine types *ToyCar* and *ToyTrain*, while BC outperformed AE for machine types *gearbox* and *valve*, and further improvements are yielded by ensembling these two methods, suggesting that each of them focuses on different features of each machine type.

Next, we focus on performance when using the evaluation data. BC ens outperformed AE ens for machine types *pump*, *slider* and *valve*, regardless of the domain. In these types of machines, where the sound generated is non-stationary (i.e., it includes a variety of intermittent sounds, such as clicks), the BC-based method was found to be superior. Unlike our results when using the development data, ASD performance for *ToyCar* when using AE ens decreased. These results suggest that model performance tends to be influenced not only by machine type, but also by the type of anomalous sound that is present. Finally, we found that all of the proposed methods outperformed all of the baseline methods with both datasets, and that the AE+BC ens (mix) model achieved the best ASD performance.

These results demonstrated that our proposed method performed well under domain shift conditions. Furthermore, we found that using an ensemble of the results from different ASD models focusing on different features contributes to score improvement, since the outputs of the models complement each other.

5. CONCLUSION

In this paper, we presented an ensemble ASD approach using both a conformer-based autoencoder and a binary classification model with metric learning. Our experimental evaluation showed that the proposed methods significantly outperformed the baseline methods by achieving higher ASD scores. We also demonstrated that by using an ensemble of completely different ASD methods, we were able to obtain better performance. These results suggest that different ASD methods focus on different audio data features to detect anomalous sounds, so it is important to ensemble models that can pick out different features. In future work, we will develop a method that can obtain the better ASD performance using fewer models.

6. ACKNOWLEDGMENT

This paper was partly supported by a project, JPNP20006, commissioned by NEDO.

7. REFERENCES

[1] B. Bayram, T. B. Duman, and G. Ince, "Real time detection of acoustic anomalies in industrial processes using sequential autoencoders," *Expert Systems*, vol. 38, no. 1, p. e12564, 2021.

[2] K. Suefusa, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, "Anomalous Sound Detection Based on Interpolation Deep Neural Network," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 271–275.

[3] T. Hayashi, T. Yoshimura, and Y. Adachi, "Conformer-based id-aware autoencoder for unsupervised anomalous sound detection," DCASE2020 Challenge, Tech. Rep., July 2020.

[4] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series," in *Proceedings*, vol. 89. Presses universitaires de Louvain, 2015, pp. 89–94.

[5] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara, "Latent Space Autoregression for Novelty Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] P. Primus, V. Haunschmid, P. Praher, and G. Widmer, "Anomalous sound detection as a simple binary classification problem with careful selection of proxy outlier examples," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 170–174.

[7] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep Semi-Supervised Anomaly Detection," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HkgH0TEYwH

[8] L. Ruff, R. A. Vandermeulen, B. J. Franks, K.-R. Müller, and M. Kloft, "Rethinking Assumptions in Deep Anomaly Detection," in *ICML 2021 Workshop on Uncertainty & Robustness in Deep Learning*, 2021.

[9] I. Kuroyanagi, T. Hayashi, K. Takeda, and T. Toda, "Anomalous sound detection using a binary classification model and class centroids," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1995–1999.

[10] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proceedings of Interspeech 2019*, 2019, pp. 2613–2617.

[11] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[12] Y. Koizumi *et al.*, "Description and discussion on dcase2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 81–85.

[13] A. Gulati *et al.*, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech 2020*, 2020, pp. 5036–5040. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-3015

[14] K. Suefusa, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, "Anomalous sound detection based on interpolation deep neural network," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 271–275.

[15] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[18] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.

[19] R. Wightman, "PyTorch Image Models," https://github.com/rwightman/pytorch-image-models, 2019.

[20] R. Tanabe, H. Purohit, K. Dohi, T. Endo, Y. Nikaido, T. Nakamura, and Y. Kawaguchi, "MIMII DUE: Sound dataset for malfunctioning industrial machine investigation and inspection with domain shifts due to changes in operational and environmental conditions," *arXiv preprint arXiv:2006.05822*, 2021.

[21] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, "ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions," *arXiv preprint arXiv:2106.02369*, 2021.

[22] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and discussion on DCASE2021 Challenge Task 2: Unsupervised anomalous sound detection for machine condition monitoring under domain shifted conditions," *arXiv preprint arXiv:2106.04492*, 2021.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of 3rd International Conference on Learning Representations, ICLR 2015*, 2015, pp. 1–15.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of 2017 Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[25] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, T. Pham, Ed., vol. 11006, International Society for Optics and Photonics. SPIE, 2019, pp. 369 – 386. [Online]. Available: https://doi.org/10.1117/12.2520589

THE IMPACT OF NON-TARGET EVENTS IN SYNTHETIC SOUNDSCAPES FOR SOUND EVENT DETECTION

Francesca Ronchini[1], *Romain Serizel*[1], *Nicolas Turpault*[1], *Samuele Cornell*[2]

[1]Université de Lorraine, CNRS, Inria, Loria, Nancy, France
[2] Department of Information Engineering, Universita Politecnica delle Marche, Italy

ABSTRACT

Detection and Classification Acoustic Scene and Events Challenge 2021 Task 4 uses a heterogeneous dataset that includes both recorded and synthetic soundscapes. Until recently only target sound events were considered when synthesizing the soundscapes. However, recorded soundscapes often contain a substantial amount of non-target events that may affect the performance. In this paper, we focus on the impact of these non-target events in the synthetic soundscapes. Firstly, we investigate to what extent using non-target events alternatively during the training or validation phase (or none of them) helps the system to correctly detect target events. Secondly, we analyze to what extend adjusting the signal-to-noise ratio between target and non-target events at training improves the sound event detection performance. The results show that using both target and non-target events for only one of the phases (validation or training) helps the system to properly detect sound events, outperforming the baseline (which uses non-target events in both phases). The paper also reports the results of a preliminary study on evaluating the system on clips that contain only non-target events. This opens questions for future work on non-target subset and acoustic similarity between target and non-target events which might confuse the system.

Index Terms— Sound event detection, synthetic soundscapes, open-source datasets, deep learning

1. INTRODUCTION

The main goal of ambient sound and scene analysis is to automatically extract information from sounds that surround us and analyze them for different purposes and applications. Between the different area of interest, ambient sound analysis have a considerable impact on applications such as noise monitoring in smart cities [1, 2], domestic applications such as smart homes and home security solutions [3, 4], health monitoring systems [5], multimedia information retrieval [6] and bioacoustics domain [7]. Sound Event Detection (SED) aims to identify the onset and offset of the sound events present in a soundscape and to correctly classify them, labeling the events according to the target sound classes that they belong to. Nowadays, deep learning is the main method used to approach the problem. However, one of the main limitations of deep learning models is the requirement of large amounts of labeled training data to reach good performance. The process of labeling data is time-consuming and bias-prone mainly due to human errors and disagreement given the subjectivity in the perception of some sound event onsets and offsets [8]. To overcome these limitations, recent works are investigating alternatives to train deep neural networks with a small amount of labeled data together with a bigger set of unlabeled data [3, 9, 10, 8, 11]. Among them, Detection and Classification Acoustic Scenes and Events Challenge (DCASE) 2021 Task 4 uses an heterogeneous dataset that includes both recorded and synthetic soundscapes [8]. This latter soundscapes provide a cheap way to obtain strongly labeled data. Until recently, synthesized soundscapes were generated considering only target sound events. However, recorded soundscapes also contain a considerable amount of non-target events that might influence the performance of the system.

The purpose of this paper is to focus on the impact on the system's performance when non-target events are included in the synthetic soundscapes of the training dataset. The study has been mainly divided into three stages. Firstly, we investigate to what extent using non-target events alternatively during training or validation helps the system to correctly detect the target sound events. Mainly motivated from the results of the first experiment, in the second part of the study, we focus on understanding to what extend adjusting the target to non-target signal-to-noise ratio (TNTSNR) at training improves the sound event detection performance. Results regarding a preliminary study on the evaluation of the system using clips containing only non-target events are also reported, opening questions for future studies on possible acoustic similarity between target and non-target sound events which might confuse the SED system. [1]

2. PROBLEM DEFINITION AND DATASET GENERATION

2.1. Problem definition

The primary goal of the DCASE 2021 Challenge Task 4 is the development of a semi-supervised system for SED, exploiting an heterogeneous and unbalanced training dataset. The goal of the system is to correctly classify the sound event classes and to localize the different target sound events present in an audio clip in terms of timing. Each audio recording can contain more than one event. Some of those could also be overlapped. The use of a larger

This work was made with the support of the French National Research Agency, in the framework of the project LEAUDS Learning to understand audio scenes (ANR-18-CE23-0020), the project CPS4EU Cyber Physical Systems for Europe (Grant Agreement number: 826276) and the French region Grand-Est. Experiments presented in this paper were carried out using the Grid5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000).

[1]To promote reproducibility, the code, https://github.com/DCASE-REPO/DESED_task, and pre-trained models https://zenodo.org/record/5529692, are made available under an open-source license.

amount of unlabeled recorded clips is motivated by the limitations related to annotating a SED dataset (human-error-prone and time-consuming). Alternatively, synthesized soundscapes are an easy way to have strongly annotated data. In fact, the user can easily generate the soundscapes starting from isolated sound events. On the other hand, in most of the recorded soundscapes the target sound classes are almost never present alone. For this reason, one of the main novelties of the DCASE 2021 Challenge Task 4 is the introduction of non-target isolated events in the synthetic soundscapes[2]. This paper explores the impact of the non-target sound events on the baseline system performance, with the final goal of understanding and highlighting how to correctly exploit them to generate realistic soundscapes.

2.2. Dataset generation

The dataset used in this paper is the DESED dataset[3] [12, 13], which is the same provided for the DCASE 2021 Challenge Task 4. It is composed of 10 seconds length audio clips either recorded in a domestic environment or synthesized to reproduce such an environment[4]. The synthetic part of the dataset is generated with Scaper [14], a Python library for soundscape synthesis and augmentation, which allows to control audio parameters. The recorded soundscapes are taken from AudioSet [15]. The foreground events (both target and non-target) are obtained from the Freesound Dataset (FSD50k) [16], while the background sounds are obtained from the SINS dataset (activity class "other") [17] and TUT scenes 2016 development dataset [18]. In particular, non-target events are the intersection of FUSS dataset [19] and FSD50k dataset in order to have compatibilty with the source separation baseline system.

In this article, we modify only the synthetic subset of the dataset. Starting from the synthetic part of the DESED dataset, we generated different versions of it in order to investigate how non-target events impact the system performance and to what extent their relationship with the target events affects the training phase of the system. The following subsections describe the different subsets used for the experiments, which have been generated using Scaper.

2.2.1. *Synthetic training set*

The synthetic training set is the same set of data released for the DCASE 2021 Challenge Task 4. It includes 10000 audio clips where both target and non-target sound events could be present in each clip. The distribution of the sound events among the files have been determined considering the co-occurrences between the different sound events. The co-occurrences have been calculate considering the strong annotations released for the AudioSet dataset [20][5]. A second version of this dataset has been generated where only target events are present. The datasets will be hereafter referred as **synth_tg_ntg** (used by the official baseline system) and **synth_tg** for the synthetic subset including target and non-target events and the synthetic subset including only target events, respectively.

2.2.2. *Synthetic validation set*

The synthetic validation set is the same as the synthetic validation dataset supplied for the DCASE 2021 Challenge Task 4. It includes 3000 audio clips including target and non-target events, which distribution has been defined calculating the co-occurrences between sound events. We generated a second version of the dataset containing only target events. The datasets will be referred to as **synth_tg_ntg_val** (used by the baseline system) and **synth_tg_val** (only target sound events).

2.2.3. *Synthetic evaluation set*

The synthetic 2021 evaluation set is composed by 1000 audio clips. In the context of the challenge, this subset is used for analysis purposes. We will refer to it as **synth_tg_ntg_eval**. It contains target and non-target events distributed between the different audio clips according to the pre-calculated co-occurrences. Two different versions of the **synth_tg_ntg_eval** set have been generated, **synth_tg_eval** (only target sound events) and **synth_ntg_eval** (only non-target sound events).

2.2.4. *Varying TNTSNR training and validation set*

With the aim of studying what would be the impact of varying the TNTSNR on the system performance, different versions of **synth_tg_ntg** and **synth_tg_ntg_val** have been generated. In particular, for each of them, three versions have been created. The SNR of the non-target events have been decreased by 5 dB, 10 dB and 15 dB compared to their original value. The original SNR of the sound events is randomly selected between 6 dB and 30 dB, so the more we decrease the SNR, the less the sound will be audible, with some of the events that will not be audible at all. These subsets will be subsequently referred to as **synth_5dB**, **synth_10dB**, **synth_15dB** for the training subsets and **synth_5dB_val**, **synth_10dB_val**, **synth_15dB_val** for the validation subsets.

2.2.5. *Public evaluation set*

The public evaluation set is composed of recorded audio clips extracted from Youtube videos that are under creative common licenses. This is part of the evaluation dataset released for the evaluation phase of the DCASE 2021 Challenge Task 4 and considered for ranking. The set will be referred to as **public**.

3. EXPERIMENTS TASK SETUP

In order to compare the results with the official baseline, we used the same SED mean-teacher system released for this year challenge. More information regarding the system can be found at Turpault et al. [8] and on the official webpage of the DCASE Challenge Task 4. All the different models have been trained 5 times. This paper reports the average of the scores and the confidence intervals related to those. Only for the baseline model we do no report the confidence intervals because we have considered the results using the checkpoint made available for it[6]. The metrics considered for the study are the two polyphonic sound detection score (PSDS) [21] scenarios defined for the DCASE 2021 Challenge Task 4, since these are the official metrics used in the challenge.

[2] http://dcase.community/challenge2021

[3] https://project.inria.fr/desed/

[4] For a detailed description of the DESED dataset and how it is generated the reader is referred to the original DESED article [13] and DCASE 2021 task 4 webpage: http://dcase.community/challenge2021

[5] The co-occurrences distribution and the code used to compute them will be distributed.

[6] https://zenodo.org/record/4639817

Non-target		PSDS1	PSDS2
Train	Val		
✓		33.81 (0.36)	52.62 (0.19)
	✓	35.92 (0.49)	54.85 (0.29)
		34.90 (0.82)	53.07 (1.22)
✓	✓	**36.40**	**58.00**

Table 1: Evaluation results for the **public** set, considering the different combinations of using target and non-target sound events at training and validation.

The scope of these experiments is twofold: understand the impact of non-target events on the system performance and investigate to what extend the TNTSNR helps the network to correctly predict the sound events in both matched and mismatched conditions. In order to do so, we divided the experiment into three stages. The first part of the study is focused on understanding the influence of training the system with non-target events. This experiment is described and discussed in Section 4. Section 5 reports the results and the relative discussion of the second part of the experiment where we investigate if a mismatch in terms of TNTSNR between datasets could have an impact on the output of the system. Section 6 reports preliminary results of the last stage of the experiment, regarding the evaluation of the system on the **synth_ntg_eval** dataset, formed by only non-target sound events, in order to investigate if some classes could get acoustically confused at training, having a negative impact on the performance. The last stage has been motivated by the results of the second part of the experiment.

4. USING TARGET/NON-TARGET AT TRAINING

In the first experiment we concentrate on training the system with different combinations of the training dataset. Table 1 reports the results of the experiment evaluating the system on the **public** set. We check-marked the columns NT Train or/and NT Val according to if the non-target sound events are present or not in the synthetic sounscapes. From the results it is possible to observe that using non-target sound events during training and validation improves the performance by a large margin with relaxed segmentation constraints (PSDS2) but only marginally with strict segmentation constraints (PSDS1). In this latter case what matters the most is the use of non-target sound events during the validation. A possible explanation is that synthetic soundscapes with non-target sound events are actually too difficult and confuse the systems when used during the training but they still help reducing the mismatch with recorded soundscapes during model selection (validation).

Table 2 reports the results considering the **synth_tg_ntg_eval** and **synth_tg_eval** evaluation sets. In all cases the best performance is obtained in matched training/evaluation conditions. The performance obtained on **synth_tg_ntg_eval** are lower than the performance obtained on **synth_tg_eval** even in matched conditions. Not surprisingly, this confirm that including non-target sound events makes the SED task more difficult. Interestingly, as opposed to the previous experiment, the most important here is to have matched conditions during training and to a lesser extent during validation. In order to verify the low impact of non-target sound events at training when evaluating on recorded soundscapes, in the next experiment we investigate a possible mismatch in terms in TNTSNR.

Non-target		Eval set	PSDS1	PSDS2
Train	Val			
✓		synth_tg_ntg_eval	23.22 (1.33)	36.44 (2.62)
	✓	synth_tg_ntg_eval	20.08 (0.39)	31.33 (1.29)
		synth_tg_ntg_eval	20.13 (0.35)	30.99 (1.07)
✓	✓	synth_tg_ntg_eval	**25.14**	**40.12**
✓		synth_tg_eval	42.82 (2.42)	58.26 (2.08)
	✓	synth_tg_eval	46.92 (1.02)	**62.79 (0.55)**
		synth_tg_eval	**47.73 (0.33)**	62.54 (1.00)
✓	✓	synth_tg_eval	43.22	61.09

Table 2: Evaluation results for the **synth_tg_ntg_eval** set and **synth_tg_eval** set, considering the different combination of using target and non-target sound events at training and validation.

Non-target		PSDS1	PSDS2
Train	Val		
Original	5 dB	35.57 (0.28)	56.68 (1.77)
5 dB	Original	36.25 (1.26)	57.53 (1.06)
5 dB	5 dB	35.46 (0.46)	**58.09** (0.74)
Original	Original	**36.40**	58.00

Table 3: Evaluation results for the second part of the experiment, varying TNTSNR by 5 dB (**synth_5dB** and **synth_5dB_val**). Evaluating with **public** set.

Non-target		PSDS1	PSDS2
Train	Val		
Original	10 db	36.23 (1.11)	57.82 (1.37)
10 db	Original	**36.42 (0.77)**	**58.94 (0.89)**
10 db	10 db	36.20 (1.14)	57.92 (1.04)
Original	Original	36.40	58.00

Table 4: Evaluation results for the second part of the experiment, varying TNTSNR by 10 dB (**synth_10dB** and **synth_10dB_val**). Evaluating with **public** set.

Non-target		PSDS1	PSDS2
Train	Val		
Original	15 dB	36.08 (1.13)	57.78 (1.33)
15 dB	Original	**37.37 (0.70)**	**58.64 (1.34)**
15 dB	15 dB	36.10 (0.50)	57.36 (0.89)
Original	Original	36.40	58.00

Table 5: Evaluation results for the second part of the experiment, varying TNTSNR by 15 dB (**synth_15dB** and **synth_15dB_val**). Evaluating with **public** set.

5. VARYING TNTSNR AT TRAINING

The second part of the study focuses on understanding the impact of varying the TNTSNR at training and validation aiming at finding a TNTSNR condition that could match better the recorded soundscapes. For each TNTSNR, we use similar combinations as the ones used in Section 4, replacing the set without non-target sound events by a set with adjusted TNTSNR. For example, considering the 5 dB

Validation set	PSDS1	PSDS2
synth_5dB_val	38.68 (1.07)	60.57 (0.78)
synth_10dB_val	**39.07 (0.75)**	**60.75 (0.80)**
synth_15dB_val	37.95 (0.53)	59.99 (1.14)

Table 6: Evaluation results of the SED system, training with **synth_tg**, validating with varying TNTNSNR set and evaluating with **public** set.

case, the combinations considered would be:

- training using the **synth_tg_ntg** set and validating with **synth_5dB_val**;
- training with **synth_5dB** and validating with **synth_tg_ntg_val**;
- training and validating with **synth_5dB** and **synth_5dB_val**.

The fourth combination is the official DCASE Task 4 baseline. Repeating the experiment with all the varying TNTSNR, allow us to analyse to what extend the loudness of the non-target events helps matching the evaluation conditions on recorded clips. Table 3, 4 and 5 report the performance on the **public** set when using a TNTSNR of 5 dB, 10 dB and 15 dB, respectively. When the TNTSNR is 5 dB or 10 dB, the performance changes only marginally between configurations. Increasing the TNTSNR to 15 dB leads to a behaviour more similar to the one obtained in Table 1. The best performance is obtained when training with TNTSNR is 15 dB and validating on **synth_tg_ntg_val**. This could be explained by the fact TNTSNR 15 dB is a condition closer to that of the recorded soundscapes and the fact that it allows for selecting models that will be more robust towards non-target events at test time.

In the last experiment, we investigate the impact of varying the TNTSNR during validation phase, while using the **synt_tg** for training. Results are reported on Table 6, where it is possible to observe that all of them overcome the baseline or are comparable with it, with the best performance obtained for 10 dB TNTSNR. These experiments could indicate that recorded soundscapes in **public** in general have a TNTSNR of about 10 – 15 dB which should be confirmed by complementary experiments.

6. EVALUATING ON NON-TARGET EVENTS ONLY

Based on the previous experiments, TNTSNR could be one reason of mismatch between the synthetic soundscapes and the recorded soundscapes. But this could not explain all the performance differences observed here. In particular why in general having lower TNTSNR during training is decreasing the performance regardless of the validation. One possibility is that the system gets acoustically confused by a possible similarity in sound between events when soundscapes tend to be less dominated by target events. So we evaluated the system using the **synth_ntg_eval**, where only non-target events are considered, to see for which classes the system would output false positives. We evaluated the system on the **public** set; considering the systems trained for the first experiment (see Table 1). Results show that some sound events are detected more than others. For some classes as Speech, this could be explained by the original event distribution (indicated in the first column) but for some other classes as Dishes there is a discrepancy between the original distribution and the amount of false alarms. Interestingly the amount of false alarms is decreased sensibly for most of the classes when including non-target sound events during training.

Classes	Nref	Nsys A	Nsys B	Nsys C	Nsys Base
Dog	197	135	126	146	79
Vacuum_cleaner	127	31	42	44	47
Alarm_bell	191	47	50	52	59
Running_water	116	34	41	61	30
Dishes	405	1478	395	1270	305
Blender	100	63	32	55	19
Frying	156	70	41	60	33
Speech	1686	206	181	180	201
Cat	141	99	103	98	73
Electric_shaver	103	21	18	18	7

Table 7: Preliminary evaluation results by classes, evaluating the system with **synth_ntg_eval**. Nsys (A): training with **synth_tg**, validating with **synth_tg_val**; Nsys (B): training with **synth_tg_ntg**, validating with **synth_tg_val**; Nsys (C): training with **synth_tg**, validating with **synth_tg_ntg_val**; Base: baseline using target and non-target events for training and validation.

7. CONCLUSIONS AND FUTURE WORK

This paper analyzes the impact of including non-target sound events in the synthetic soundscapes of the training dataset for SED systems trained on heterogeneous dataset. In particular, the experiments are divided into three stages: in the first part, we explore to what extend using non-target sound events at training has an impact on the system's performance, secondly we investigate the impact of varying TNTSNR and we conclude the study by analyzing a possible confusion of the SED model in case of false alarms triggered by non-target sound events.

From the results reported on this paper, we can conclude that using non-target sound events can help the SED system to better detect the target sound events, but it is not clear to what extend and what would be the best way to generate the soundscapes. Results show that the final SED performance could depend on mismatches between synthetic and recorded soundscapes, part of which could be due to the TNTSNR but not only. Results on the last experiment show that using non-target events at training decreases the amount of false alarms at test but from this experiment it is not possible to conclude on the impact of non-target sound events on the confusion between the target sound events. This is a first track for future investigation on the topic. Additionally, the impact of the non-target sound events at training on the ability of the system to better segment the target sound events in noisy soundscapes would have to be investigated. A final open question is the impact of the per class distribution of the sound events (both target and non-target) and their co-occurrence distribution on the SED performance.

8. ACKNOWLEDGEMENTS

We would like to thank all the other organizers of DCASE 2021 Challenge Task 4. In particular, we thank Eduardo Fonseca and Daniel P. W. Ellis for their help with the strong labels of the AudioSet dataset used to compute the events co-occurrences, and Justin Salamon and Prem Seetharaman for their help with Scaper.

9. REFERENCES

[1] J. P. Bello, C. Silva, O. Nov, R. L. DuBois, A. Arora, J. Salamon, C. Mydlarz, and H. Doraiswamy, "Sonyc: A system for the monitoring, analysis and mitigation of urban noise pollution," *arXiv preprint arXiv:1805.00889*, 2018.

[2] J. P. Bello, C. Mydlarz, and J. Salamon, "Sound analysis in smart cities," in *Computational Analysis of Sound Scenes and Events*. Springer, 2018, pp. 373–397.

[3] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. P. Shah, "Large-scale weakly labeled semi-supervised sound event detection in domestic environments," *arXiv preprint arXiv:1807.10501*, 2018.

[4] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring activities of daily living in smart homes: Understanding human behavior," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 81–94, 2016.

[5] Y. Zigel, D. Litvak, and I. Gannot, "A method for automatic fall detection of elderly people using floor vibrations and sound—proof of concept on human mimicking doll falls," *IEEE transactions on biomedical engineering*, vol. 56, no. 12, pp. 2858–2867, 2009.

[6] Q. Jin, P. Schulam, S. Rawat, S. Burger, D. Ding, and F. Metze, "Event-based video retrieval using audio," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[7] V. Morfi, R. F. Lachlan, and D. Stowell, "Deep perceptual embeddings for unlabelled animal sound events," *The Journal of the Acoustical Society of America*, vol. 150, no. 1, pp. 2–11, 2021.

[8] N. Turpault, R. Serizel, A. Parag Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, New York City, United States, October 2019. [Online]. Available: https://hal.inria.fr/hal-02160855

[9] R. Serizel and N. Turpault, "Sound event detection from partially annotated data: Trends and challenges," in *IcETRAN conference*, 2019.

[10] A. Shah, A. Kumar, A. G. Hauptmann, and B. Raj, "A closer look at weak label learning for audio events," *arXiv preprint arXiv:1804.09288*, 2018.

[11] B. McFee, J. Salamon, and J. P. Bello, "Adaptive pooling operators for weakly labeled sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2180–2193, 2018.

[12] R. Serizel, N. Turpault, A. Shah, and J. Salamon, "Sound event detection in synthetic domestic environments," in *ICASSP 2020 - 45th International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain, May 2020. [Online]. Available: https://hal.inria.fr/hal-02355573

[13] N. Turpault, R. Serizel, A. Parag Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, New York City, United States, Oct. 2019. [Online]. Available: https://hal.inria.fr/hal-02160855

[14] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello, "Scaper: A library for soundscape synthesis and augmentation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 344–348.

[15] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[16] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: an open dataset of human-labeled sound events," *arXiv preprint arXiv:2010.00475*, 2020.

[17] G. Dekkers, S. Lauwereins, B. Thoen, M. W. Adhana, H. Brouckxon, B. Van den Bergh, T. Van Waterschoot, B. Vanrumste, M. Verhelst, and P. Karsmakers, "The sins database for detection of daily activities in a home environment using an acoustic sensor network," *Detection and Classification of Acoustic Scenes and Events 2017*, pp. 1–5, 2017.

[18] A. Mesaros, T. Heittola, and T. Virtanen, "Tut database for acoustic scene classification and sound event detection," in *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1128–1132.

[19] S. Wisdom, H. Erdogan, D. P. Ellis, R. Serizel, N. Turpault, E. Fonseca, J. Salamon, P. Seetharaman, and J. R. Hershey, "What's all the fuss about free universal sound separation data?" in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 186–190.

[20] S. Hershey, D. P. Ellis, E. Fonseca, A. Jansen, C. Liu, R. C. Moore, and M. Plakal, "The benefit of temporally-strong labels in audio event classification," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 366–370.

[21] Ç. Bilen, G. Ferroni, F. Tuveri, J. Azcarreta, and S. Krstulović, "A framework for the robust evaluation of sound event detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 61–65.

WHAT MAKES SOUND EVENT LOCALIZATION AND DETECTION DIFFICULT? INSIGHTS FROM ERROR ANALYSIS

Thi Ngoc Tho Nguyen[1], *Karn N. Watcharasupat*[1],
Zhen Jian Lee, Ngoc Khanh Nguyen, Douglas L. Jones[2], *Woon Seng Gan*[1]

[1] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.
[2] Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, IL, USA.
{nguyenth003, karn001}@e.ntu.edu.sg, zhenjianlee@gmail.com,
ngockhanh5794@gmail.com, dl-jones@illinois.edu, ewsgan@ntu.edu.sg

ABSTRACT

Sound event localization and detection (SELD) is an emerging research topic that aims to unify the tasks of sound event detection and direction-of-arrival estimation. As a result, SELD inherits the challenges of both tasks, such as noise, reverberation, interference, polyphony, and non-stationarity of sound sources. Furthermore, SELD often faces an additional challenge of assigning correct correspondences between the detected sound classes and directions of arrival to multiple overlapping sound events. Previous studies have shown that unknown interferences in reverberant environments often cause major degradation in the performance of SELD systems. To further understand the challenges of the SELD task, we performed a detailed error analysis on two of our SELD systems, which both ranked second in the team category of DCASE SELD Challenge, one in 2020 and one in 2021. Experimental results indicate polyphony as the main challenge in SELD, due to the difficulty in detecting all sound events of interest. In addition, the SELD systems tend to make fewer errors for the polyphonic scenario that is dominant in the training set.

Index Terms— DCASE, error analysis, polyphony, sound event localization and detection

1. INTRODUCTION

Sound event localization and detection (SELD) has many applications in urban sound sensing [1], wildlife monitoring [2], surveillance [3], autonomous driving [4], and robotics [5]. SELD is an emerging research field that aims to combine the tasks of sound event detection (SED) and direction-of-arrival estimation (DOAE) by jointly recognizing the sound classes, and estimating the directions of arrival (DOA), the onsets, and offsets of detected sound events [6].

The introduction of the SELD task in the 2019 Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) has significantly accelerated SELD research. Many significant contributions have been made over the last three years in terms of datasets, evaluation metrics, and algorithms [7]. The TAU Spatial Sound Events dataset [8] used in DCASE 2019 included only stationary sound sources, with 72 room impulse responses (RIRs) from 5 different locations, and only 20 distinct samples for each of the 11 sound classes. The TAU-NIGENS Spatial Sound Events dataset [9] used in DCASE 2020 saw an introduction of moving sound sources, more RIRs from 15 different locations, and 14 sound classes extracted from the NIGENS General Sound Events Database [10], with around 30 to 50 distinct samples per class. The 2021 edition [11] introduced unknown directional interferences, making the sound scenes more realistic, in addition to the increase in the maximum polyphony of target events to three, from two in the 2019 and 2020 runs. The number of sound classes was reduced to 12, as some classes were used as interferences. All three SELD datasets provide both first-order ambisonic (FOA) and microphone array (MIC) formats.

The SELD evaluation metrics have evolved over the past three years. In DCASE 2019, SED and DOAE performances were evaluated independently. Segment-wise error rate (ER) and F1 score evaluation were used for SED [12], while frame-wise DOA error and frame recall were used for DOAE [13]. Since 2020, SED and DOAE were evaluated jointly with location-dependent ER and F1 score for SED, and class-dependent localization error (LE) and localization recall (LR) for DOAE [14]. The 2021 metrics further take into account overlapping same-class events [11].

On the algorithm aspect, there have been many developments for SELD, inside and outside the DCASE Challenges, in the areas of data augmentation, feature engineering, model architectures, and output formats. In 2015, an early monophonic SELD work by Hirvonen [15] formulated SELD as a classification task, where each output class represents a sound class-location pair. In 2018, Adavanne et al. pioneered a seminal polyphonic SELD work that used a single-input multiple-output convolutional recurrent neural network (CRNN) model, SELDnet, to jointly detect sound events and estimate the corresponding DOAs [6]. In 2019, Cao et al. proposed a two-stage strategy by training separate SED and DOA models [16], then using the SED outputs as masks to select the DOA outputs, significantly outperforming the jointly-trained SELDnet. Cao et al. later proposed an end-to-end SELD network [17] that used soft parameter sharing between the SED and DOAE encoder branches and output trackwise predictions. An improved version of this network [18] replaced the bidirectional gated recurrent units (GRU) with multi-head self-attention (MHSA) to decode the SELD outputs [18]. In 2020, Shimada et al. proposed a new output format for SELD which unified SED and DOAE into one loss function [19]. This was amongst the few works which successfully used the linear-frequency for spectrograms and interchannel phase differences as input features, instead of the mel spectrograms. A new

This research was supported by the Singapore Ministry of Education Academic Research Fund Tier-2, under research grant MOE2017-T2-2-060.

K. N. Watcharasupat acknowledges the support from the CN Yang Scholars Programme, Nanyang Technological University, Singapore.

CNN architecture, D3Net [20], was adapted into a CRNN for this work and showed promising results. In another research direction, Nguyen et al. proposed to solve SED and DOAE separately, use a bidirectional GRU to match the SED and DOAE output sequences, then produce event-wise SELD outputs [21, 22]. This was based on the observation that different sound events often have different onsets and offsets, resulting in temporal matching in the SED and DOAE output sequences. In 2021, Nguyen et al. proposed a new input feature, SALSA, which spectrotemporally aligns the spatial cues with the signal power in the linear-frequency scale to improve SELD performance [23].

The top SELD system for DCASE 2019 trained four separate models for sound activity detection, SED, single-source DOAE, and two-source DOAE [24]. The top systems for both DCASE 2020 and 2021 synthesized a larger dataset from the original data, employed many data augmentation techniques, and combined different SELD models into ensembles [25, 26]. Other highly ranked solutions also intensively used data augmentation and ensemble methods.

Since SELD consists of both SED and DOAE tasks, it inherits many challenges from both SED and DOAE, such as noise, reverberation, interference, polyphony, and non-stationarity of sound sources. Furthermore, SELD often faces an additional challenge in correctly associating SED and DOAE outputs of multiple overlapping sound events. In an attempt to dissect the difficulties of the SELD task, Politis et al. compared the performances of the same SELD system in different acoustic environments [11] with different combinations of noise, reverberation, and unknown interferences. The authors founded that, in absence of unknown interferences, ambiance noise has little negative effects on SELD performance, while reverberation significantly reduces the SELD performance in all noise combinations. Unknown interferences degrade SELD performances by the largest margin compared to noise and reverberation. In addition, using the FOA format generally produces better performance than the MIC format.

To further understand the challenges facing SELD, we performed detailed error analysis on the SELD outputs, with the focus on polyphony, moving source, class-location interdependence, class-wise performance, and DOA errors, using our two SELD systems which both ranked second in the team category for the 2020 and 2021 DCASE Challenges [23, 27]. Experimental results showed that polyphony is the main factor that decreases the SELD performance across all the evaluation metrics, explaining why unknown interferences reduced the SELD performance by the largest extent. Interestingly, we also found that SELD systems do not necessarily favor single-source scenarios, which is easier than polyphonic cases. Instead, SELD systems achieved lower error rates in polyphonic cases which dominate the training dataset. The rest of the paper is organized as follows. Section 2 describes our analysis method. Section 3 presents the experimental results and discussions. Finally, we conclude the paper in Section 4.

2. ANALYSIS METHOD

In this section, brief descriptions of the SELD datasets and systems are provided. Error analyses were performed on the SELD outputs of the two SELD systems which both ranked second in the team category for the 2020 and 2021 DCASE Challenges [23, 27]. The 2021 version of the evaluation metrics was used in all analyses. For convenience, the TAU-NIGENS Spatial Sound Events 2020 and 2021 datasets used in the DCASE Challenges [9, 11] are referred to here as the SELD 2020 and 2021 datasets, respectively.

Characteristics	2020	2021
Channel format	FOA	FOA
Moving sources	✓	✓
Ambiance noise	✓	✓
Reverberation	✓	✓
Unknown interferences	×	✓
Maximum degree of polyphony	2	3
Number of target sound classes	14	12
Evaluation split	eval	test

Table 1: Comparison between 2020 and 2021 SELD datasets

2.1. Dataset

Table 1 summarizes some differences between the two SELD datasets. Since both of the SELD systems require the FOA format, only the FOA subset of the datasets were used in our experiments. Each of the dataset consists of 400, 100, 100, and 200 one-minute audio recordings for the train, validation, test, and evaluation splits respectively. The azimuth and elevation ranges are $[-180°, 180°)$ and $[-45°, 45°]$, respectively. During the developmental stage, the validation set was used for model selection while the test set was used for evaluation. During the evaluation stage, the train, validation, and test data (collectively known as the development split) were used for training evaluation models. For the 2020 SELD dataset, the results on the evaluation split were used for the error analyses. Since the ground truth for the evaluation split of the 2021 SELD dataset has not been publicly released at the time of writing, the results on the test split of the 2021 SELD dataset were used for error analysis instead.

2.2. Evaluation metrics

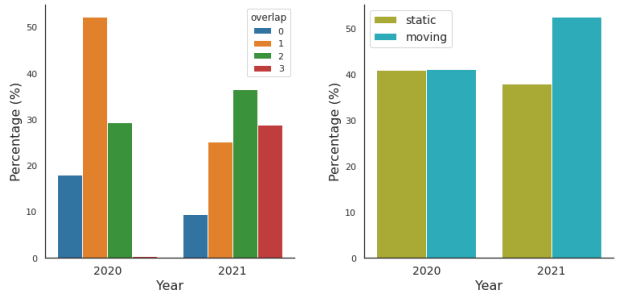
To evaluate the SELD performance, we used the official SELD evaluation metrics [7] from the DCASE 2021 Challenge. The metrics not only jointly evaluate SED and DOAE, but also take into account the cases where multiple instances of the same class overlap. The SELD evaluation metrics consist of location-dependent error rate ($\text{ER}_{\leq T}$) and F1 score ($\text{F}_{\leq T}$) for SED; and class-dependent localization error (LE_{CD}), and localization recall (LR_{CD}) for DOAE. A sound event is considered a correct detection only if it has a correct class prediction and its estimated DOA is also less than T away from the DOA ground truth, where $T = 20°$ for the official challenge. The DOAE metrics are also class-dependent, that is, the detected DOA is only counted if its corresponding detected sound class is correct. A good SELD system should have low $\text{ER}_{\leq T}$, high $\text{F}_{\leq T}$, low LE_{CD}, and high LR_{CD}.

2.3. SELD systems

We denote two of our SELD systems that ranked second in the team categories of the 2020 and 2021 DCASE challenges as NTU'20 and NTU'21, respectively. Table 2 shows the performances of the baselines, the top-ranked solutions, and our second-ranked systems in 2020 and 2021. NTU'20 is an ensemble of sequence matching networks [21, 27] while NTU'21 is an ensemble of different models trained on our new proposed SALSA features for SELD [23]. Both systems use the class-wise output format, which can only detect a maximum of one event of a particular class at a time. Both systems outperformed the respective baselines by a large margin, and only

Year	System	$ER_{\leq 20^\circ}$	$F_{\leq 20^\circ}$	LE_{CD}	LR_{CD}
2020 (eval)	Baseline [9]	0.69	0.413	23.1°	0.624
	#1: USTC'20 [25]	**0.20**	**0.849**	**6.0°**	0.885
	#2: NTU'20 [27]	0.23	0.820	9.3°	**0.900**
2021 (test)	Baseline [11]	0.73	0.307	24.5°	0.448
	#1: Sony'21 [26]	0.43	0.699	**11.1°**	0.732
	#2: NTU'21 [23]	**0.37**	**0.737**	11.2°	**0.741**

Table 2: Performance of selected SELD systems.



(a) Polyphonic distribution (b) Static vs moving

Figure 1: Segment-wise polyphonic and static distribution per year.

perform slightly worse than the respective top-ranked system. The 2020 results in Table 2 were computed using the 2020 SELD evaluation metrics. For subsequent sections, the results of the NTU'20 system were recomputed using the 2021 metrics.

3. EXPERIMENTAL RESULTS AND DISCUSSION

In each subsection concerning a factor of variation, we performed an analysis on the data distribution of 2020 and 2021 SELD datasets, followed by an analysis of the SELD results. Overall, the 2021 dataset is much more challenging than the 2020 dataset. For detailed analyses, $ER_{\leq T}$ is further broken down into substitution, deletion, and insertion errors, while $F_{\leq T}$ is further broken down into precision and recall. Since the SELD metrics are segment-based, i.e., outputs are divided into segments of 1 s before being evaluated, we used the provided ground truth to group the segments based on polyphony (0, 1, 2, and 3 sources), static and moving sources to compute the metrics for each case.

3.1. Effect of polyphony

Figure 1(a) shows the segment-wise polyphonic distribution of 2020 and 2021 datasets, which are dominated by single-source and two-source segments, respectively. On average, there are 1.11 and 1.85 events per segment in the 2020 and 2021 datasets, respectively. Table 3 shows the breakdown of the SELD performance for each polyphonic case. The DOAE metrics clearly show that polyphony is a major cause of performance degradation. For both NTU'20 and NTU'21 systems, as the number of overlapping sources increases, LE_{CD}increases and LR_{CD} decreases. Interestingly, polyphony does not always degrade SED performance. The peak performances of $ER_{\leq 20^\circ}$ and precision were achieved in the degree of polyphony that dominates the respective dataset, which is single-source for the 2020 dataset and two-source for the 2021 dataset. This result suggests that one possible solution to tackle polyphony is to introduce more data samples for difficult cases.

	2020			2021			
Metrics	1	2	All	1	2	3	All
↓ $ER_{\leq 20^\circ}$	**0.108**	0.331	0.232	0.349	**0.338**	0.394	0.372
↓ Substitution	**0.029**	0.072	0.052	**0.093**	0.104	0.129	0.114
↓ Deletion	**0.042**	0.155	0.103	**0.091**	0.137	0.182	0.152
↓ Insertion	**0.038**	0.104	0.078	0.164	0.096	**0.083**	0.105
↑ $F_{\leq 20^\circ}$	**0.930**	0.765	0.845	**0.784**	0.763	0.704	0.737
↑ Precision	**0.932**	0.788	0.875	0.757	**0.780**	0.746	0.756
↑ Recall	**0.928**	0.743	0.833	**0.813**	0.747	0.666	0.719
↓ LE_{CD}	**5.6**	13.4	9.4	**6.8**	10.3	13.5	11.2
↑ LR_{CD}	**0.930**	0.775	0.846	**0.816**	0.764	0.701	0.741

Table 3: SELD performance w.r.t. degree of polyphony

	2020			2021		
Metrics	Static	Moving	All	Static	Moving	All
↓ $ER_{\leq 20^\circ}$	**0.214**	0.239	0.232	0.379	**0.357**	0.372
↑ $F_{\leq 20^\circ}$	**0.854**	0.841	0.845	0.731	**0.745**	0.737
↓ LE_{CD}	**8.7**	10.0	9.4	**10.5**	11.7	11.2
↑ LR_{CD}	**0.847**	0.846	0.846	0.725	**0.751**	0.741
↓ $ER_{\leq 180^\circ}$	**0.166**	0.168	0.171	0.334	**0.298**	0.318
↑ $F_{\leq 180^\circ}$	**0.898**	0.891	0.892	0.778	**0.800**	0.789

Table 4: SELD performance of static and moving sources.

When the number of overlapping sources increases, the SED error compositions also change. The deletion error rate rapidly increases, the insertion error rate sharply decreases, and the substitution error rate increases. In addition, the recall rate decreases significantly. It is clear that the SELD systems struggle to detect all the present events in polyphonic cases.

In the absence of any event of interest, the insertion error rates are 0.030 and 0.122 for NTU'20 and NTU'21 systems, respectively. When comparing the SELD performances between the 2020 and 2021 setups, the single-source results in 2021 are significantly worse than those in 2020 across all metrics. In addition, the substitution errors across all degrees of polyphony are much higher in the 2021 setup, than in 2020. These results show the detrimental effect of unknown interferences that were introduced in the 2021 dataset, consistent with the findings in [11].

3.2. Effect of moving sound sources

Figure 1(b) shows the segment-wise distribution of static and moving sound sources, not counting empty segments, based on the provided ground truth. A segment is considered a moving one if at least one sound source is moving. Since there are more overlapping sources in the 2021 dataset, the proportion of moving segments is significantly higher than the 2020 dataset. Table 4 presents the SELD performance for both cases. The LE_{CD} of moving-source cases is higher than those of static-source cases, as expected. For the 2020 dataset, the LR_{CD} are similar for both cases, and the performance gap for SED disappears when we compute location-independent SED metrics (by setting the DOA threshold to $T = 180^\circ$). These results suggest that moving sources have little effect on SED performance and mainly affect DOAE. For the

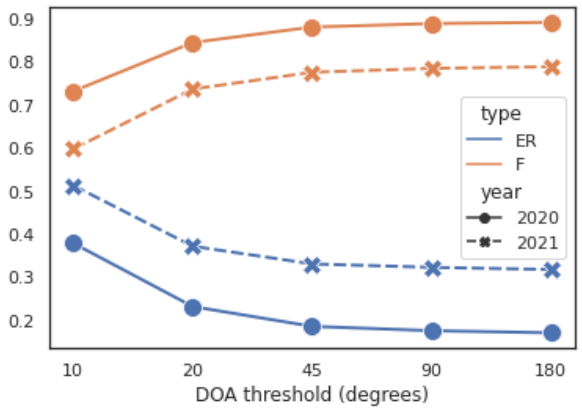
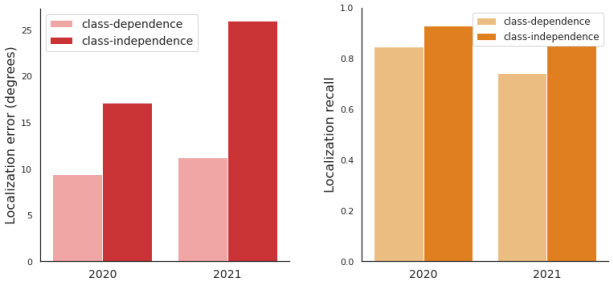


Figure 2: SED performance across different DOA thresholds.



(a) Localization error (b) Localization recall

Figure 3: Localization error and recall by class dependencies.

2021 dataset, all metrics are better for moving-source cases compared to single-source cases. This contradictory result may be due to the skewed distribution and requires further investigation once the evaluation ground truth is made available.

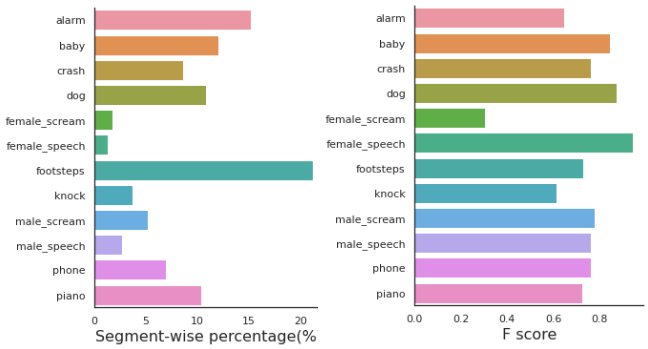
3.3. Class and location interdependency

To understand the dependency of location-dependent SED metrics on the correctness of the detected DOAs, we investigate the effect of the different DOA thresholds $T°$ on $ER_{\leq T°}$ and $F_{\leq T°}$, as shown in Figure 2. The gaps between the SED metrics for $T = 20°$ and the location-independent $T = 180°$ are not significantly large, suggesting that many estimated DOAs are within the 20° threshold. However, the location-dependent SED metrics deteriorate quickly as the DOA threshold reduces to 10°, suggesting a significant number of the estimated DOAs deviate by more than 10° from the ground truth.

To understand the dependency of classification-dependent DOA metrics on the correctness of the predicted classes, we show the classification-dependent and classification-independent LE and LD in Figure 3. When not accounting for the predicted class, the LR significantly increases, leading to some unwanted rise in LE.

3.4. Class-wise performance

Due to space constraints, we only included the segment-wise class distribution and the class-wise performance of 2021 setup in Figure 4. The segment-wise class distribution in Figure 4(a) is highly skewed, with the *footstep* class accounting for the highest proportion of 21.2 %, while the *female speech* accounting for the lowest at 1.3 %. However, the class-wise $F_{\leq 20°}$ scores are more even, and the class with the highest segment-wise proportion does not correspond to highest $F_{\leq 20°}$ score. One possible reason is that it is difficult to detect all *footstep* sound due to discontinuities, low bandwidth, and low energy. In addition, class-wise performance is highly dependent on the SELD model and the quality of training samples. Interestingly, the *female speech* class with the highest $F_{\leq 20°}$ score of 94.2 % has the lowest segment-wise proportion. Other classes such as *knock* and *male speech* also have high $F_{\leq 20°}$ scores despite the low segment-wise proportions.



(a) Class distribution (b) Class-wise $F_{\leq 20°}$

Figure 4: Segment-wise class distribution of 2021 SELD dataset (test split) and class-wise location-dependent F score of NTU'21 system.

3.5. Azimuth vs elevation error

For the NTU'20 system, the LE_{CD} contributed by azimuth and elevation are 6.3° and 5.3°, respectively. For the NTU'21 system, the LE_{CD} contributed by azimuth and elevation are 7.9° and 6.2°, respectively. The azimuth and elevation errors are similar although the azimuth range of $[-180°, 180°)$ is much larger than elevation range of $[-45°, 45°]$, suggesting that it is more difficult to estimate elevation angles than azimuth angles.

4. CONCLUSION

In realistic acoustic conditions with noise and reverberation, polyphony and unknown interferences appear to be the biggest challenges for SELD. In the presence of unknown interferences, SELD systems tend to make more substitution errors. When there are several sound events, either due to polyphony or unknown interferences, the SELD systems struggle to detect all events of interests, leading to low recall and high deletion error rate. Interestingly, the overall SED error rate is at the lowest for the polyphonic case that dominates the dataset. Moving sound sources mainly increase the localization errors, leading to small reduction in location-dependent SED metrics. High segment-wise representation of a class also does not necessary translate to high SED performances. Localization error reduction poses significant challenge beyond a threshold, especially as elevation errors are often as high as azimuth errors. The study of same-class polyphonic events is left for future works due to the limitations of the current systems studied.

5. REFERENCES

[1] J. Salamon and J. P. Bello, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 279–283, 2017.

[2] D. Stowell, M. Wood, Y. Stylianou, and H. Glotin, "Bird detection in audio: A survey and a challenge," in *IEEE Int. Workshop Mach. Learn. for Signal Process.*, 2016, pp. 1–6.

[3] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Audio Surveillance of Roads: A System for Detecting Anomalous Sounds," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 1, pp. 279–288, 2016.

[4] M. K. Nandwana and T. Hasan, "Towards smart-cars that can listen: Abnormal acoustic event detection on the road," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2016, pp. 2968–2971.

[5] J. M. Valin, F. Michaud, B. Hadjou, and J. Rouat, "Localization of simultaneous moving sound sources for mobile robot using a frequency-domain steered beamformer approach," in *Proc. IEEE Int. Conf. Robotics Autom.*, 2004, pp. 1033–1038.

[6] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound Event Localization and Detection of Overlapping Sources Using Convolutional Recurrent Neural Networks," *IEEE J. Sel. Top. Signal Process.*, vol. 13, no. 1, pp. 34–48, 2019.

[7] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen, "Overview and Evaluation of Sound Event Localization and Detection in {DCASE} 2019," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 684–698, 2020.

[8] S. Adavanne, A. Politis, and T. Virtanen, "A Multi-room Reverberant Dataset for Sound Event Localization and Detection," in *Proc. 4th Workshop Detect. Classif. Acoust. Scenes Events*, 2019, pp. 10–14.

[9] A. Politis, S. Adavanne, and T. Virtanen, "A Dataset of Reverberant Spatial Sound Scenes with Moving Sources for Sound Event Localization and Detection," *arXiv*, 2020.

[10] I. Trowitzsch, J. Taghia, Y. Kashef, and K. Obermayer, "The NIGENS General Sound Events Database," *arXiv*, 2019.

[11] A. Politis, S. Adavanne, D. Krause, A. Deleforge, P. Srivastava, and T. Virtanen, "A Dataset of Dynamic Reverberant Sound Scenes with Directional Interferers for Sound Event Localization and Detection," *arXiv*, 2021.

[12] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Appl. Sci. (Switzerland)*, vol. 6, no. 6, p. 162, 2016.

[13] S. Adavanne, A. Politis, and T. Virtanen, "Direction of arrival estimation for multiple sound sources using convolutional recurrent neural network," in *Proc. Eur. Signal Process. Conf.*, 2018, pp. 1462–1466.

[14] A. Mesaros, S. Adavanne, A. Politis, T. Heittola, and T. Virtanen, "Joint measurement of localization and detection of sound events," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2019, pp. 333–337.

[15] T. Hirvonen, "Classification of spatial audio location and content using Convolutional neural networks," in *Proc. 138th Audio Eng. Soc. Conv.* , 2015, pp. 622–631.

[16] Y. Cao, Q. Kong, T. Iqbal, F. An, W. Wang, and M. D. Plumbley, "Polyphonic Sound Event Detection and Localization using a Two-Stage Strategy," in *Proc. 4th Workshop Detect. Classif. Acoust. Scenes Events*, 2019.

[17] Y. Cao, T. Iqbal, Q. Kong, Y. Zhong, W. Wang, and M. D. Plumbley, "Event-independent Network for Polyphonic Sound Event Localization and Detection," in *Proc. 5th Workshop Detect. Classif. Acoust. Scenes Events*, 2020.

[18] Y. Cao, T. Iqbal, Q. Kong, F. An, W. Wang, and M. D. Plumbley, "An Improved Event-Independent Network for Polyphonic Sound Event Localization and Detection," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 885–889.

[19] K. Shimada, Y. Koyama, N. Takahashi, S. Takahashi, and Y. Mitsufuji, "ACCDOA: Activity-Coupled Cartesian Direction of Arrival Representation for Sound Event Localization And Detection," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 915–919.

[20] N. Takahashi and Y. Mitsufuji, "Densely connected multidilated convolutional networks for dense prediction tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021.

[21] T. N. T. Nguyen, D. L. Jones, and W. Gan, "A Sequence Matching Network for Polyphonic Sound Event Localization and Detection," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 71–75.

[22] T. N. T. Nguyen, N. K. Nguyen, H. Phan, L. Pham, K. Ooi, D. L. Jones, and W.-S. Gan, "A General Network Architecture for Sound Event Localization and Detection Using Transfer Learning and Recurrent Neural Network," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 935–939.

[23] T. N. T. Nguyen, K. Watcharasupat, N. K. Nguyen, D. L. Jones, and W. S. Gan, "DCASE 2021 Task 3: Spectrotemporally-aligned Features for Polyphonic Sound Event Localization and Detection," *IEEE AASP Chall. Detect. Classif. Acoust. Scenes Events* , 2021.

[24] S. Kapka and M. Lewandowski, "Sound Source Detection, Localization and Classification using Consecutive Ensemble of CRNN Models," *IEEE AASP Chall. Detect. Classif. Acoust. Scenes Events* , 2019.

[25] Q. Wang, H. Wu, Z. Jing, F. Ma, Y. Fang, Y. Wang, T. Chen, J. Pan, J. Du, and C.-H. Lee, "The USTC-iFlytek System for Sound Event Localization and Detection of DCASE2020 Challenge," *IEEE AASP Chall. Detect. Classif. Acoust. Scenes Events* , 2020.

[26] K. Shimada, N. Takahashi, Y. Koyama, S. Takahashi, E. Tsunoo, M. Takahashi, and Y. Mitsufuji, "Ensemble of ACCDOA- and EINV2-based Systems with D3Nets and Impulse Response Simulation for Sound Event Localization and Detection," *IEEE AASP Chall. Detect. Classif. Acoust. Scenes Events* , 2021.

[27] T. N. T. Nguyen, D. L. Jones, and W. S. Gan, "Ensemble of sequence matching networks for dynamic sound event localization, detection, and tracking," in *Proc. 5th Workshop Detect. Classif. Acoust. Scenes Events*, 2020, pp. 120–124.

A DATASET OF DYNAMIC REVERBERANT SOUND SCENES WITH DIRECTIONAL INTERFERERS FOR SOUND EVENT LOCALIZATION AND DETECTION

Archontis Politis[1], *Sharath Adavanne*[1], *Daniel Krause*[1], *Antoine Deleforge*[2]
Prerak Srivastava[2], *Tuomas Virtanen*[1],

[1] Audio Research Group, Tampere University, Tampere, Finland
[2] Universite de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

ABSTRACT

This report presents the dataset and baseline of Task 3 of the DCASE2021 Challenge on Sound Event Localization and Detection (SELD). The acoustical synthesis remains the same as in the previous iteration of the challenge, however the new dataset brings more challenging conditions of polyphony and overlapping instances of the same class. The most important difference is the introduction of directional interferers, meaning sound events that are localized in space but do not belong to the target classes to be detected and are not annotated. Since such interfering events are expected in every real-world scenario of SELD, the new dataset aims to promote systems that deal with this condition effectively. A modified SELDnet baseline employing the recent ACCDOA representation of SELD problems accompanies the dataset and it is shown to outperform the previous one. The new dataset is shown to be significantly more challenging for both baselines according to all considered metrics. To investigate the individual and combined effects of ambient noise, interferers, and reverberation, we study the performance of the baseline on different versions of the dataset excluding or including combinations of these factors. The results indicate that by far the most detrimental effects are caused by directional interferers.

Index Terms— Sound event localization and detection, sound source localization, acoustic scene analysis, microphone arrays

1. INTRODUCTION

Sound event localization and detection (SELD) is an audio processing task that aims to jointly detect temporally target classes of sound events and localize them in space when active. In that sense it differs from the classic sensor array task of sound source localization (SSL) which utilizes only spatial information to detect, localize, and track sources independently from their signal content [1]. It also differs from the popular sound event detection (SED) task which is focused on the temporal detection and classification part, omitting the spatial information of the scene. The spatiotemporal characterization of the scene produced by SELD makes it suitable for a range of applications such as robot audition and machine listening in general [2, 3], acoustic monitoring [4, 5], smart home environments [5, 6], improved human-machine interaction [7], speech recognition [8], and sonic information visualization [9], among others.

Research interest in SELD grew quickly during the last couple of years, with deep learning methods handling the task jointly [10], or fusing information from solving individual subtasks of SED and SSL [11, 12]. This interest culminated in the task becoming part of the DCASE Challenge in 2019, with participants bringing novel approaches to the problem, summarized in [13]. The dataset used in the challenge [14] included sound scenes from two different array formats with sound events spatialized in both azimuth and elevation using spatial room impulse responses (SRIRs) of real rooms. Additionally, spatial ambient noise captured in situ was added to the recordings. For the next iteration of the task in the DCASE Challenge 2020, a new dataset was generated based on SRIRs from additional rooms with more realistic and challenging conditions beyond the limitations of the first one [14]. More specifically, the discrete grid of potential directions-of-arrival (DOAs) of the older dataset was replaced with continuous DOA trajectories and, apart from static events, moving sources using interpolated SRIRs were emulated at different speeds. Furthermore, the newer SRIRs were captured in rooms of more diverse acoustical properties and from a wider range of distances, resulting in longer reverberation times and more challenging direct-to-reverberant ratios (DRRs).

The second iteration of the SELD task in DCASE2020 brought additional innovations, with participants experimenting with homogeneous joint loss functions [15, 16], self-attention layers [16, 17], advanced spatial augmentation strategies [15, 17], combinations of model-based localization with learning-based SED [18, 19], data-based fusion of individual SSL and SED systems [18, 20], and event- or track-based prediction modeling, instead of class-based prediction [21, 19]. The latter development specifically tried to address the case of same-class events occurring simultaneously [12, 21, 22], a case that distinguishes the SELD task from SED and becomes possible mainly due to spatial information. Research following the DCASE2020 challenge investigated fusion of pre-trained SED and SSL models [20], or parameter sharing between joint, semi-joint SELD models, and models fusing SSL and SED subsystems [22].

This report introduces the new **TAU-NIGENS Spatial Sound Events 2021**[1] dataset and the baseline[2] of the SELD challenge task in DCASE2021[3]. The major difference of this dataset with the previous one is the introduction of localized interfering events outside of the target classes. This condition, naturally encountered in a real environment, introduces new challenges to the task. Apart from the dataset and baseline description, we present an extensive evaluation of the baseline on different versions of the dataset with and without the presence of ambient noise, directional interferers, and reverberation.

[1] https://doi.org/10.5281/zenodo.4844825
[2] https://github.com/sharathadavanne/seld-dcase2021
[3] http://dcase.community/challenge2021/task-sound-event-localization-and-detection

2. DATASET

Similarly to the dataset of the previous iteration, the current one consists of 800 one-minute spatial recordings, of which 600 constitute the development set of the dataset, and the other 200 the evaluation set. The recordings are sampled at 24kHz, and they are offered in two 4-channel spatial audio formats, the raw signals of a tetrahedral microphone array and first-order Ambisonics, abbreviated as MIC and FOA for the rest of the paper. Detailed descriptions of the formats in terms of their directional encoding properties can be found in the previous challenge dataset report [14].

2.1. Sound events

The sound event samples are sourced from the *NIGENS general sound events database* [23], which consists of 14 classes of specific sound types, and an additional general one with disparate sounds not belonging to any of the other classes. We use the sounds in the 12 classes *alarm, crying baby, crash, barking dog, female scream, female speech, footsteps, knocking on door, male scream, male speech, ringing, phone, piano* as target events, and the sounds in the classes *running engine, burning fire* and the general class as directional interferers. This division results in about 500 distinct sound samples distributed across the target events of the dataset, and about 400 across the interfering events.

2.2. Dataset synthesis

The synthesis of the spatial sound reccordings are based on a collection of SRIRs acquired continuously along measurement trajectories inside 13 enclosures of Tampere University. The RIR collection and synthesis process is described in more detail in [14]. We summarize briefly the acoustical properties of the dataset. SRIRs are extracted along the measurement trajectories with an approximate resolution of 1 degree, resulting on about 1184 to 6480 possible RIRs/DOAs per room, depending on the type (circular/linear) and number of measurement trajectories. Events added in a single recording can be static or moving. The source position for a static event is drawn randomly from the pool of SRIRs of a single room used in that recording, while moving events are synthesized for one of the measured trajectories in the room. Moving events are synthesized to have an approximate speed of 10°/sec, 20°/sec, or 40°/sec, drawn randomly. The dataset is split into 8 folds with distinct rooms and samples in each of them. Distinct rooms result in different reverberation conditions, and even though similar ranges of DOAs may occur between rooms, the source distance, DRR, and reverberation conditions are distinct between folds for a certain DOA.

The events are laid out in layers in each recording, with the total number of layers determining the maximum polyphony possible. The parameter determining the density of events per layer and, hence, the average per-frame polyphony is the total gap time distributed between events in each layer. A larger gap time results in fewer events per layer and a lower average polyphony, while a smaller gap time results in higher event density and average polyphony. The last event per layer is truncated to fit the total 1 minute duration. For the present dataset there are three layers of target events and an additional layer of interfering events, resulting in a total maximum polyphony of 4. In addition to the spatialized reverberant events, multichannel ambient noise that was collected in each room with the same recording setup as the SRIRs is truncated to 1 minute segments and added to the event mixtures. The noise is scaled to result in signal-to-noise ratios (SNRs) drawn uniformly from noiseless (30dB) to noisy (6dB) conditions, with respect to the total energy of the target events excluding silences. An example of the layering of events in one recording is shown in Fig. 1.

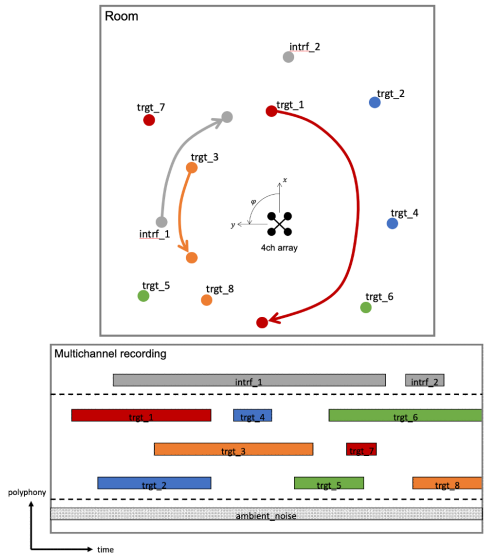


Figure 1: A graphic depiction of an emulated recording, with colored objects indicating target classes, gray objects indicating interferers and ambient noise, and arrows indicating moving events.

2.3. Differences with DCASE2020 task 3 dataset

Even though the acoustical and synthesis characteristics of the new dataset are similar to the dataset of the previous DCASE2020 Challenge, the following differences make it more challenging:

1. Directional interferers, out of the target classes of a detection system, are common in real conditions and they add to the challenge by forcing a strong joint modeling and training strategy that can learn to ignore them.
2. The overall maximum polyphony is increased from 2 to 3 target events.
3. The recordings are not anymore divided into recordings with no overlap (polyphony 1), and recordings with two simultaneous events (polyphony 2). Instead all recordings have the maximum level of polyphony, with all intermediate levels (from silence to 3 simultaneous target events + interference) varying during the duration of the recording. This choice reflects more natural recording conditions in a real dataset.
4. Even though the dataset of DCASE2020 had instances of the same class occurring at the same time, such occurrences were fairly rare. In the present dataset, these occurences have been increased in order to give a clear advantage to systems that can resolve this difficult but realistic case.

3. BASELINE

Similar to the previous iterations of the challenge, we adopt a modified version of SELDnet [10] as the baseline method, due to its conceptual simplicity. Its architecture remains a convolutional recur-

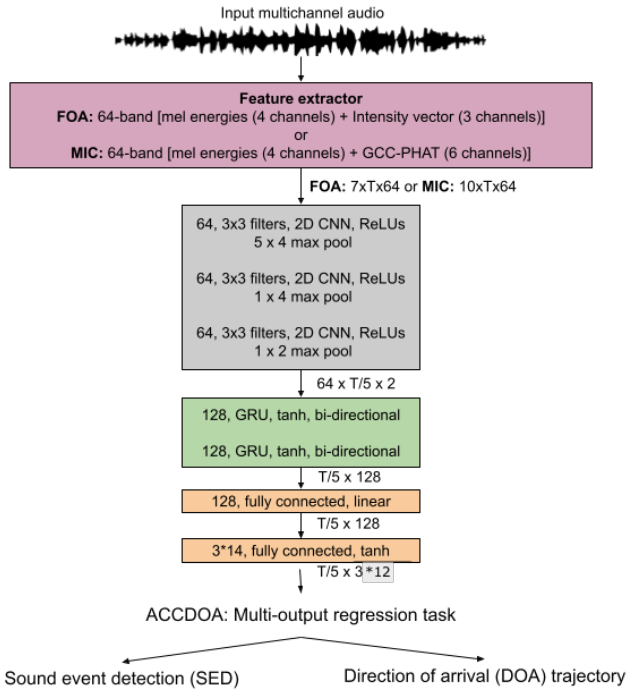


Figure 2: Convolutional recurrent neural network with ACCDOA loss for SELD.

rent neural network (CRNN) receiving multichannel log-mel spectrograms as inputs, together with acoustic intensity vectors [24] for the FOA dataset, and generalized cross-correlation (GCC-PHAT) sequences for the MIC dataset, added as extra channels. The baseline implementation extracts log-mel spectrograms in 64 mel-bands from 1024-point FFTs, using a 40 ms window and 20 ms hop length at 24kHz. The intensity vectors are similarly extracted for every FFT bin and aggregated in the same number of mel-bands as the spectrograms, while the GCC sequences are also truncated to the same number of lag values as the mel-bands, adopted from [11]. More details on the architecture and features can be found in [14].

The only difference of the current SELDnet baseline with respect to the previous DCASE challenge iteration is the output format and the respective loss function. The original SELDnet architecture employs separate output branches for detection and localization, with as many classification outputs and as many localization regressors as the number of classes. In the current baseline, we adopt the *activity-coupled cartesian direction of arrival* representation (ACCDOA) introduced in DCASE2020 Challenge by Shimada et al. [15], which unifies the SED and SSL losses into a single homogeneous regression loss, simplifying the overall architecture while simultaneously improving its performance. Using the ACCDOA representation, the network receives a sequence of T STFT frames of multichannel features and outputs $T/5 \times 3$ Cartesian vector coordinates for each of the target classes, with the direction of each vector indicating DOA and the vector length indicating class activity probability. A value of 0.5 on the length is used as the class activity threshold. The reduction in temporal resolution is intended to match the 100 ms resolution of annotations in the challenge. A block diagram of the current baseline is shown in Fig. 2.

4. EVALUATION

The dataset and baseline are delivered to the challenge participants at the commencement of the challenge, along with the development set of the dataset consisting of the 6 first folds, while the last two folds are made available during the evaluation phase of the challenge. Participants are required to report results on the test set of the development set using the predefined split of Table 1, so that conclusions can be drawn among the submissions on the same configuration. The evaluation split on Table 1, however, applies only to the evaluation results of the baseline presented herein, since during the evaluation phase participants have to report results on the testing folds (7–8) using the development dataset (folds 1–6) for training and validation in any way they see fit.

Table 1: Evaluation setup

	Splits		
Dataset	**Training**	**Validation**	**Testing**
Development	1,2,3,4	5	6
Evaluation	2, 3, 4, 5, 6	1	7, 8

The submissions are evaluated using the same combination of joint detection/localization metrics studied in [25, 13] and introduced in DCASE2020. Closer to SED evaluation, the localization-dependent error rate (ER_X) and F1-score (F_X) express detection performance but they penalize correct detections that occur further from the reference than some threshold distance X. On the other hand, the class-dependent localization error (LE_{CD}) and localization recall (LR_{CD}) are inspired by classical localization metrics, but are computed for each class individually before being averaged. The LE_{CD} is a mean angular localization error after pairing the predicted DOAs to their closest reference DOAs, while LR_{CD} is a simple recall metric on the detected localized events without any spatial threshold. Since in the SELD case there can be multiple simultaneous references of the same class, the detection metrics are modified to consider multiple instances of the same class and penalize cases where, e.g., only one of the predictions belong to that class. For the exact formulation of the metrics the reader is referred to [13]. The submissions are first ranked for each of the four metrics individually, and the final rank of each system is determined by the sum of the four individual ranks.

5. RESULTS

In order to evaluate the performance of the new baseline utilizing the ACCDOA loss, we compare it against the previous SELDnet baseline of DCASE2020, on the development set of DCASE2020 and the current one. Table 2 shows a clear improvement of the ACCDOA version in all metrics. Especially in the more challenging new dataset, the ACCDOA loss brings large gains in detection and improves localization accuracy by about 25%. A significant decrease of performance for both methods is also observed from the DCASE2020 dataset to the DCASE2021 dataset. This suggests that the new dataset is more challenging, as intended.

To get a more detailed picture on the effect of the various components in the scene, namely reverberation, ambient noise, and directional interferers, we generate various versions of the dataset including those components in various combinations. More specifically, the *targets*, *targets+ambience*, *targets+interferers*, *targets+ambience+interferers* develop from the presence of targets

Table 2: Comparison between the DCASE2020 baseline (*2020-multi*) and the current one (*2021-accdoa*) on the development set of DCASE2020 and the current development set.

	FOA				MIC			
	$ER_{20^\circ} \downarrow$	$F_{20^\circ} \uparrow$	$LE_{CD} \downarrow$	$LR_{CD} \uparrow$	$ER_{20^\circ} \downarrow$	$F_{20^\circ} \uparrow$	$LE_{CD} \downarrow$	$LR_{CD} \uparrow$
DCASE2020 development set								
2020-multi	0.70	44.4%	24.3°	61.9%	0.71	40.4%	25.4°	55.4%
2021-accdoa	0.60	51.9%	17.9°	59.8%	0.61	48.5%	19.3°	55.2%
DCASE2021 development set								
2020-multi	0.77	24.7%	32.1°	44.8%	0.81	19.1%	41.6°	47.4%
2021-accdoa	0.73	30.7%	24.5°	40.5%	0.75	23.4%	30.6°	37.8%

Table 3: Performance of the DCASE2021 baseline for different versions of the dataset with increasingly adverse conditions. The highlighted row corresponds to the version of the dataset used in the challenge.

	Development set								Evaluation set							
	FOA				MIC				FOA				MIC			
	$ER_{20^\circ} \downarrow$	$F_{20^\circ} \uparrow$	$LE_{CD} \downarrow$	$LR_{CD} \uparrow$	$ER_{20^\circ} \downarrow$	$F_{20^\circ} \uparrow$	$LE_{CD} \downarrow$	$LR_{CD} \uparrow$	$ER_{20^\circ} \downarrow$	$F_{20^\circ} \uparrow$	$LE_{CD} \downarrow$	$LR_{CD} \uparrow$	$ER_{20^\circ} \downarrow$	$F_{20^\circ} \uparrow$	$LE_{CD} \downarrow$	$LR_{CD} \uparrow$
Non-reverberant results																
targets	0.49	62.0%	16.3°	65.7%	0.54	55.4%	20.8°	63.7%	0.45	64.7%	15.8°	69.2%	0.50	58.7%	19.0°	67.0%
targets+ambience	0.49	61.2%	16.4°	65.6%	0.57	51.2%	20.8°	58.9%	0.47	62.7%	16.4°	67.6%	0.49	59.6%	18.6°	66.1%
targets+interferers	0.69	36.9%	24.1°	45.2%	0.72	27.7%	30.5°	42.2%	0.61	45.8%	21.2°	53.2%	0.69	31.5%	27.8°	44.9%
targets+ambience+ interferers	0.66	40.3%	22.7°	46.9%	0.73	26.7%	30.4°	42.5%	0.63	44.5%	22.0°	52.8%	0.70	32.1%	28.2°	46.6%
Reverberant results																
targets	0.55	53.7%	19.9°	61.3%	0.59	47.0%	22.0°	57.3%	0.52	56.4%	19.6°	64.7%	0.53	54.6%	20.8°	61.2%
targets+ambience	0.57	50.3%	20.2°	59.3%	0.62	44.2%	22.8°	53.6%	0.52	56.4%	19.0°	62.7%	0.56	51.6%	21.0°	58.5%
targets+interferers	0.71	32.7%	26.7°	44.2%	0.76	24.0%	32.6°	39.4%	0.69	34.8%	24.7°	43.7%	0.75	24.8%	32.7°	40.4%
targets+ambience+ interferers	0.73	30.7%	24.5°	40.5%	0.75	23.4%	30.6°	37.8%	0.67	37.2%	23.9°	45.8%	0.73	27.1%	30.8°	40.6%

only, to the inclusion of ambient noise or interferers separately, to the full dataset combining all components. Excluding the effect of reverberation is less straightforward due to the use of real SRIRs for the synthesis. In order to generate reverberation-free versions of the dataset, the sound events for each recording in the original dataset are spatialized with anechoic IRs of the same Eigenmike spherical microphone array used to capture the SRIRs. The anechoic array IRs are computed for the same measurement trajectories and DOAs as the measured SRIRs in each room, and stored in a similar data structure. Additionally, each IR is delayed and scaled according to the source distance of the respective measured SRIR, following an inverse distance law and a speed of sound of $c = 343m/sec$. Delaying and scaling ensures that the events between the reverberant and non-reverberant versions are approximately time-aligned and with comparable distance-dependent attenuation. The Eigenmike responses were measured in an equirectangular grid of 5° azimuth and 5° elevation in the large anechoic chamber of Aalto University, as described in [26]. Since the DOAs in the reverberant dataset do not necessarily coincide with the measurement grid of the array, array response interpolation is performed to recover anechoic IRs at the DOAs of the measured SRIRs, based on a spherical harmonic expansion of the array steering vectors, as in [26].

The results are presented in Table 3. As expected, reverberation affects negatively all combinations, increasing error rates and decreasing F-scores and localization recall in a consistent manner between the same scenarios. Additionally, it decreases localization accuracy by 2°–4°. Inclusion of the ambient noise has a small but noticeable effect when added to the targets, without interferers. The small effect may be due to the large range of possible positive SNRs (6–30dB) distributed uniformly across the recordings. Interestingly, together with directional interference, inclusion of ambient noise seems to improve certain results slightly. This may be due to potential regularization effects of noise and is worth further investigation.

The most detrimental effects happen with the inclusion of the directional interferers, proving that this challenging case will need to be taken into account for future SELD systems. Error rate ER increases up to about 40% in the non reverberant case for the FOA recordings, and up to about 33% for the MIC recordings. Similarly, in reverberant scenarios, the ER increases up to about 28% for both FOA and MIC formats. F-scores decrease by up to 40% on the FOA dataset and up to 50% on the MIC dataset, for both anechoic and reverberant conditions. The localization recall (LR) also drops by about 30% for both formats and both anechoic and reverberant conditions. Finally, localization errors increase by up to about 7° in the case of FOA recordings and up to 10° in the case of MIC recordings, for both anechoic and reverberant conditions. In general, the MIC dataset exhibits a worse performance than FOA in all cases. This fact may be attributed to the input features employed in the baseline for each format. GCC sequences for the MIC format may become very noisy in complex scenes with multiple simultaneous events, while the intensity vectors of the FOA format can potentially retain robustness due to their narrowband nature and sparsity of the event signals in the time-frequency domain.

6. CONCLUSIONS

In this report we describe the new dataset and baseline for the SELD task of the DCASE2021 challenge. The differences with the dataset of DCASE2020 challenge are highlighted; namely, inclusion of directional interferers, higher polyphony, and more frequent simultaneous same-class event occurences. The evaluation task setup is also described, including a predefined fixed split on the development data for straightforward comparison of submissions. The new baseline using the ACCDOA representation shows improved performance compared to the previous one. A detailed analysis of the baseline on various versions of the dataset shows that between reverberation, ambient noise, and directional interferers, the latter has the most detrimental effect in all evaluation metrics.

7. REFERENCES

[1] C. Evers, H. W. Löllmann, H. Mellmann, A. Schmidt, H. Barfuss, P. A. Naylor, and W. Kellermann, "The LOCATA challenge: Acoustic source localization and tracking," *IEEE/ACM Trans. Audio, Speech, and Language Proc.*, vol. 28, pp. 1620–1643, 2020.

[2] J.-M. Valin, F. Michaud, J. Rouat, and D. Létourneau, "Robust sound source localization using a microphone array on a mobile robot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 2, 2003, pp. 1228–1233.

[3] W. He, P. Motlicek, and J.-M. Odobez, "Joint localization and classification of multiple sound sources using a multi-task neural network," in *Interspeech*, 2018, pp. 312–316.

[4] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in *IEEE Conf. on Advanced Video and Signal Based Surveillance*, 2007, pp. 21–26.

[5] H. M. Do, M. Pham, W. Sheng, D. Yang, and M. Liu, "RiSH: A robot-integrated smart home for elderly care," *Robotics and Autonomous Systems*, vol. 101, pp. 74–92, 2018.

[6] O. Brdiczka, J. L. Crowley, and P. Reignier, "Learning situation models in a smart home," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 39, no. 1, pp. 56–63, 2008.

[7] J. Cech, R. Mittal, A. Deleforge, J. Sanchez-Riera, X. Alameda-Pineda, and R. Horaud, "Active-speaker detection and localization with microphones and cameras embedded into a robotic head," in *IEEE/RAS Int. Conf. on Humanoid Robots*, 2013, pp. 203–210.

[8] W. He, P. Motlicek, and J.-M. Odobez, "Deep neural networks for multiple speaker detection and localization," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 74–79.

[9] Y. G. Matsinos, A. D. Mazaris, K. D. Papadimitriou, A. Mniestris, G. Hatzigiannidis, D. Maioglou, and J. D. Pantis, "Spatio-temporal variability in human and natural sounds in a rural landscape," *Landscape ecology*, vol. 23, no. 8, pp. 945–959, 2008.

[10] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE J. Selected Topics in Sig. Proc.*, vol. 13, no. 1, pp. 34–48, 2018.

[11] Y. Cao, Q. Kong, T. Iqbal, F. An, W. Wang, and M. Plumbley, "Polyphonic sound event detection and localization using a two-stage strategy," in *Work. on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2019.

[12] T. N. T. Nguyen, D. L. Jones, and W.-S. Gan, "A sequence matching network for polyphonic sound event localization and detection," in *IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2020, pp. 71–75.

[13] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen, "Overview and evaluation of sound event localization and detection in DCASE 2019," *IEEE/ACM Trans. Audio, Speech, and Language Proc.*, 2020.

[14] A. Politis, S. Adavanne, and T. Virtanen, "A dataset of reverberant spatial sound scenes with moving sources for sound event localization and detection," in *Work. on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 165–169.

[15] K. Shimada, Y. Koyama, N. Takahashi, S. Takahashi, and Y. Mitsufuji, "ACCDOA: Activity-coupled cartesian direction of arrival representation for sound event localization and detection," in *IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2021, pp. 915–919.

[16] H. Phan, L. Pham, P. Koch, N. Q. K. Duong, I. McLoughlin, and A. Mertins, "On multitask loss function for audio event detection and localization," in *Work. on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 160–164.

[17] Q. Wang, J. Du, H.-X. Wu, J. Pan, F. Ma, and C.-H. Lee, "A four-stage data augmentation approach to resnet-conformer based acoustic modeling for sound event localization and detection," *arXiv preprint arXiv:2101.02919*, 2021.

[18] T. N. T. Nguyen, D. L. Jones, and W. S. Gan, "Ensemble of sequence matching networks for dynamic sound event localization, detection, and tracking," in *Work. on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 120–124.

[19] A. Pérez-López and R. Ibáñez-Usach, "Papafil: A low complexity sound event localization and detection method with parametric particle filtering and gradient boosting," in *Work. on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 155–159.

[20] T. N. T. Nguyen, N. K. Nguyen, H. Phan, L. Pham, K. Ooi, D. L. Jones, and W.-S. Gan, "A general network architecture for sound event localization and detection using transfer learning and recurrent neural network," in *IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2021, pp. 935–939.

[21] Y. Cao, T. Iqbal, Q. Kong, Y. Zhong, W. Wang, and M. D. Plumbley, "Event-independent network for polyphonic sound event localization and detection," in *Work. on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 11–15.

[22] Y. Cao, T. Iqbal, Q. Kong, F. An, W. Wang, and M. D. Plumbley, "An improved event-independent network for polyphonic sound event localization and detection," in *IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2021, pp. 885–889.

[23] I. Trowitzsch, J. Taghia, Y. Kashef, and K. Obermayer, "The NIGENS general sound events database," *arXiv preprint arXiv:1902.08314*, 2019.

[24] V. Pulkki, A. Politis, M.-V. Laitinen, J. Vilkamo, and J. Ahonen, "First-order directional audio coding (DirAC)," in *Parametric Time-Frequency Domain Spatial Audio*, V. Pulkki, S. Delikaris-Manias, and A. Politis, Eds. John Wiley & Sons, 2017, pp. 89–138.

[25] A. Mesaros, S. Adavanne, A. Politis, T. Heittola, and T. Virtanen, "Joint measurement of localization and detection of sound events," in *IEEE Work. on Applications of Sig. Proc. to Audio and Acoustics (WASPAA)*, 2019, pp. 333–337.

[26] S. Tervo and A. Politis, "Direction of arrival estimation of reflections from room impulse responses using a spherical microphone array," *IEEE/ACM Trans. Audio, Speech, and Language Proc.*, vol. 23, no. 10, pp. 1539–1551, 2015.

SOUND EVENT LOCALIZATION AND DETECTION BASED ON ADAPTIVE HYBRID CONVOLUTION AND MULTI-SCALE FEATURE EXTRACTOR

Xinghao Sun[1,3], *Ying Hu*[1,3], *Xiujuan Zhu*[1,3], *Liang He*[1,2],

[1] School of Information Science and Engineering, Xinjiang University, Urumqi, China
{xh_sun2019}@stu.xju.edu.cn
[2] Tsinghua National Laboratory for Information Science
and Technology, Department of Electronic Engineering, Tsinghua University, China
[3] Key Laboratory of Signal Detection and Processing in Xinjiang, China

ABSTRACT

Sound event localization and detection (SELD), which jointly performs sound event detection (SED) and sound source localization(SSL), detects the type and occurrence time of sound events as well as their corresponding direction-of-arrival (DoA) angles simultaneously. In this paper, we propose a method based on Adaptive Hybrid Convolution (AHConv) and multi-scale feature extractor. The square convolution shares the weights in each of the square areas in feature maps making its feature extraction ability limited. In order to address this problem, we propose a AHConv mechanism instead of square convolution to capture the dependencies along with the time dimension and the frequency dimension respectively. We also explore a multi-scale feature extractor that can integrate information from very local to exponentially enlarged receptive field within the block. In order to adaptive recalibrate the feature maps after the convolutional operation, we design an adaptive attention block that is largely embodied in the AHConv and multi-scale feature extractor. On TAU-NIGENS Spatial Sound Events 2021 development dataset, our systems demonstrate a significant improvement over the baseline system. Only the first-order Ambisonics (FOA) dataset was considered in this experiment.

Index Terms— DCASE2021, Sound source localization, Sound event detection, Adaptive hybrid convolution

1. INTRODUCTION

Sound Event Localization and Detection refers to the problem of identifying the presence of independent or temporally-overlapped sound sources, correctly identifying to which sound class it belongs, and estimating their spatial directions while they are active. In realistic aural environments, there are numerous co-occurring different sounds emitted from the sources distributed in space. Even humans cannot all correctly identify and locate multiple sources of sound, so it is very challenging for machines. To solve the SELD problem, two key issues denoted as sound event detection (SED) [1–5] and sound source localization (SSL) [6–13] have to be addressed.

The methodology proposed in this paper is based on the SELD-Net proposed by Adavanne et al [14]. A convolutional recurrent neural network (CRNN) model was proposed for joint SSL and SED of multiple overlapping sound events in three-dimensional space. The phase and magnitude of spectrogram were calculated separately on each audio channel as input features. In order to learn both inter-channel and intra-channel features, the input was fed through three consecutive convolutional blocks. Bidirectional Gate Recurrent Unit (BiGRU) was used for temporal context information learning. The output of the BiGRU is fed into two parallel branches of fully-connected blocks. The classes for all sound events would be output on each time-frame, and the sound source would be located in the three-dimensional Cartesian coordinate system.

Compared with DCASE2020 challenge task 3, the main difference is the emulation of scene recordings with a more natural temporal distribution of target events and, more importantly, the inclusion of directional interferences, meaning sound events out of the target classes that are also point-like in nature. For each reverberant environment and every emulated recording, Interferences are spatialized in the same way as the target events, resulting in recordings that are more challenging and closer to real-life conditions. The other difference is the elimination of the dedicated event classification output branch, by adopting the activity-coupled cartesian direction of arrival (ACCDOA) training target which unifies the localization and classification losses in a homogenous regression vector loss, pioneered by Shimada et al [15].

In this paper, we also propose a CRNN framework based on SELD-Net architecture. We adopt Adaptive Hybrid Convolution (AHConv) mechanism and multi-scale feature extractor to handle feature learning insufficiently. The logmel spectrogram and normalized sound intensity vector are extracted as input features. Instead of conventional square convolution, the AHConv structure is design to process richer spatial features and increase feature diversity by asymmetric convolution. We adopt a multi-scale feature to extract strategy that was designed to capture the longer temporal context information than the conventional convolution. Moreover, the parallel structure is applied in adaptive attention block which adaptive mitigates interference between the channel-wise and time-frequency-wise by exploring two different branches. Additionally, the adaptive attention block can also promote the robustness when a single branch is disturbed by the ambient noise without the presence of sound events. Furthermore, we conduct experiments on TAU-NIGENS Spatial Sound Events 2021 development dataset to verify the effectiveness of our proposed method.

This paper is organized as follow: we will introduce the proposed method in Section 2. The experiment setup will be stated in Section 3. The development results compared with the baseline method will be described in Section 4. Finally, we draw a conclusion and future work in Section 5.

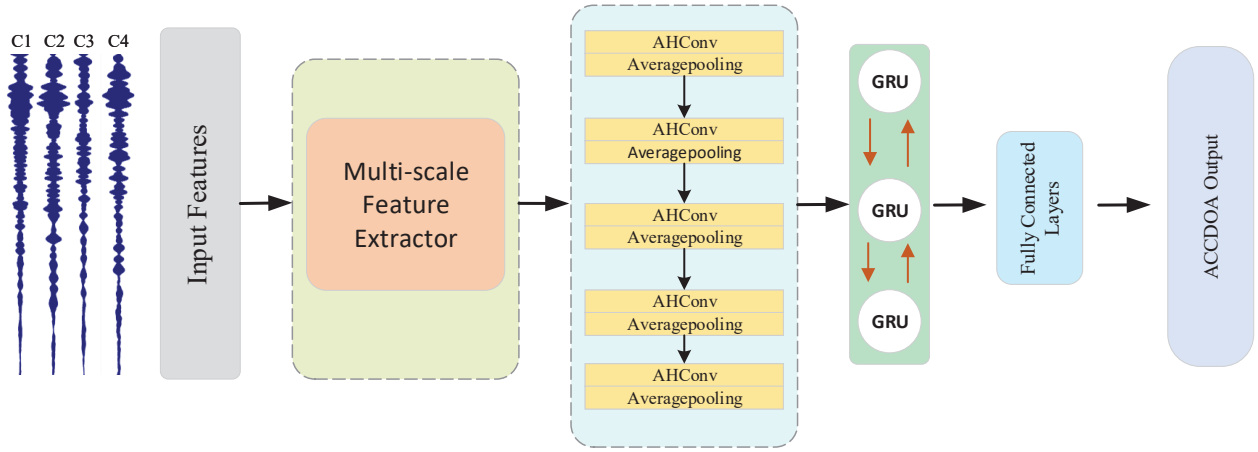


Figure 1: The overall of our proposed method.

2. PROPOSED METHOD

We proposed a method with Adaptive Hybrid Convolution (AHConv) and multi-scale feature extractor which achieves great performance to deal with SELD task in the noisy and reverberant scenes. The proposed network can predict the sound event classes active for each of the input frames along with their respective spatial location, and produce the temporal activity and DOA trajectory for each sound event class. The network diagram is illustrated in Fig. 1. For the multichannel audio, the logmel spectrogram and sound intensity vector are extracted as the input features of the network. The multi-scale feature extractor as depicted in Fig. 2, then followed five AHConv blocks and five average pooling layers. After that, the time dimension is downsampled 5 times and the frequency dimension is downsampled 32 times. Bidirectional Gated Recurrent Unit (Bi-GRU) is used to learn the temporal context information. This is followed by fully connected layers. We adopt the ACCDOA output which unifies the SED and SSL losses into a single homogeneous regression loss.

2.1. Multi-scale Feature Extractor

Among the various CNN architectures, if the network contains shorter connections between layers close to the input and those close to the output, it can be substantially deeper, more accurate, and efficient to train, to further improve the information flow between layers [16]. In this work, we combine the advantages of DenseNet and dilated convolution, and propose an extractor called multi-scale feature extractor. To properly combine DenseNet with the dilated convolution [17], we propose a multi-scale feature extractor that has a multiple dilation factor within a single layer. The dilation rate depends on which skip connection the channels come from, as shown in Fig. 2. The output of each dilated layer is fed into an adaptive attention block. The adaptive attention block reweighs the information of channel-wise and of spatial-wise dimension. That can enhance the important features and weaken the less important features. The outputs of the lth layer x_l receives the feature-maps of all preceding layers express as:

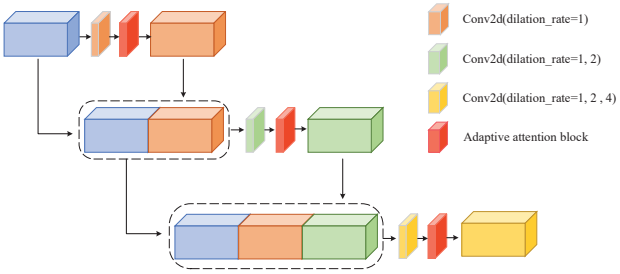


Figure 2: Multi-scale feature extractor, the feature maps of each layer are concatenated together, and the dotted box indicates the concatenate operation

$$x_l = \psi([x_0, x_1, x_2, ...x_{l-1}] \circledast k_l^{d=1,2,...2^{l-1}}) \quad (1)$$

where $[x_0, x_1, x_2, ...x_{l-1}]$ denotes the concatenation of the feature maps from $1, ...l-1$ layers, ψ is a nonlinear transformation consisting of batch normalization (BN) followed by ReLU and dilated convolution with the k_l kernel, $\circledast$ denotes convolution operation and d is the dilated rate in each layer.

2.2. Adaptive Hybrid Convolution

Some of the prior works [18, 19] have shown that a standard square convolutional layer with a filter size of k $\times$ k can be factorized as a sequence of two layers with k $\times$ 1 and 1 $\times$ k filters to reduce network complexity and lighten the computational burden. This asymmetrical convolutional [18] structure is better than a square convolutional structure for processing more and richer spatial features and increasing feature diversity. In addition, asymmetric convolution can obtain faster calculation speed and smaller parameter amounts while ensuring performance. The weight learning of the square convolution relies on the network but is limited by the size of the filter. Therefore, the square convolution is not captured fine-grained time-frequency features. In order to address this problem,

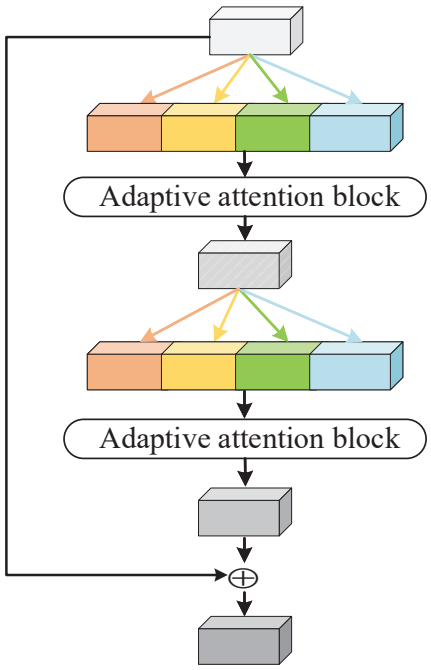


Figure 3: Adaptive Hybrid Convolution (AHConv). Each color represents a different convolution kernel, and the squares represent the convolution graph.

we propose a hybrid convolution mechanism based on the asymmetric convolutional structure, as shown in Fig. 3.

A parallel structure is composed of a filter size 1×3 and 1×5 for time frames, and a filter size 3×1 and 5×1 for frequency bins, thus the time dependency and frequency dependency are capture respectively. Then, the feature maps concatenated along the channel dimension will undergo an adaptive attention block to select the feature adaptively according to the importance. The output of the adaptive attention module will be fed into four asymmetric convolutions and one adaptive convolution again, and the importance of features will be marked more accurately. Finally, in order to preserve the original feature information, we add the original input to the recalibrated output.

2.3. Adaptive Attention block

We design an adaptive attention block as seen in Fig 4. The up half part denotes the path of channel attention (CA) [20], and the lower half part the time-frequency attention (TFA) [21]. In the channel attention path, Global Average Pooling (GAP) converts the information of the TF field of each channel into a value that has the overall information of the channel. To make full use of the aggregated information in the GAP operation, we follow it with fully connected convolution which aims to capture channel-wise dependencies. In the time-frequency attention path, a 2-D convolutional layer with (1,1) kernel size is employed to obtain the global feature maps across the time-frequency (TF) domain. Then sigmoid activation limits the values in the range of (0,1).

After that, different weights are applied to the channel and the TF domain, which can guide the network to pay different attention

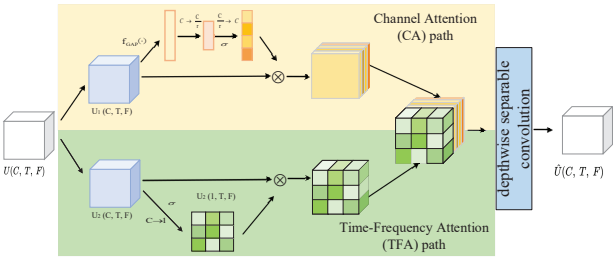


Figure 4: Adaptive attention block.

to the features of channel-wise and time-frequency-wise. The feature maps of each part will concatenate along the channel dimension and then pass through a depthwise separable convolution [22]. The depthwise separable convolution can not only adaptively capture useful information between channels, but also reduce the number of operational parameters. The adaptive attention block is largely embodied in the AHConv and multi-scale feature extractor.

3. EXPERIMENT SETUP

3.1. Dataset

The development set of TAU-NIGENS Spatial Sound Events 2021 has two types of data, one is 4 channel directional microphone array (MIC) from the tetrahedral array and the other one is first-order ambisonic (FOA) data. We used the FOA format for the challenge. The SELD development dataset consists of 600 one-minute audio clips divided into training, validation, and test set of size 400, 100, and 100 clips, respectively. The development dataset is distributed between 12 classes of alarm, crying baby, crash, barking dog, footsteps, knocking on door, female speech, male speech, female scream, male scream, ringing phone and piano. Additionally, dry recordings of disparate sounds not belonging to any of those classes are also spatialized in the same way to serve as directional interference. The sounds are sourced from the running engine, burning fire, and general classes of NIGENS database [23]. The source position for a static event is drawn randomly from the pool of spatial room impulse responses (SRIRs) of a single room used in that recording, while moving events are synthesized for one of the measured trajectories in the room.

3.2. Evaluation metrics

The performance of our proposed model is evaluated by the individual metrics for SED task and SSL task. Standard polyphonic SED metrics, F-score (F1) and error rate (ER) across segments of one second without overlapping are utilized [24]. The DOA estimation in the SSL task was evaluated using frame-wise metrics [25] of DOA error (DE) and frame recall (FR). Considering that a TP is predicted only when the spatial error for the detected event is within the given threshold of 20° deviates from the reference, ER and F1 replaced with ER_{20° and F_{20°. Classification-dependent localization metrics are computed only across each class, instead of across all outputs, DE and FR are replaced with LE_{CD} and LR_{CD}. A more detailed description can be obtained in [25, 26].

3.3. Training procedure

The sampling frequency was used at 24 kHz in our method. Extracting log-mel spectrograms in 64 melbands from 1024-point FFTs, using a 40 ms window and 20 ms hop length. We use a batchsize of 64. Moreover, to ensure a fair comparison, all models were trained for 500 epochs with the Adam optimizer of the same initialized parameters. An early stopping mechanism is used to avoid overfitting during training, where the training is stopped if no improvements on validation split for 50 epochs.

4. RESULT AND DISCUSSION

In this section, we will describe and discuss the experimental results. Firstly, we explored the most appropriate combination of asymmetric convolution for AHConv, and then analyzed the effect of dilated convolution and adaptive attention block in the multi-scale feature extractor with ablation experiments. All the experiments were performed without data augmentation.

Table 1: Explore the combination type of AHConv (+A denotes adding adaptive attention block)

The type of combination	ER_{20°	F_{20°**(%)**	LE_{CD}	LR_{CD}**(%)**
Baseline(3×3)	0.73	30.7	24.5	44.8
1×3,3×1	0.68	42.2	22.6	51.6
(1×3,3×1)+A	0.61	44.7	21.0	54.4
1×5,1×3,3×1,5×1	0.64	43.7	21.9	52.4
(1×5,1×3,3×1,5×1)+A	**0.56**	**46.0**	**20.7**	**55.7**
1×7,1×5,1×3,3×1,5×1,7×1	0.66	43.1	23.1	50.7
(1×7,1×5,1×3,3×1,5×1,7×1)+A	0.58	44.8	20.8	53.3

In order to explore the AHConv in Fig. 3, we performed the experiments without the multi-scale feature extractor. That is the input features of log-mel spectrum and sound intensity vector were directly fed into AHConv. In table. 1, we have explored many combinations of asymmetric convolution. Only using 1×3 and 3×1 can't get enough features on frequency domain and time domain, while using 1×7, 1×5, 1×3, 3×1, 5×1 and 7×1 will degrade performance which may result in too many useless features being captured. The effect is best when the combinations of asymmetric convolution are 1×5, 1×3, 3×1 and 5×1. The results show that this combination of hybrid convolution can fully learn the features of different frequency domains and time domains simultaneously, which is very effective for SELD task. In addition, we also add adaptive attention block to the experiment. The experimental comparison of the same kind of hybrid convolution shows that the performance can be improved by adding the adaptive convolution.

Table 2 shows the results of ablation experiments of our proposed method. The first row denotes the scores of the baseline method. This method is the official baseline system of DCASE 2021 challenge task 3, and all of our experiments are based on it. The second row denotes the scores of the baseline method that adding the multi-scale feature extractor. After the multi-scale feature extractor, the AHConv is replaced by conventional CNN similar to the baseline method. Compared with the results of the first row, the scores in the second row decrease 0.16 and 4.5 on ER_{20° and LE_{CD}, and increase 18.7% and 12.0% on F_{20° and LR_{CD}, respectively. This proves the usefulness of multi-scale feature extractor. The last row denotes the scores of the method that adding multi-scale feature extractor and AHConv. This is the network that we proposed in Fig. 1. Compared with the results of the second row, the scores in the last row further decrease 0.04 and 1.2 on ER_{20° and LE_{CD}, and increase 5.7% and 4.8% on F_{20° and LR_{CD}, respectively. These results verified the effectiveness of the AHConv.

Table 2: The results of ablation experiments

Method	ER_{20°	$F_{20^\circ}(\%)$	LE_{CD}	$LR_{CD}(\%)$
baseline	0.73	30.7	24.5	44.8
$+Extractor$	0.57	49.4	20.0	56.8
$+Extractor + AHConv$	**0.53**	**55.1**	**18.8**	**61.6**

Table 3: The results of exploring the validity of depthwise separable convolution (DSConv)

Method	ER_{20°	$F_{20^\circ}(\%)$	LE_{CD}	$LR_{CD}(\%)$
Conv(1×1)	0.57	52.6	19.6	58.1
DSConv	0.53	55.1	18.8	61.6

In addition, we also explored the effectiveness comparison of conventional 2-D convolution and DSConv in the adaptive attention block. The method in the first row denotes adding a 2-D convolution with 1×1 kernel at the end of each path and then adding it. The second row is using depthwise separable convolution in our proposed method as seen in Fig. 4. Comparing two adaptive methods, the results showed that DSConv performed better.

5. CONCLUSIONS

In this paper, we propose a SELD method based on Adaptive Hybrid Convolution (AHConv) and multi-scale feature extractor. AHConv is designed to capture the time and frequency dependencies. Multi-scale feature extractor is designed to extract the multi-scale feature maps. We also propose an adaptive attention block embodied in AHConv and multi-scale feature extractor. Through a series of ablation experiments on the development dataset, we verify the effectiveness of AHConv and multi-scale feature extractor respectively. The results also show that our proposed method outperforms the baseline method on four evaluation metrics. Next we will introduce data augmentation methods to improve the performance of our proposed method.

6. ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China (NSFC) (61761041,U1903213), Tianshan Innovation Team Plan Project of Xinjiang (202101642)

7. REFERENCES

[1] Y. Li, M. Liu, K. Drossos, and T. Virtanen, "Sound event detection via dilated convolutional recurrent neural networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 286–290.

[2] J. Yan, Y. Song, L.-R. Dai, and I. McLoughlin, "Task-aware mean teacher method for large scale weakly labeled semi-supervised sound event detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 326–330.

[3] L. Lin, X. Wang, H. Liu, and Y. Qian, "Specialized decision surface and disentangled feature for weakly-supervised polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1466–1478, 2020.

[4] H. Wang, Y. Zou, D. Chong, and W. Wang, "Environmental sound classification with parallel temporal-spectral attention," *Proceedings of INTERSPEECH 2020*, 2020.

[5] X. Zheng, Y. Song, J. Yan, L.-R. Dai, I. McLoughlin, and L. Liu, "An effective perturbation based semi-supervised learning method for sound event detection," *Proc. Interspeech 2020*, pp. 841–845, 2020.

[6] S. Chakrabarty and E. A. Habets, "Broadband doa estimation using convolutional neural networks trained with noise signals," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 136–140.

[7] N. Ma, J. A. Gonzalez, and G. J. Brown, "Robust binaural localization of a target sound source by combining spectral source models and deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2122–2131, 2018.

[8] S. Adavanne, A. Politis, and T. Virtanen, "Direction of arrival estimation for multiple sound sources using convolutional recurrent neural network," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 1462–1466.

[9] T. N. T. Nguyen, W.-S. Gan, R. Ranjan, and D. L. Jones, "Robust source counting and doa estimation using spatial pseudo-spectrum and convolutional neural network," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2626–2637, 2020.

[10] Z. Tang, J. D. Kanu, K. Hogan, and D. Manocha, "Regression and classification for direction-of-arrival estimation with convolutional recurrent neural networks," *Proc. Interspeech 2019*, pp. 654–658, 2019.

[11] H. Sundar, W. Wang, M. Sun, and C. Wang, "Raw waveform based end-to-end deep convolutional network for spatial localization of multiple acoustic sources," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4642–4646.

[12] L. Perotin, R. Serizel, E. Vincent, and A. Guérin, "Crnn-based multiple doa estimation using acoustic intensity features for ambisonics recordings," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 1, pp. 22–33, 2019.

[13] W. He, P. Motlicek, and J.-M. Odobez, "Adaptation of multiple sound source localization neural networks with weak supervision and domain-adversarial training," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 770–774.

[14] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 1, pp. 34–48, 2018.

[15] K. Shimada, Y. Koyama, N. Takahashi, S. Takahashi, and Y. Mitsufuji, "Accdoa: Activity-coupled cartesian direction of arrival representation for sound event localization and detection," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 915–919.

[16] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[17] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[18] X. Ding, Y. Guo, G. Ding, and J. Han, "Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1911–1920.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[20] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[21] W. Xia and K. Koishida, "Sound event detection in multichannel audio using convolutional time-frequency-channel squeeze and excitation," *Proc. Interspeech 2019*, pp. 3629–3633, 2019.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[23] I. Trowitzsch, J. Taghia, Y. Kashef, and K. Obermayer, "The nigens general sound events database," *arXiv preprint arXiv:1902.08314*, 2019.

[24] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[25] A. Mesaros, S. Adavanne, A. Politis, T. Heittola, and T. Virtanen, "Joint measurement of localization and detection of sound events," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 333–337.

[26] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen, "Overview and evaluation of sound event localization and detection in dcase 2019," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

ON THE EFFECT OF CODING ARTIFACTS ON ACOUSTIC SCENE CLASSIFICATION

Nagashree K. S. Rao[1,2], *Nils Peters*[1,2]

[1] University of Erlangen-Nuremberg, Erlangen, Germany
[2] International Audio Laboratories, Erlangen, Germany
{nagashree.s.rao, nils.peters}@fau.de

ABSTRACT

Previous DCASE challenges contributed to an increase in the performance of acoustic scene classification systems. State-of-the-art classifiers demand significant processing capabilities and memory which is challenging for resource-constrained mobile or IoT edge devices. Thus, it is more likely to deploy these models on more powerful hardware and classify audio recordings previously uploaded (or streamed) from low-power edge devices. In such scenario, the edge device may apply perceptual audio coding to reduce the transmission data rate. This paper explores the effect of perceptual audio coding on the classification performance using a DCASE 2020 challenge contribution [1]. We found that classification accuracy can degrade by up to 57% compared to classifying original (uncompressed) audio. We further demonstrate how lossy audio compression techniques during model training can improve classification accuracy of compressed audio signals even for audio codecs and codec bitrates not included in the training process.

Index Terms— acoustic scene classification, data augmentation, audio coding, internet of things

1. INTRODUCTION

As it can be observed in the annual DCASE challenges, classification models to understand complex acoustic soundfields are becoming increasingly robust and accurate. The evaluation scenario indirectly presumes that scene classification is performed on the capturing device. However, many mobile and IoT devices are resource-constrained and may not be able to store and execute all proposed classification architectures due to limited memory and processing capabilities. Such models can be executed on more powerful personal companion devices or remote cloud servers. Figure 1 visualizes three scenarios where the audio capture and the scene classification are split between different devices. In such scenarios, the captured audio signals would be uploaded or streamed from the capture device to the scene classification device. The service limitations of the (often wireless) network are met by utilizing perceptual audio codecs. Perceptual audio coding can significantly reduce the data rate of an audio signal for storage or transmission by removing imperceivable signal components based on psychoacoustic principles [2]. Compared to the original audio material, perceptual audio coding does not intend to preserve the signal waveform, it rather aims to maintain a perceptually similar or even equal audio experience. The transcoding of an audio signal from one audio codec to another when passing through a network may introduce further coding artifacts. Given the maturity of perceptual audio coding, one can presume that today's perceptual audio codecs do not affect the human ability to classify acoustic scenes. But is this also true for state-of-the-art classification algorithms? In this contribution, we will study the effect of lossy perceptual coding on the model accuracy. To our knowledge, this may be the first study in the context of acoustic scene classification.

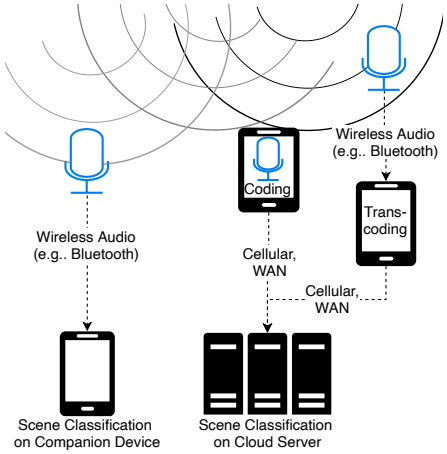


Figure 1: DCASE application scenarios where acoustic sensing and classification are decoupled and signals are coded for data transmission.

The paper is organized as follow: We start with introducing the classification architecture we use as a basis for this study. In Section 2 we present an experiment where the overall accuracy of the pre-trained reference model is evaluated as a function of differently coded audio files. Section 3 reports on a series of experiments with the goal to improve the model accuracy by various training conditions. Before we conclude the paper we summarize and discuss our findings in Section 4.

1.1. Hu et al. DCASE 2020 Classification Model

Thanks to the authors of [1] who made source code and pre-trained models publicly available, we are able to study the effect of audio coding artifacts on one of the best performing models of the DCASE 2020 Scene Classification Challenge. For Task 1a, this proposal was ranked as the third best architecture, achieving a model accuracy of 76.2% on the secret 10-class evaluation dataset[1]. With a total of 130 million parameters, this architecture would likely be deployed on cloud servers rather than on resource-limited edge devices.

[1] taken from challenge results at http://dcase.community/challenge2020/task-acoustic-scene-classification-results-a

This architecture (see Figure 2) consists of two groups of ensemble classifiers namely 3-class and 10-class. The 10-class models classify the input features (Log-mel filterbank features) to one of the ten original scene classes: airport, shopping mall, metro station, pedestrian street, public square, street traffic, tram, bus, metro, and park. The 3-class classifier is trained to predict one of the three categories: indoor, outdoor, and transportation. The final scene class is estimated by score fusion of the 3-class and the 10-class classifier.

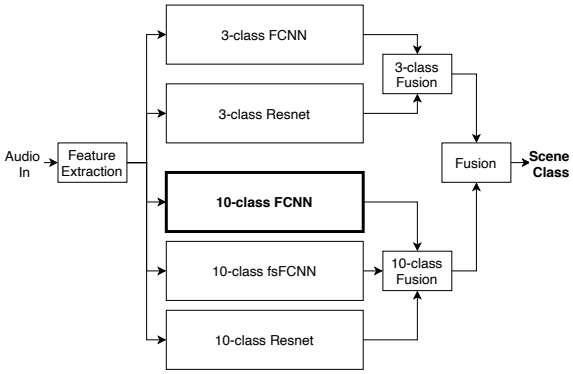


Figure 2: The scene classification ensemble architecture by [1]

.

2. EXPERIMENT A

In this experiment we want to explore whether the classification accuracy in the classification architecture described in Section 1.1 changes when inference is done on perceptually coded audio data rather than on original (uncompressed) audio.

2.1. Methodology

For this experiment we evaluate the complete pre-trained ensemble classifier for DCASE 2020 Task 1a by the authors of [1] using the official evaluation split from the DCASE 2020 development set [3]. Figure 3 visualizes the data flow of this experiment. The evaluation split consists of 2968 monaural audio files at 44.1 kHz, each 10 seconds long. For the sake of reproducibility, all audio files are encoded using [4] with the perceptual audio codecs listed in Table 1. The codecs are chosen because of their support in popular mobile operating systems and thus, likely to be used for compressing recorded signals prior scene classification. As model input log-mel filterbank features are extracted from the decoded audio signals. The evaluation compares the classification result with the ground truth class labels of the dataset.

Table 1: Overview of perceptual audio codecs used in this paper

Codec	Reference	Bit Rates [kbps]	
		Exp A	Exp B
AAC (LC)	[5]	64	32, 48, 64
HE-AAC (v1)	[6]	32	16, 32
MP3	[7]	32, 64	32, 64
Opus	[8]	32	64
SBC	[9]	64	64

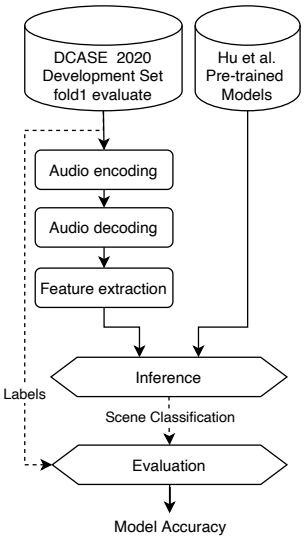


Figure 3: Exp A: Evaluation of pre-trained models with coded audio

2.2. Results

Table 2 summarizes the results of Experiment A. We found significant degradation of the overall model performance when scene classification is performed on previously compressed audio data. In our experiments, the performance of the model accuracy could drop from 0.820 (classification accuracy on the uncompressed original evaluation files) to 0.352, a decrease by 57.1%. Not surprising, audio compression using higher bit rates (e.g., AAC at 64 kbps) tends to achieve a better model performance.

To understand if the model accuracy changes with the perceptual audio quality due to audio coding, we computed the PEAQ objective difference grade (ODG) [10] between the uncompressed and the previously compressed audio files. For a comparative analysis of PEAQ and other quality metrics see [11]. We computed the average ODG across all evaluation files per codec under test and compare it with the model accuracy from Table 2. As visible in Figure 4 a high ODG (i.e., little coding artifacts) results in a good model performance. However, for lower ODGs (more coding artifacts), this relationship vanishes: Two codecs that achieved an ODG around -3.1 result in a very different classification accuracy (0.61 vs. 0.35). Further, the model accuracy is also not a reliable predictor for PEAQ's ODG: A scene classification accuracy of about 0.65 could be achieved for codecs with an ODG between -3.1 and -2.1. In summary, the scene classification performance correlates only weakly with PEAQ's estimated perceptual audio quality ($R^2 = .44$).

3. EXPERIMENT B

Based on the results of Experiment A and to improve the scene classification performance of previously compressed audio data, we propose to retrain the model using additional data augmentation. In this series of experiments, we focus on the 10-class FCNN classification sub model (see highlight in Figure 2). These experiments also contribute to a better understand of the trade-offs between executing low-complexity classification models on the edge using un-

Table 2: Exp A: Model accuracy for differently coded signals

Codec	Bit Rate [kbps]	Model Acc.	Relative Decrease
None (Original)	1058	.820	N/A
AAC	64	.741	9.6 %
MP3	64	.724	11.7 %
Opus	32	.691	15.7 %
HE-AAC	32	.653	20.4 %
SBC	64	.632	22.9 %
MP3	32	.352	57.1 %

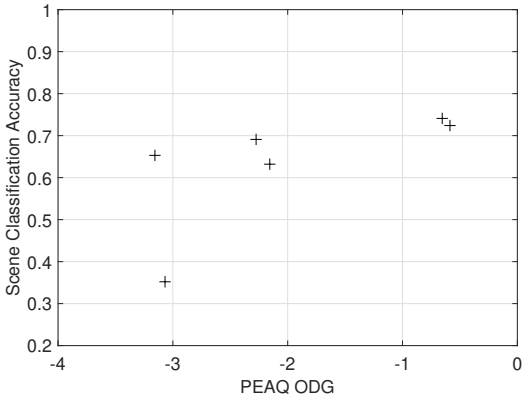


Figure 4: Experiment A: Relationship between PEAQ ODG and model accuracy for differently coded signals

compressed audio vs. executing complex DCASE models on a cloud server using compressed audio.

3.1. Methodology

The 10-class FCNN sub model is retrained with different sets of pre-compressed audio data. The workflow is shown in Figure 5. All hyperparameters for the training are kept as in [1] to allow for comparison across experiments. First, augmentation data are generated by converting the training dataset into different encoded audio formats and bitrates [4]. To decide which codec configurations to use for the training, we analysis the log-mel features across uncompressed and previously compressed audio data. The aim of this analysis was to maximize the differences in the feature data via a subset of audio codec configurations. As a result of this analysis, MP3 at 64kbps, AAC at 32kbps and HE-AAC at 16kbps and 32kbps were selected.

The newly trained models are evaluated with the same audio content as in Experiment A in the following conditions: First, the original evaluation data are used to verify that the re-training has no degrading effects on the classification of uncoded audio data. Then, a subset of the evaluation conditions use codecs at bit rates that were part of the training (i.e., MP3 at 64 kbps, AAC at 32 kbps), other conditions feature the same codec used for training, but at different bit rates (i.e., MP3 at 32 kbps, AAC at 48 and 64 kbps). In the final two conditions, audio codecs that were not part of the model training are applied (i.e., Opus at 64 kbps, SBC at 64 kbps).

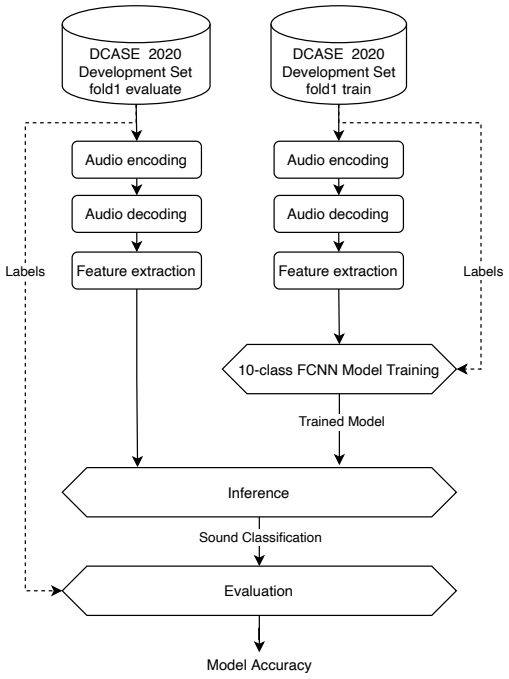


Figure 5: Exp B: Model training and evaluation with coded audio

3.2. Results

The results of experiment B are summarized in the Table 3. From the first row in Table 3 we can see that accuracy of the model when trained with no augmentation data and evaluated with compressed audio codecs was poor compared to the accuracy with the original audio data. When the model was trained using the data augmentation methods as proposed in [1] (essentially our baseline), the accuracy when evaluated with the original files increased moderately from 0.703 to 0.721, but the accuracy was still comparatively lower when evaluated with coded signals (e.g., 0.301 for $MP3_{32}$). Notably, as we start including coded audio files in the training, the performance for the codec evaluation conditions improved significantly. With inclusion of MP3 files at 64 kbps, there was decrease in the accuracy when evaluated with original files but the performance of the model improved when evaluated with coded files e.g., AAC files and MP3 files at 64 kbps. The variation of the accuracy when the model was evaluated with different codec conditions is discussed below:

No Coding condition: The classification performance of uncoded audio decreased when the $MP3_{64}$ files were included in the training. The performance improved gradually with inclusion of $HE\text{-}AAC_{16}$ and AAC_{32} and further increased when the model was trained in the final training condition (fully augmented dataset as in [1], $MP3_{64}$, $HE\text{-}AAC_{16}$, and AAC_{32}). Here, the accuracy (0.72) was comparable to the baseline [1], suggesting that additional data augmentation using perceptual audio codecs does not degrade the classification of original audio signals.

Seen Codec With Bitrate conditions (AAC_{32}, $MP3_{64}$): The performance increased with the baseline training from 0.458 to 0.558 and from 0.550 to 0.615 respectively. The accuracy further improved with the addition of $MP3_{64}$, and HE-AAC. The highest

Table 3: Experiment B: Model accuracy for different training conditions evaluated with different types of previously compressed signals. The subscript in the codec name indicates the bitrate in kbps

	Training		Codec for Evaluation							
Index	**Condition**	**File count**	**None**	**AAC_{32}**	**$MP3_{64}$**	**$MP3_{32}$**	**AAC_{48}**	**AAC_{64}**	**$Opus_{64}$**	**SBC_{64}**
1	None (only original training data)	13962	.703	.458	.550	.261	.477	.525	.587	.391
2	Fully augmented data set as in [1]	121911	**.721**	.558	.615	.301	.573	.622	.638	.555
3	1+$MP3_{64}$	27924	.668	.566	.635	.249	.602	.630	.556	.363
4	3 + HE-AAC_{16}	41886	.696	.631	.662	.477	.638	.651	.587	.534
5	3 + HE-AAC_{32}	41886	.699	.630	.666	.319	.648	.663	.593	.508
6	4 + AAC_{32}	55848	.697	**.673**	.675	.561	.664	.671	.610	.560
7	2 + 6	163797	.720	.670	**.685**	**.598**	**.685**	**.690**	**.650**	**.589**
	relative performance increase from 2 [%]		-0.1	20.1	11.4	98.7	19.6	10.9	1.9	6.1

achieved accuracy for the $MP3_{64}$ condition was 0.685 when the model was trained with the baseline augmentation data [1] as well as $MP3_{64}$, AAC_{32}, and HE-AAC_{16} as additional training data. The highest accuracy for the AAC_{32} condition (0.673) was obtained when the model was trained by including the HE-AAC_{16} and AAC_{32} training data. The relative performance for the classes AAC_{32} and $MP3_{64}$ improved by 20.1% and 11.4% respectively, as compared to the baseline performance.

Unseen Bitrate conditions ($MP3_{32}$, AAC_{48}, AAC_{64}): Here, the models were trained with MP3 and AAC but evaluated at different bitrates. Initially, the accuracy of these conditions was low: 0.261, 0.477, and 0.525 respectively. In case of the condition $MP3_{32}$ the performance of the model improved with the inclusion of HE-AAC at 16kbps and AAC at 32 kbps training data. The model accuracy for both AAC evaluation conditions improved for every audio codec added to the training. The relative performance of classes $MP3_{32}$, AAC_{48}, AAC_{64} improved by 98.7%, 19.6% and 10.9% respectively, as compared to the baseline performance.

Unseen Codec conditions ($Opus_{64}$, SBC_{64}): Opus and SBC were not used in training. The performance of $Opus_{64}$ initially did not improve when augmented with different coded files, but when the model was trained with fully augmented dataset as in [1] and $MP3_{64}$, HE-AAC_{16}, and AAC_{32}, there was a slight improvement in the performance by 1.9%. The performance of SBC_{64} decreased with inclusion of $MP3_{64}$ as augmentation data but improved by the inclusion of HE-AAC_{16} and AAC_{64} as augmentation file. The relative classification accuracy for SBC_{64} improved by 6.1% compared to the baseline performance.

In summary the final training condition resulted in an average increase in classification accuracy by 24.1% across all 7 codec conditions. Moreover, at the final training condition, every evaluated codec condition achieved now at least 81% of the classification accuracy as if original audio data would have been inferred. Compared to the initial training method this is a significant improvement in the robustness across codec conditions.

4. DISCUSSION AND FUTURE WORK

As a result of our experiments, we can state that perceptual compression artifacts can significantly degrade the accuracy of today's scene classification models. Including perceptual audio coding in the data augmentation strategy:

1. does neither harm nor improve model performance when classification is performed on the original audio data.
2. improves model accuracy when the inferred signals have been perceptually coded.
3. can improve model accuracy even when the inferred signals have been perceptually coded with an unseen codec or from a seen codec with an unseen bitrate configuration.
4. seems to better harmonize the classification accuracy for input signals with unknown coding history.

Whereas we used the coding framework [4] for the sake of reproducibility, results may differ when other encoder implementations are used. Also, since we studied one recent scene classification architecture, our results may not translate perfectly to other systems. However, since many architectures use log-mel filterbank input features, we believe our findings generally hold for those architectures.

Although our proposed data augmentation strategy is working, it generally increases the training time due to the additional training data and requires retraining of existing models. Thus, it would be interesting to consider alternative approaches to improve robustness against perceptual coding artifacts, e.g., hyperparameter optimization, or transfer learning [12].

5. CONCLUSION

We demonstrated how lossy perceptual audio coding can degrade the scene classification accuracy of a state-of-the-art system by up to 57% compared to uncompressed audio captures, depending on audio codec and bit rate. To increase robustness against compression artifacts, we propose a data augmentation strategy for model training that includes perceptual audio coding. We showed that such strategy can increases the accuracy when classifying perceptually coded signals even for audio codecs and/or bitrates not part of the model training. These findings are beneficial to improve scene classification robustness whenever audio signals are subject to perceptual audio coding, e.g., due to transmission or lossy data storage.

6. ACKNOWLEDGMENT

The International Audio Laboratories Erlangen are a joint institution between the University of Erlangen-Nuremberg and Fraunhofer IIS. We would like to thank the authors of [1] for making their architecture publicly available and especially Chao-Han Huck Yang and Hu Hu for their support in executing the software.

7. REFERENCES

[1] H. Hu, C.-H. H. Yang, X. Xia, X. Bai, X. Tang, Y. Wang, S. Niu, L. Chai, J. Li, H. Zhu, *et al.*, "Device-robust acoustic scene classification based on two-stage categorization and data augmentation," 2020, [Online:] https://arxiv.org/abs/2007.08389, [Source Code:] https://github.com/MihawkHu/DCASE2020_task1.

[2] T. Painter and A. Spanias, "Perceptual coding of digital audio," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 451–515, 2000.

[3] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in DCASE 2020 challenge: generalization across devices and low complexity solutions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020. [Online]. Available: https://arxiv.org/abs/2005.14623

[4] [Online]. Available: https://www.ffmpeg.org

[5] R. Geiger, R. Yu, J. Herre, S. Rahardja, S.-W. Kim, X. Lin, and M. Schmidt, "ISO/IEC MPEG-4 high-definition scalable advanced audio coding," *Journal of the Audio Engineering Society*, vol. 55, no. 1/2, pp. 27–43, 2007.

[6] J. Herre and M. Dietz, "MPEG-4 high-efficiency AAC coding," *IEEE Signal Processing Magazine*, vol. 25, no. 3, pp. 137–142, 2008.

[7] K. Brandenburg, "MP3 and AAC explained," *Journal of the Audio Engineering Society*, September 1999.

[8] J.-M. Valin, G. Maxwell, T. B. Terriberry, and K. Vos, "High-quality, low-delay music coding in the Opus codec," *arXiv preprint arXiv:1602.04845*, 2016.

[9] Bluetooth Audio Video Working Group and others, "Bluetooth specification: Advanced audio distribution profile," *Bluetooth SIG Inc*, 2002.

[10] T. Thiede, W. C. Treurniet, R. Bitto, C. Schmidmer, T. Sporer, J. G. Beerends, and C. Colomes, "PEAQ-the ITU standard for objective measurement of perceived audio quality," *Journal of the AES*, vol. 48, no. 1/2, pp. 3–29, 2000.

[11] M. Torcoli and S. Dick, "Comparing the effect of audio coding artifacts on objective quality measures and on subjective ratings," in *144th AES Convention*, 2018.

[12] A. I. Mezza, E. A. P. Habets, M. Müller, and A. Sarti, "Feature projection-based unsupervised domain adaptation for acoustic scene classification," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Espoo, Finland, 2020.

CONTINUAL LEARNING FOR AUTOMATED AUDIO CAPTIONING USING THE LEARNING WITHOUT FORGETTING APPROACH

Jan Berg and Konstantinos Drossos

Audio Research Group, Tampere University, Finland
{firstname.lastname}@tuni.fi

ABSTRACT

Automated audio captioning (AAC) is the task of automatically creating textual descriptions (i.e. captions) for the contents of a general audio signal. Most AAC methods are using existing datasets to optimize and/or evaluate upon. Given the limited information held by the AAC datasets, it is very likely that AAC methods learn only the information contained in the utilized datasets. In this paper we present a first approach for continuously adapting an AAC method to new information, using a continual learning method. In our scenario, a pre-optimized AAC method is used for some unseen general audio signals and can update its parameters in order to adapt to the new information, given a new reference caption. We evaluate our method using a freely available, pre-optimized AAC method and two freely available AAC datasets. We compare our proposed method with three scenarios, two of training on one of the datasets and evaluating on the other and a third of training on one dataset and fine-tuning on the other. Obtained results show that our method achieves a good balance between distilling new knowledge and not forgetting the previous one.

Index Terms— Automated audio captioning, continual learning, learning without forgetting, WaveTransformer, Clotho, AudioCaps

1. INTRODUCTION

Automated audio captioning (AAC) is the inter-modal translation task, where a method takes as an input a general audio signal and generates a textual description of the contents of the audio signal [1]. AAC methods learn to describe sound sources/events, spatiotemporal relationships of events, textures and sizes, and higher-level knowledge like counting [1, 2], but not speech transcription [3, 4]. In a typical AAC scenario, a deep learning method is optimized in a supervised or reinforcement learning scheme and using an AAC dataset [5, 6, 7, 8, 9]. Audio clips are given as an input to the AAC method and the method generates captions for its inputs. Then, the method is optimized by trying to reduce the difference between the predicted and the actual (i.e ground truth) captions. Given that the existing AAC datasets are limited, the above scheme creates some limitations. For example, since the available information from the audio clips in the different datasets are most likely not overlapping and the described information and expression variability differs given that different annotators have been used [3, 10], then an AAC method optimized with one dataset will have problems when evaluated with another AAC dataset. Even if some technique is used for adapting an AAC method to another dataset, e.g. like transfer learning, it would be required to have all the new data for the adaptation. This creates limitation of continuously adapting an AAC method to new information.

The above presented problem of continuously adapting is not new and has been attacked using continual learning, sometimes also called lifelong learning [11, 12], which is the process of continuously adapting a method to new data and/or tasks. The advantage of continual learning over other techniques, e.g. transfer learning, is that the latter usually introduces the phenomenon of catastrophic forgetting, where the method is adapted to the new information but forgets the initially learned one [11, 13, 14, 15]. Though, continual learning methods seem that tackle this phenomenon [14, 15]. There are different approaches for continual learning, e.g. like joint training [12], though our focus is on the cases where the new data are not required a priori, because it is often not possible to have all data beforehand due to storing reasons (e.g. cannot store all the data) or to degradation of data (e.g. data have been lost over time). Approaches that do not require having the data to do the adaptation, can be roughly divided into three categories [11], namely regularization methods like learning without forgetting (LwF) [15] and elastic weight consolidation (ECW) [14], dynamic architectures like dynamically expandable networks (DEN) [16], and replay models like gradient episodic memory (GEM) [17].

In this paper we consider the scenario where an AAC method continuously adapts to new and unseen data, using unseen ground truth captions. This scenario can resemble, for example, an online platform where new audio data and captions can be provided by human users and the AAC method continuously learn from the new data. Focusing on this, we present a first method for continual learning for AAC, adopting the LwF approach. Although there are published continual learning approaches for audio classification using different approaches [18, 19], we employ LwF due to its simplicity, reduced need for resources, and the facts that LwF is model agnostic and no modifications are needed for the employed AAC model. Although, previous research has shown that one of the weaknesses of LwF is that its effectiveness is dependent on the similarity of the tasks at hand [20, 11, 21], we deem that this is not applicable to our case since we use LwF for continuously adapting to new data on the same task.

For our work presented here, we employ a freely available and pre-optimized AAC method called WaveTransformer (WT) [22] and two freely available AAC datasets, namely Clotho [3] and AudioCaps [10]. Since WT method has achieved state-of-the-art results on Clotho, we use AudioCaps as the new data that the AAC method will adapt. Given that there are no other published continual learning approaches for AAC, in this paper we do not consider the case of the mismatched set of words in the two employed datasets. The rest of the paper is organized as follows. Section 2 presents our method and Section 3 presents the adopted evaluation process. Obtained results are in Section 4 and Section 5 concludes the paper.

2. METHOD

Our method is model agnostic, based on LwF and knowledge distillation [15, 23]. It employs a pre-optimized AAC model, a copy of the AAC model, an iterative process, a regularization-based loss, and a stream of new audio data with captions that are used for learning the new information. The stream of new audio data and captions is used to provide input to the original and copy AAC models. The output of both models is used against the provided captions from the stream, but only the parameters of the copy model is updated. At every update of the parameters, the copy model can be used as an output of our continual learning method. An illustration of our continual learning approach is in Figure 1.

In more detail, we start by having a pre-optimized AAC model $\mathrm{M}_{\text{base}}(\cdot;\theta_{\text{base}})$, having the pre-optimized parameters θ_{base}. M_{base} is pre-optimized using a dataset of K input-output examples $\mathbb{D}_{\text{ori}} = \{(\mathbf{X}', \mathbf{Y}')_k\}_{k=1}^{K}$, where $\mathbf{X}' \in \mathbb{R}^{T_{\text{a}} \times F}$ is a sequence of T_{a} audio feature vectors having F features and $\mathbf{Y}' \in \{0,1\}^{T_{\text{w}} \times W}$ a sequence of T_{w} one-hot encoded vectors of W elements, that indicate the most probable word for each t_{w}-th word index. M_{base} generates its output as

$$\hat{\mathbf{Y}}'_k = \mathrm{M}_{\text{base}}(\mathbf{X}'_k; \theta_{\text{base}}), \tag{1}$$

where $\hat{\mathbf{Y}}'_k$ is the predicted caption by M_{base} when having as input the $\mathbf{X}'_k$. The optimization of θ_{base} is performed by minimizing the loss

$$\mathcal{L}(\theta_{\text{base}}, \mathbb{D}_{\text{ori}}) = \sum_{k=1}^{K} \mathrm{CE}(\mathbf{Y}'_k, \hat{\mathbf{Y}}'_k), \tag{2}$$

where CE is the cross-entropy loss between $\mathbf{Y}'_k$ and $\hat{\mathbf{Y}}'_k$.

Then, we create a copy of M_{base}, $\mathrm{M}_{\text{new}}(\cdot;\theta_{\text{new}})$, having same hyper-parameters as M_{base} and the parameters θ_{new}. Our target is to continuously update θ_{new} for new data, without making M_{new} to deteriorate its performance on $\mathbb{D}_{\text{ori}}$. The new data are coming from a stream of data, $\mathcal{S}$, which continually produces new and unseen data (i.e. data not in $\mathbb{D}_{\text{ori}}$). We sample data from $\mathcal{S}$ in batches of B examples, creating the input-output examples as

$$\mathbb{D}_{\text{new}} = \{(\mathbf{X}, \mathbf{Y})_b : (\mathbf{X}, \mathbf{Y}) \sim \mathcal{S} \wedge b = 1, \ldots, B\}, \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{T_{\text{a}} \times F}$ is a sequence of audio features, similar to $\mathbf{X}'$, and $\mathbf{Y} \in \{0,1\}^{T_{\text{w}} \times W}$ is a sequence of one-hot encoded vectors similar to $\mathbf{Y}'$. Here has to be noted that the captions coming from $\mathcal{S}$ can (and most likely will) have different set of words with $\mathbf{Y}'$. Though, our approach is not considering the problem of the different set of words. For that reason, we consider from $\mathbf{Y}$ only the words that are common with $\mathbf{Y}'$.

We use the sampled data $\mathbb{D}_{\text{new}}$ as an input to both M_{base} and M_{new}, resulting to

$$\hat{\mathbf{Y}}_b^{\text{base}} = \mathrm{M}_{\text{base}}(\mathbf{X}_b; \theta_{\text{base}}), \text{ and} \tag{4}$$

$$\hat{\mathbf{Y}}_b^{\text{new}} = \mathrm{M}_{\text{new}}(\mathbf{X}_b; \theta_{\text{new}}), \tag{5}$$

where $\hat{\mathbf{Y}}_b^{\text{base}}$ and $\hat{\mathbf{Y}}_b^{\text{new}}$ are the predicted outputs of M_{base} and M_{new}, respectively, when having as an input $\mathbf{X}_b$.

Having $\hat{\mathbf{Y}}_b^{\text{base}}$ and $\hat{\mathbf{Y}}_b^{\text{new}}$, we define the loss

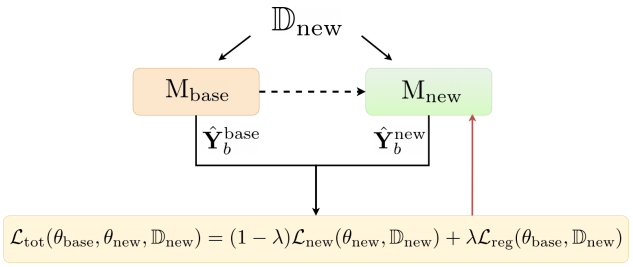


Figure 1: Our proposed continual learning method for AAC. The dotted line represents the copying of the parameters of M_{base} to M_{new}, and it takes place only once at the beginning of the process. Red line indicates backpropagation for updating the parameters of M_{new}.

$$\mathcal{L}_{\text{tot}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}) = (1-\lambda)\mathcal{L}_{\text{new}}(\theta_{\text{new}}, \mathbb{D}_{\text{new}}) + \lambda\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \mathbb{D}_{\text{new}}), \text{ where} \tag{6}$$

$$\mathcal{L}_{\text{new}}(\theta_{\text{new}}, \mathbb{D}_{\text{new}}) = \sum_{b=1}^{B} \mathrm{CE}(\mathbf{Y}_b, \hat{\mathbf{Y}}_b^{\text{new}}), \tag{7}$$

$$\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}) = \sum_{b=1}^{B} \mathrm{KL}(\hat{\mathbf{Y}}_b^{\text{base}}, \hat{\mathbf{Y}}_b^{\text{new}}), \text{ and} \tag{8}$$

λ is a factor that weights the contribution of $\mathcal{L}_{\text{new}}$ and $\mathcal{L}_{\text{reg}}$ to $\mathcal{L}_{\text{tot}}$, and $\mathrm{KL}(a, b)$ is the KL-divergence between a and b. We use λ in order to balance the learning of the new information and the non-forgetting of the old information. The non-forgetting is implemented with the $\mathcal{L}_{\text{reg}}$, where the predictions of M_{new} are sought to be as similar to the predictions of M_{base}.

Finally, after calculating the $\mathcal{L}_{\text{tot}}$ for each sampling of data from $\mathcal{S}$, we obtain new optimized parameters for M_{new} as

$$\theta^{\star}_{\text{new}} = \underset{\theta_{\text{new}}}{\operatorname{argmin}}\ \mathcal{L}_{\text{tot}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}), \tag{9}$$

where $\theta^{\star}_{\text{new}}$ are the new, optimized parameters. After obtaining $\theta^{\star}_{\text{new}}$, we update θ_{new} as

$$\theta_{\text{new}} = \theta^{\star}_{\text{new}}. \tag{10}$$

Thus, M_{new} is updated with the new information and also remembers old learned information, after applying (10). The iterative process of our continual method for AAC is the process described by Equations (3) to (10). The result of our method is the M_{new} after the application of Eq. (10).

3. EVALUATION

In order to evaluate our method, we use a freely available and pre-optimized method as our M_{base} and a freely available dataset different from $\mathbb{D}_{\text{ori}}$ to simulate $\mathcal{S}$, namely WaveTransformer (WT) and AudioCaps, respectively. $\mathbb{D}_{\text{ori}}$ used for WT is Clotho. We use mini-batches of size B from AudioCaps to simulate $\mathbb{D}_{\text{new}}$, using only one epoch over AudioCaps. The performance of the continual learning is evaluated using metrics adopted usually in AAC task. Our code used for the implementation of our method can be found online[1].

[1] https://github.com/JanBerg1/AAC-LwF

3.1. Datasets and pre-processing

Clotho [3] is a freely available dataset for AAC, containing 3840 audio clips for training, 1046 for validation, and 1046 for evaluation. Each audio clip is of 15-30 seconds long and is annotated with five captions of eight to 20 words. This results to 19 200, 5230, and 5230 input-output examples for training, validating, and evaluating an AAC method, respectively. AudioCaps [10] is also a freely available AAC dataset, based on AudioSet [24]. AudioCaps has 38 118 audio clips for training, 500 for validation, and 979 for testing. All audio clips are 10 seconds long, and clips for training are annotated with one caption while clips for validation and testing with five captions. These result to 38 118, 2500, and 4895 input-output examples for training, validating, and evaluating, respectively. In all experiments, as $\mathbb{D}_{\text{ori}}$ we use the training split of the corresponding dataset and as $\mathbb{D}_{\text{new}}$ the training split from the other AAC dataset. During the stage of hyper-parameter tuning we used as the validation split from $\mathbb{D}_{\text{ori}}$ and $\mathbb{D}_{\text{new}}$ to evaluate the performance of our method, while during testing we used the evaluation split as $\mathbb{D}_{\text{new}}$, from the corresponding dataset. These result to $K = 19200$ for Clotho and $K = 38118$ for AudioCaps.

From all audio clips we extract $F = 64$ log mel-band energies, using a 46 second long Hamming window with 50% overlap. This results to $1292 \leq T_{\text{a}} \leq 2584$ for Clotho and $T_{\text{a}} = 862$ for AudioCaps. Additionally, for Clotho there are $8 \leq T_w \leq 20$ words in a caption and there are $W = 4367$ unique words, while for AudioCaps there are $2 \leq T_w \leq 51$ words in a caption and there are $W = 4506$ unique words. But, when M_{base} is optimized on either Clotho or AudioCaps the M_{new} is evaluated at the other dataset (i.e. M_{base} trained on Clotho and M_{new} evaluated on AudioCaps, and vice-versa). Since in our method we do not consider the case of learning new words, we keep only the common words from the dataset used for evaluation. For example, in the case of training on Clotho and evaluating on AudioCaps, we keep from AudioCaps only the words that exist in Clotho. The amount of words that we remove from AudioCaps is 1715.

3.2. M_{base} model

As M_{base} we use the WT AAC model, presented in [22]. WT consists of four learnable processes, three used for audio encoding and one for decoding the learned audio information to captions. WT takes as an input a sequence of audio features, e.g. $\mathbf{X}'$ or $\mathbf{X}$, and generates a sequence of words, e.g. $\mathbf{Y}'$ or $\mathbf{Y}$. Input audio features are processed in parallel by two different learnable processes, one for learning temporal patterns, $E_{\text{temp}}(\cdot)$, and one for learning time-frequency patterns, $E_{\text{tf}}(\cdot)$. E_{temp} consists of 1D convolutional neural networks (CNNs), set-up after the WaveNet model [25] and using gated and dilated convolutions. E_{tf} is based on 2D depth-wise separable CNNs, capable to learn time-frequency information and proven to give state-of-the-art results in sound event detection [26]. Both E_{temp} and E_{tf} do not alter the temporal resolution of their input and their output is concatenated and given as an input to a third learnable process, $E_{\text{merge}}(\cdot)$. E_{merge} learns to intelligently merge the information from E_{temp} and E_{tf}, producing as an output an encoded sequence of the input audio, containing both temporal and time-frequency information.

The output of E_{merge} is given as an input to a decoder, $D(\cdot)$ that is based on the Transformer model [27], using three stacked multi-head attention blocks. Each attention block takes as an input a sequence of tokens/words and uses two different multi-head attention processes. The first is a masked self-attention, for each token/word attending only to its previous ones in the input sequence. The second multi-head attention is a cross-modal attention, attending to the output of E_{merge} given the output of the first, self-attention process. The first multi-head attention block D takes as an input its outputs shifted right and applies a positional encoding. The output of the last multi-head attention block is given as an input to a classifier, which shares its weights through time and predicts the most probable word in each time-step of the output caption. WT is illustrated in Figure 2, after [22].

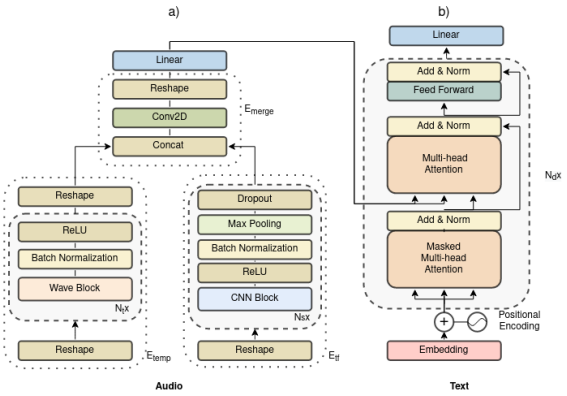


Figure 2: WT architecture, where a) is the encoder and b) the decoder, after [22].

3.3. Training, hyper-parameters, and evaluation

We compare the performance of our proposed method against the following baseline scenarios: i) WT pre-trained on Clotho and evaluated on Clotho and AudioCaps, ii) WT pre-trained on AudioCaps and evaluated on Clotho and AudioCaps, and iii) WT pre-trained on Clotho, fine-tuned on AudioCaps, and evaluated on Clotho and AudioCaps. We term the above cases as $\text{WT}_{\text{cl-au}}$, $\text{WT}_{\text{au-cl}}$, and $\text{WT}_{\text{cl-ft}}$, respectively. For pre-training M_{base}, we use the training split of the corresponding dataset, employing the early stopping policy by using the corresponding validation split and the associated SPIDEr score. For both datasets we use 10 consecutive epochs for early stopping, detecting not improving SPIDEr score. As an optimizer we use Adam [28] with the proposed values for the hyper-parameters. Additionally, we use a temperature hyper-parameter at the softmax non-linearity of the classifier of M_{new}, as this has been found to improve the performance [15]. We use the value of 2 for this hyper-parameter.

Using the above protocol, we evaluate the performance of our method using $\lambda = 0.70, 0.75, \ldots, 0.95, 1.0$ and $B = 4, 8, 12$. We use the pre-trained WT on Clotho, and we simulate $\mathcal{S}$ as mini-batches of size B from AudioCaps, as described by Eq. 3. We assess the performance of the M_{new} at the 50th, 75th, and 150th update, and after using only once all data from AudioCaps, using SPIDEr score [29]. SPIDEr [29] is the weighted average of CIDEr and SPICE metrics. CIDEr [30] employs weighted cosine similarity of n-grams, based on the term-frequency inverse-document-frequency (TFIDF), effectively quantifying the difference of the predicted and ground truth captions on using the same words to convey information. On the other hand, SPICE [31] analyzes the described scene and quantifies the differences of the predicted and ground truth caption in describing the same objects, attributes, and their relation-

Table 1: SPIDEr score of the baseline scenarios

Baseline scenario	SPIDEr $\mathbb{D}_{\text{ori}}$	SPIDEr $\mathbb{D}_{\text{new}}$
$\text{WT}_{\text{cl-au}}$	0.182	0.108
$\text{WT}_{\text{au-cl}}$	0.318	0.102
$\text{WT}_{\text{cl-ft}}$	0.065	0.247

ships.

4. RESULTS

In Table 1 are the results of M_{base}, regarding the three different baseline scenarios. In Table 2 are the obtained results of our method, for various values of B and λ, focusing on the SPIDEr score for $\mathbb{D}_{\text{ori}}$ and $\mathbb{D}_{\text{new}}$. As can be seen from Table 1 and from the cases of $\text{WT}_{\text{cl-au}}$ and $\text{WT}_{\text{au-cl}}$, the AAC method performs better on the $\mathbb{D}_{\text{ori}}$ than $\mathbb{D}_{\text{new}}$. This clearly shows that the model cannot perform equally well on the two different datasets, just by pre-training on one of them. Focusing on the $\text{WT}_{\text{cl-ft}}$, can be seen that the AAC method can perform good on the second dataset, i.e. $\mathbb{D}_{\text{new}}$, but the performance of the method on $\mathbb{D}_{\text{ori}}$ degrades considerably. This strengthens the need for our method, which aims at alleviating the degradation of performance on the $\mathbb{D}_{\text{ori}}$.

As can be seen from Table 2, it seems that the value of B has an observable impact on the performance on $\mathbb{D}_{\text{ori}}$. That is, lower values of B seem to not benefit the performance on $\mathbb{D}_{\text{ori}}$ for any value of λ. Specifically, for values of $B = 4$, the SPIDEr score on $\mathbb{D}_{\text{ori}}$ is lower than the SPIDEr score for $\mathbb{D}_{\text{ori}}$ and for $B > 4$, for any value of λ. The same stands mostly true for $B = 8$ and $B > 8$, with the exception where $\lambda = 0.7$. The above observation for B suggests that the batch size for sampling the stream of data $\mathcal{S}$ can also act as a regularizer for the not-forgetting of information from the $\mathbb{D}_{\text{ori}}$. Regarding the impact of λ, one can directly see the effect of the $1-\lambda$ and λ factors in Eq. (6), having $1-\lambda$ for scaling the effect of $\mathcal{L}_{\text{new}}$ and λ for scaling the effect of $\mathcal{L}_{\text{reg}}$. Specifically, for $\lambda = 1$ the SPIDEr score for $\mathbb{D}_{\text{new}}$ is lower than the SPIDEr score for $\mathbb{D}_{\text{ori}}$. This trend is in accordance with the observations from Table 1, and is an expected trend since the loss from $\mathbb{D}_{\text{new}}$ is turned to 0 for $\lambda = 1$. Given the observations for B from the same Table 2, it is indicated that using just the loss $\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}})$ for updating θ_{new} can enhance, up to an extent, the performance of the M_{new} on the new data from $\mathcal{S}$. Similarly, for values of $\lambda < 1.00$ the performance of M_{new} on $\mathbb{D}_{\text{new}}$ increases for all values of B. Additionally, the value of λ and the SPIDEr score on $\mathbb{D}_{\text{new}}$ have a reverse analogous relationship.

In terms of better performing combination of λ and B, we see two trends. There is the combination of $B = 4$ and $\lambda = 0.85$, which yields the best performance on $\mathbb{D}_{\text{new}}$ of SPIDEr$= 0.239$. Additionally, there is the combination of $B = 12$ and $\lambda = 0.80$, which seems to act as the best regularizer for the performance on $\mathbb{D}_{\text{ori}}$, with SPIDEr$= 0.186$. These results are in accordance with the previous observations for B and λ, indicating some kind of trade-off for the values of B and λ. Finally, comparing Tables 1 and 2, one can see the benefit of our method, giving a good balance between the top performance on $\mathbb{D}_{\text{new}}$ and not deteriorating the performance on $\mathbb{D}_{\text{ori}}$.

Table 2: Results of continual learning using Learning without Forgetting for AAC, for various B and λ. With bold are indicated the best SPIDEr scores for each dataset.

batch size B	λ	SPIDEr $\mathbb{D}_{\text{ori}}$	SPIDEr $\mathbb{D}_{\text{new}}$
4	0.70	0.098	**0.239**
	0.75	0.102	0.215
	0.80	0.093	0.214
	0.85	0.115	0.230
	0.90	0.133	0.215
	0.95	0.155	0.192
	1.00	0.163	0.119
8	0.70	0.113	0.210
	0.75	0.119	0.223
	0.80	0.132	0.220
	0.85	0.133	0.190
	0.90	0.156	0.187
	0.95	0.178	0.157
	1.00	0.165	0.114
12	0.70	0.109	0.211
	0.75	0.160	0.197
	0.80	**0.186**	0.157
	0.85	0.171	0.179
	0.90	0.182	0.153
	0.95	0.185	0.145
	1.00	0.176	0.115

5. CONCLUSIONS

In the paper we presented a first study of continual learning for AAC. Our method is based on the learning without forgetting method, which focuses on continuously updating the knowledge of a pre-trained AAC method on new AAC data, without degrading the performance of the AAC method on the originally used dataset during pre-training. For that reason, we employed a freely available and pre-trained AAC method and two freely available AAC datasets. We use the adopted AAC method which is pre-trained on one of the employed AAC datasets, and we use the other AAC dataset as a continuous stream of AAC data. We update the knowledge of the employed AAC method given the stream of AAC data. We compare our method against three baselines, two for training on one of the AAC datasets and evaluating on the other, and a third of training on one of the AAC datasets and fine-tuning the trained method to the other. Our results show that our method manages to not let the performance of the AAC method to deteriorate on the original AAC dataset, while, in the same time, manages to distil information from the new data to the employed AAC method.

For future research, utilizing AAC datasets set in more distinct domains and training those in consecutive way to the model would provide more data on how effective these methods can be when used for AAC. Recent years continuous learning has been a hot issue and more methods have been introduced just during last few years, many of which might effective when utilized for AAC as well.

6. ACKNOWLEDGMENT

The authors wish to acknowledge CSC-IT Center for Science, Finland, for computational resources. K. Drossos has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, project MARVEL.

7. REFERENCES

[1] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2017, pp. 374–378.

[2] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A transformer-based audio captioning model with keyword estimation," in *INTERSPEECH 2020*, 2020.

[3] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[4] S. Lipping, K. Drossos, and T. Virtanen, "Crowdsourcing a dataset of audio captions," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2019)*, 2019.

[5] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre- and post-processings for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 190–194.

[6] E. Çakır, K. Drossos, and T. Virtanen, "Multi-task regularization based on infrequent classes for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 6–10.

[7] X. Xu, H. Dinkel, M. Wu, and K. Yu, "A crnn-gru based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 225–229.

[8] K. Chen, Y. Wu, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on transformer and pre-trained cnn," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 21–25.

[9] K. Nguyen, K. Drossos, and T. Virtanen, "Temporal subsampling of audio feature sequences for automated audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 110–114.

[10] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *NAACL-HLT*, 2019.

[11] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[12] Z. Chen, B. Liu, R. Brachman, P. Stone, and F. Rossi, *Lifelong Machine Learning*, 2nd ed. Morgan & Claypool Publishers, 2018.

[13] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[14] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," 2017.

[15] Z. Li and D. Hoiem, "Learning without forgetting," 2017.

[16] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," 2018.

[17] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, pp. 6467–6476, 2017.

[18] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong, "Few-shot class-incremental learning," 2020.

[19] Y. Wang, N. J. Bryan, M. Cartwright, J. Pablo Bello, and J. Salamon, "Few-shot continual learning for audio classification," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 321–325.

[20] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," 2017.

[21] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2021.

[22] A. Tran, K. Drossos, and T. Virtanen, "Wavetransformer: An architecture for audio captioning based on learning temporal and time-frequency information," in *29th European Signal Processing Conference (EUSIPCO)*, Aug. 2021.

[23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[24] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[25] A. den Oord et al., "Wavenet: A generative model for raw audio," in *9th International Speech Communication Association (ISCA) Speech Synthesis Workshop*, 2016.

[26] K. Drossos, S. I. Mimilakis, S. Gharib, Y. Li, and T. Virtanen, "Sound event detection with depthwise separable and dilated convolutions," in *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020.

[27] A. Vaswani, L. Jones, N. Shazeer, N. Parmar, J. Uszkoreit, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representation (ICLR)*, 2014.

[29] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[30] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "CIDEr: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015.

[31] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *European Conference on Computer Vision*, 2016.

FEW-SHOT BIOACOUSTIC EVENT DETECTION: A NEW TASK AT THE DCASE 2021 CHALLENGE

Veronica Morfi[1], *Inês Nolasco*[1], *Vincent Lostanlen*[2], *Shubhr Singh*[1], *Ariana Strandburg-Peshkin*[3,4], *Lisa Gill*[5], *Hanna Pamuła*[6], *David Benvent*[7], *Dan Stowell*[8]

[1] Centre for Digital Music (C4DM), Queen Mary University of London, London, UK
[2] CNRS, Laboratoire des sciences du numérique de Nantes (LS2N), Nantes, France
[3] Dept. of Biology & Centre for the Advanced Study of Collective Behaviour, University of Konstanz, Germany
[4] Dept. for the Ecology of Animal Societies, Max Planck Institute of Animal Behavior, Germany
[5] BIOTOPIA Naturkundemuseum Bayern, Munich, Germany
[6] AGH University of Science and Technology, Kraków, Poland
[7] Cornell Lab of Ornithology, Cornell University, Ithaca, NY, US
[8] Tilburg University, Tilburg, The Netherlands; Naturalis Biodiversity Centre, Leiden, The Netherlands

ABSTRACT

Few-shot bioacoustic event detection is a novel area of research that emerged from a need in monitoring biodiversity and animal behaviour: to annotate long recordings, that experts usually can only provide very few annotations for due to the task being specialist and labour-intensive. This paper presents an overview of the first evaluation of few-shot bioacoustic sound event detection, organised as a task of the DCASE 2021 Challenge. A set of datasets consisting of mammal and bird multi-species recordings in the wild, along with class-specific temporal annotations, was compiled for the challenge, for the purpose of training learning-based approaches and for evaluation of the submissions in a few-shot labelled dataset. This paper describes the task in detail, the datasets that were used for both development and evaluation of the submitted systems, along with how system performance was ranked and the characteristics of the best-performing submissions. Some common strategies that the participating teams used are discussed, including input features, model architectures, transferring of prior knowledge, use of public datasets and data augmentation. Ranking for the challenge was based on overall performance of the evaluation set, however in this paper we also present results on each of the subsets of the evaluation set. This new analysis reveals submissions that performed better on specific subsets and gives an insight as to characteristics of the subsets that can influence performance.

Index Terms— Few-shot learning, bioacoustics, sound event detection, DCASE challenge

1. INTRODUCTION

The task of bioacoustic event detection refers to the retrieval of animal vocalizations in terms of onset and offset times. Thus, it shares a common methodology with other sound event detection (SED) contexts, such as offices [1], homes [2], city streets [3], and high-security spaces [4]. Yet, the application domain of bioacoustics is particularly challenging for SED, in part because of the high diversity of possible recording conditions and of vocalisation types [5]. For this reason, the field of machine learning for bioacoustics remains divided into many subfields: birds [6], land mammals, marine mammals [7], and so forth.

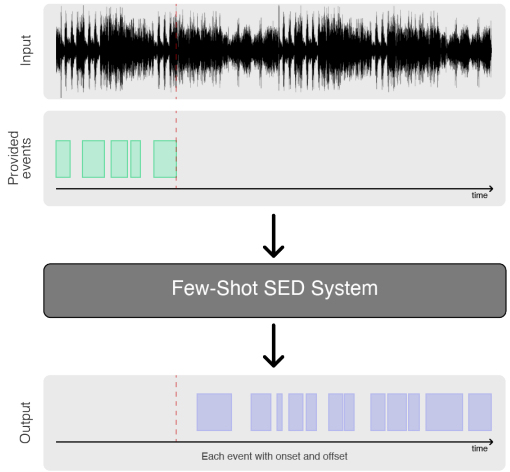


Figure 1: Overview of the proposed few-shot bioacoustic event detection task at the DCASE 2021 challenge. Green and purple rectangles represent labeled and predicted events, respectively.

The past decade witnessed the surge of deep convolutional networks (CNNs) in the time–frequency domain, which have the potential to outperform feature engineering. However, a supervised CNN for SED requires a predefined taxonomy of acoustic events as well as hundreds of annotated examples per class. Yet, collecting a large training set of animal vocalizations is not always feasible in practice, because species are unequally abundant [8]; audio annotation is costly and time-consuming [9]; and, more fundamentally, the taxonomy may vary depending on the use case [10].

We address this problem by introducing *few-shot bioacoustic event detection* as a new task to the DCASE 2021 challenge. In contrast to traditional deep learning approaches that use a large amount of data to train models, the key idea behind few-shot learning is to build accurate models with less training data [11]. More specifically, few-shot learning is usually studied using N-way-k-shot classification, where N denotes the number of classes and k the number of known examples for each class. Figure 1 illustrates the function-

ing of the system in deployment. After being trained on the first $k = 5$ occurrences of an event of interest, the system detects all the remaining occurrences of the same event in the rest of the recording.

Diverse approaches have been used to address the few-shot learning problem for classification, with no consensus on the best. Some use prior knowledge about similarity between sounds by computing embeddings (learnt representation spaces) while training and discriminate between unseen classes [11], while others exploit prior knowledge about the structure of the data by using augmentation to synthesize new data [12]. Finally, some approaches can learn models with parameters that can be fine-tuned to smaller datasets [13]. All of the above approaches deal with classification tasks in a few-shot learning setting and there is still much to be learnt in the field of few-shot SED; especially in concern to bioacoustic events.

While typical SED models must be retrained from scratch for each new use case, this few-shot formulation aims at learning generic representations of bioacoustic sounds. We encourage the community to develop an open-set SED system which bioacoustics practitioners will use on their own data after a modest amount of annotation, i.e., identifying the first k examples for each sound type.

2. DATASETS

A *development dataset* was provided for the task when the challenge was launched, consisting of predefined training and validation sets to be used for system development.[1] The development set consists of datasets from multiple sources with audio recordings and associated reference annotations in a task-specific format. More specifically, for the training set multi-class temporal annotations were provided for each recording as: positive (POS), negative (NEG) and unknown (UNK), while for the validation set single-class temporal annotations (POS/UNK) were provided for each recording.

A separate *evaluation set* was kept for evaluating the performance of the systems.[2] It consists of datasets from multiple sources. During the task five event annotations were provided for each of the recordings for the class of interest. The developed systems had to use those five annotated events and then learn to detect the same type of events throughout the rest of the recording.

Table 1 presents an overview of all the datasets in the development and evaluation sets, with information about the microphones used during recording, number of audio files, total time duration of the set, number of labels and number of annotated events.

BirdVox-DCASE-10h (BV): The BirdVox-DCASE-10h (BV) contains five audio files from four different autonomous recording units, each lasting two hours. These autonomous recording units are all located in Tompkins County, NY, US. They follow the same hardware specification: the Recording and Observing Bird Identification Node (ROBIN) developed by the Cornell Lab of Ornithology [14]. All recordings were acquired in 2015, during the fall migration season. An expert ornithologist, Andrew Farnsworth, has annotated flight calls from four families of passerines, namely: American sparrows, cardinals, thrushes, and New World warblers. The annotator found 2,662 flight calls from 11 different species in total. These flight calls have a duration in the range 50–150 milliseconds and a fundamental frequency in the range 2–10 kHz.

Hyenas (HT, HV): Spotted hyenas are a highly social species that live in "fission-fusion" groups where group members range alone or in smaller subgroups that split and merge over time, using a variety of types of vocalizations to coordinate with one another. Spotted hyena vocalization data were recorded on custom-developed audio tags designed by Mark Johnson and integrated into combined GPS/acoustic collars (Followit Sweden AB) by Frants Jensen and Mark Johnson. Collars were deployed on female hyenas of the Talek West hyena clan at the MSU-Mara Hyena Project (directed by Kay Holekamp) in the Masai Mara, Kenya as part of a multi-species study on communication and collective behavior. Recordings used as part of this task contain a variety of different vocalisations which were identified and classified into types based on the established hyena vocal repertoire [15]. The HT subset of the hyena recordings and their accompanying annotations were used as part of the development set, while the HV subset of recordings and their annotations were used as part of the validation. There is no overlap between the vocalisations annotated in the two sets. Field work was carried out by Kay Holekamp, Andrew Gersick, Frants Jensen, Ariana Strandburg-Peshkin, and Benson Pion; labeling was done by Kenna Lehmann and colleagues.

Meerkats (MT, ME): Meerkats are a highly social mongoose species that live in stable social groups and use a variety of distinct vocalisations to communicate and coordinate with one another. The meerkat vocal repertoire has been well characterized based on previous research, allowing calls to be reliably classified by human labellers [16, 17]. Recordings used in this task were acquired at the Kalahari Meerkat Project (Kuruman River Reserve, South Africa; directed by Marta Manser and Tim Clutton-Brock), as part of a multi-species study on communication and collective behavior. Recordings of the development set (MT) were recorded on small audio devices (TS Market, Edic Mini Tiny+ A77, 8 kHz) integrated into combined GPS/audio collars which were deployed on multiple members of meerkat groups to monitor their movements and vocalisations. Recordings of the evaluation set (ME) were recorded by an observer following a focal meerkat with a Sennheiser ME66 directional microphone (44.1 kHz) from a distance of less than 1 m. Recordings were carried out during daytime hours while meerkats were primarily foraging and include several different call types. Field work was carried out by Ariana Strandburg-Peshkin, Baptiste Averly, Vlad Demartsev, Gabriella Gall, Rebecca Schaefer and Marta Manser. Audio recordings were labeled by Baptiste Averly, Vlad Demartsev, Ariana Strandburg-Peshkin, and colleagues.

Jackdaws (JD): Jackdaws are corvid songbirds that usually breed, forage and sleep in large groups, but form a pair bond with the same partner for life. They produce thousands of vocalisations per day, but many aspects of their vocal behaviour remain unexplored due to the difficulty in recording and assigning vocalisations to specific individuals. In a multi-year field study (Max-Planck-Institute for Ornithology, Seewiesen, Germany), wild jackdaws were equipped with small backpacks containing miniature voice recorders (Edic Mini Tiny A31, TS-Market Ltd., Russia) to investigate the vocal behaviour of individuals interacting with their group and behaving freely in their natural environment. The JD dataset contains a 10-minute on-bird sound recording (22050 Hz) of one male jackdaw during the breeding season 2015. Field work was conducted by Lisa Gill, Magdalena Pelayo van Buuren and Magdalena Maier. Sound files were annotated by Lisa Gill, based on a previously established video-validation in a captive setting [18].

Polish Baltic Sea bird flight calls (PB): The PB dataset consists of six 30 minute recordings of bird flight calls recorded along the Polish Baltic Sea coast. The recordings are the excerpt from Hanna Pamuła's project, focused on the acoustic monitoring of birds migrating at night along the Polish Baltic Sea coast. Three autonomous recording units were used with the same hardware set-

[1] https://doi.org/10.5281/zenodo.4543504
[2] https://doi.org/10.5281/zenodo.4864755

	Dataset	mic type	# audio files	total duration	# labels (excl. UNK)	# events (excl. UNK)
Development Set: Training	BV	fixed	5	10 hours	11	2,662
	HT	mobile	3	3 hours	3	435
	MT	mobile	2	70 mins	4	1,234
	JD	mobile	1	10 mins	1	355
Development Set: Validation	HV	mobile	2	2 hours	2	50
	PB	fixed	6	3 hours	2	260
Evaluation Set	ME	handheld	2	20 mins	2	70
	ML	various	17	20 mins	17	1,035
	DC	fixed	13	105 mins	3	967

Table 1: Information on each dataset.

tings (Song Meters SM2, Wildlife Acoustics, Inc). They were deployed close to each other (<100m) - near the lake, on the dune, and on the forest clearing - to provide diverse acoustic background. The recordings were acquired during the 2016, 2017 and 2018 fall migration seasons. The passerines night flight calls were annotated by Hanna Pamuła. The PB dataset is part of the development set used for validation. In each recording only one bird species is the target class: song thrush, *Turdus philomelos* (3 recordings); blackbird, *Turdus merula* (3 recordings). Each recording contains 22–93 calls in the 8–400 milliseconds range. The usual fundamental frequency range for calls of the chosen species is 5–9 kHz.

Macaulay Library (ML): The Macaulay Library is a digital archive of images, videos, and sounds from animals.[3] As of 2021, it contains 175k audio recordings from 10k species of birds and 2k species of amphibians, fish, mammals and insects. These recordings are contributed by amateur and professional recordists around the world, and the catalogue is maintained by the Cornell Lab of Ornithology. For the DCASE 2021 challenge, one author (DB) curated 17 recordings from the Macaulay Library and annotated them in terms of animal vocalizations. Each recording contains calls from a different species: 14 terrestrial mammals (not including hyena or meerkat) and 3 birds (not including passeriformes). The average duration of each recording is of the order of one minute and the number of calls per minute varies in the range 10–150.

BIOTOPIA Dawn Chorus (DC): Many bird species produce vocalisations during the entire day, but their vocally most active period by far usually occurs around dawn. This natural phenomenon is called *dawn chorus*. The Dawn Chorus project is a worldwide citizen science and arts project bringing together amateurs and experts to experience and record the dawn chorus at their doorstep.The DC dataset used as part of the evaluation set stems from dawn chorus recordings, made using Zoom H2 recorders at 44100 Hz, at three different locations in Southern Germany (Haspelmoor, Munich's Nymphenburg Schlosspark, and Nantesbuch), by Moritz Hertel and Rudi Schleich. The vocalisations of three target species were annotated by LG (Common cuckoo, *Cuculus canorus*: 6 files, ca. 9 minutes, 543 labels; European robin, *Erithacus rubecula*: 3 files, ca. 43 min, 381 labels; Eurasian wren, *Troglodytes troglodytes*: 3 files, ca. 50 min, 268 labels).

3. BASELINE METHODS

We propose two systems as baselines to measure submitted methods performance with. One is an approach commonly used in bioacoustics based on spectrogram cross-correlation and the other is a deep learning approach based on prototypical networks [11].

[3]Official website: `https://www.macaulaylibrary.org/`

3.1. Template Matching

Our first baseline is a spectrogram cross-correlation method, based on scikit-image's `match_template` function that uses fast, normalized cross-correlation to find instances of a template in an image, returning values ranged between -1.0 and 1.0, with higher values corresponding to higher correlation. Our few-shot template matching method computes cross-correlation across the time axis between each of the events (shots) provided for a file and the rest of the recording. A different detection threshold is set for each audio file based on the max value of the cross-correlation results between the shots provided. Peak picking is performed on the results of the template matching algorithm, with any peak above the threshold corresponding to the center of a detected event in that recording. Borders of the predicted event are computed based on the length of the shot it was correlated with. Predictions from all shots of a recording are collapsed into a single binary prediction vector which will produce the final events predicted for the class of interest.

3.2. Prototypical Network

Our second baseline is based on prototypical networks [11]. The goal of prototypical networks and episodic training is to learn a classifier which can adapt quickly to new classes with only a few examples. Each episode of the training is configured as a N*-way-*k*-shot classification*, where N denotes the number of classes and k the number of known samples per class. A mini batch is sampled from the training set and split into a support set consisting of k labelled samples, $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ where $x_i \in \mathbb{R}^D$ and $y_i \in \{1, 2, \ldots, N\}$ is the corresponding label, with the remaining samples comprising the query set Q. Prototypical networks compute an M-dimensional class prototype $c_n \in \mathbb{R}^M$, through an embedding function $f_\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$ with learnable parameters ϕ. In our baseline $D = 128$ and $M = 64$.

We compute a prototype for each class as the mean of the embedded support points belonging to it. Then, for each sample x_q from the query set, a distance function is used to calculate the Euclidean distance of x_q from each prototype, following which a softmax function over the distances produces a distribution over the classes. Learning proceeds by minimizing the negative log probability $J(\phi) = -\log p_\phi(y_q = n|x_q)$ over the true class k via stochastic gradient descent.

During evaluation, we adopt a binary classification strategy inspired by [26]. The first 5 positive (POS) annotations are used for calculation of positive class prototype and the rest of the audio file is treated as the negative class, based on the assumption that the positive class is relatively sparse in the recording. We randomly sample from the negative class to calculate the negative prototype. Each

Rank	Team name	Evaluation set: F-score % (97.5% confidence interval)	Validation set: F-score %	DC F-score %	ME F-score %	ML F-score %
1	Zou_PKU [19]	**38.4** (36.2 - 40.6)	55.3	20.6	68.0	67.3
2	Tang_SHNU [20]	38.3 (36.1 - 40.5)	51.4	25.6	61.5	43.3
3	Anderson_TCD [21]	35.0 (33.1 - 37.0)	26.2	19.9	56.6	56.8
4	Baseline_TempMatch	34.8 (32.6 - 37.1)	2.0	**32.2**	47.1	29.5
5	Cheng_BIT [22]	23.8 (21.9 - 25.7)	46.3	10.6	53.5	**78.8**
6	Baseline_PROTO	20.1 (18.2 - 21.9)	41.5	8.5	**72.7**	55.7
7	Zhang_uestc [23]	16.8 (15.5 - 18.2)	54.4	8.1	45.1	29.9
8	Johannsmeier_OVGU [24]	15.2 (13.7 - 16.7)	**58.6**	6.5	64.3	35.8
9	Bielecki_SMSNG [25]	8.4 (7.1 - 9.7)	51.8	3.1	56.3	51.4

Table 2: F-score results per team on evaluation and validation sets.

query sample is assigned a probability based on the distance from the positive and negative prototype. Onset and offset predictions are made based on thresholding probabilities at a value of 0.5 across the query set. The prediction process for each file is repeated 5 times, with the negative prototype created by random sampling each time. The final prediction probability for each query frame is the average of predictions across all iterations. Finally, post-processing is applied to the outputs in order to remove possible false positives. For each audio file, predicted events with shorter duration than 60% of the duration of the shortest shot provided for that file are removed.

4. EVALUATION AND RESULTS

For the evaluation of this task we employ an event-based F-measure with macro-averaged metric. The main challenge is related to the detection of a match between ground truth events and predicted events. Traditional approaches use onset detection based metrics and fixed-size evaluation windows. Given the great variation between datasets and characteristics of the events we want to detect in this task, these approaches are not suitable. Instead, we use the Intersection over Union (IoU), with 30% minimum overlap to produce a list of possible matches of the predictions. For each ground truth event, a single best match is selected by applying the Hopcroft-Karp-Karzanov algorithm for bipartite graph matching.

In a SED task we can define True Positives (TP) as predicted events that match ground truth events, False Positives (FP) as predicted events that do not match any ground truth events, and False Negatives (FN) as ground truth events that are not predicted. In this task, ground truth events consist of POS events of the class and UNK events that have some uncertainty associated to the assigned class. The procedure we employ is:

1. Apply IoU and bipartite graph matching between predicted events and ground truth POS events only, resulting in TP.
2. Apply IoU and bipartite graph matching between remaining predicted events, that did not match with any POS event, and ground truth UNK events only.
3. Compute FP as the number of predicted events that were not matched to either POS or UNK events.
4. Compute FN as the number of POS ground truth events that were not matched by any predicted event.

This is applied to each dataset in the evaluation set where we compute the F-score metric. The reported results are the harmonic mean over all the datasets, which is appropriate for combining percentage results, and ensures that a system should perform well across all datasets to achieve a strong score.

4.1. Results

DCASE 2021 task 5 had 7 teams participating with a total of 24 submitted systems. F-score results per team are presented in Table 2. All submitted systems adopted prototypical networks. Data augmentation was applied by the majority of the teams with SpecAugment[27] being the most popular choice. All systems rely on some sort of post-processing mechanism designed to removing superfluous predictions and many teams report important improvements in results due to it. Another popular choice was using Per-channel Energy Normalization (PCEN) [28] as acoustic features.

The best ranked system [19] improved over the baseline prototypical approach by applying a transductive inference method, where supplemental information is used to convey more representative prototypes of each category. A mutual learning framework designed to make the feature extraction network more task dependent is also adopted. The system ranked in second place [20] also improved over the prototypical baseline by using additional data from Audioset to train a ResNet for the feature extraction part. They have also adopted embedding propagation (EP) [29], with the objective of smoothing the decision boundaries as a way of increasing the generalisation capabilities of the few-shot system. The third ranking system [21], follows the same approach as the prototypical network baseline, with the main differences being the use of data augmentation and reducing the size of the network. Interestingly, although the results in the validation set are not on par with the other systems, this system outperforms most systems in the evaluation set.

Also of note, the work in [22] uses i-vectors as input features; both submissions in [23] and [24], explicitly create a negative class to model background noise and construct a negative prototype; and in [25], the team opted for combining the prototypical loss, with knowledge distillation and attention transfer loss.

An important observation from Table 2 is the drop in F-score from the validation to the evaluation set for the majority of the systems. This suggests that the systems are generally dataset sensitive. To highlight this aspect further, we report the F-score results per dataset in the evaluation set. Most systems tend to have a decrease in performance on the DC set, comprised of dawn chorus recordings, while perform better on ME and ML that include mainly mammal vocalisations. This leads to the conclusion that very complex environments, such as dawn chorus, need further techniques to be employed for robust SED. Our template matching baseline improved performance from the validation set to the evaluation set. This is mainly due to template matching not being trained over specific recordings, treating each audio file as a unique task without any knowledge about the rest of the set with performance only depends on the templates (shots) used for cross-correlation.

5. REFERENCES

[1] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[2] N. Turpault, R. Serizel, J. Salamon, and A. P. Shah, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," 2019.

[3] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *2010 18th European Signal Processing Conference*. IEEE, 2010, pp. 1267–1271.

[4] A. Mesaros, A. Diment, B. Elizalde, T. Heittola, E. Vincent, B. Raj, and T. Virtanen, "Sound event detection in the dcase 2017 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 6, pp. 992–1006, 2019.

[5] D. Stowell, "Computational bioacoustic scene analysis," in *Computational analysis of sound scenes and events*. Springer, 2018, pp. 303–333.

[6] S. Kahl, M. Clapp, W. Hopping, H. Goëau, H. Glotin, R. Planqué, W.-P. Vellinga, and A. Joly, "Overview of BirdCLEF 2020: Bird sound recognition in complex acoustic environments," in *CLEF 2020*, 2020.

[7] F. Frazao, B. Padovese, and O. S. Kirsebom, "Workshop report: Detection and classification in marine bioacoustics with deep learning," 2020.

[8] W.-P. Vellinga and R. Planqué, "The xeno-canto collection and its relation to sound recognition and classification," in *CLEF (Working Notes)*, 2015.

[9] A. E. Méndez Méndez, M. Cartwright, and J. P. Bello, "Machine–crowd–expert model for increasing user engagement and annotation quality," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–6.

[10] J. Cramer, V. Lostanlen, A. Farnsworth, J. Salamon, and J. P. Bello, "Chirping up the right tree: Incorporating biological taxonomies into deep bioacoustic classifiers," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 901–905.

[11] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017.

[12] Y.-X. Wang, R. B. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7278–7286, 2018.

[13] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *ArXiv*, vol. abs/1803.02999, 2018.

[14] V. Lostanlen, J. Salamon, A. Farnsworth, S. Kelling, and J. P. Bello, "Birdvox-full-night: A dataset and benchmark for avian flight call detection," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 266–270.

[15] K. D. S. Lehmann, "Communication and cooperation in silico and nature," Ph.D. dissertation, Michigan State University, 2020.

[16] M. B. Manser, "The evolution of auditory communication in suricates, suricata suricatta," Ph.D. dissertation, University of Cambridge, 1998.

[17] M. B. Manser, D. A. Jansen, B. Graw, L. I. Hollén, C. A. Bousquet, R. D. Furrer, and A. le Roux, "Chapter six - vocal complexity in meerkats and other mongoose species," ser. Advances in the Study of Behavior, M. Naguib, L. Barrett, H. J. Brockmann, S. Healy, J. C. Mitani, T. J. Roper, and L. W. Simmons, Eds. Academic Press, 2014, vol. 46, pp. 281–310. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128002865000067

[18] D. Stowell, E. Benetos, and L. F. Gill, "On-bird sound recordings: automatic acoustic recognition of activities and contexts," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1193–1206, 2017.

[19] D. Yang, H. Wang, Z. Ye, and Y. Zou, "Few-shot bioacoustic event detection = a good transductive inference is all you need," DCASE2021 Challenge, Tech. Rep., June 2021.

[20] T. Tang, Y. Liang, and Y. Long, "Two improved architectures based on prototype network for few-shot bioacoustic event detection," DCASE2021 Challenge, Tech. Rep., June 2021.

[21] M. Anderson and N. Harte, "Bioacoustic event detection with prototypical networks and data augmentation," DCASE2021 Challenge, Tech. Rep., June 2021.

[22] H. Cheng, C. Hu, and M. Liu, "Prototypical network for bioacoustic event detection via i-vectors," DCASE2021 Challenge, Tech. Rep., June 2021.

[23] Y. Zhang, J. Wang, D. Zhang, and F. Deng, "Few-shot bioacoustic event detection using prototypical network with background classs," DCASE2021 Challenge, Tech. Rep., June 2021.

[24] J. Johannsmeier and S. Stober, "Few-shot bioacoustic event detection via segmentation using prototypical networks," DCASE2021 Challenge, Tech. Rep., June 2021.

[25] R. Bielecki, "Few-shot bioacoustic event detection with prototypical networks , knowledge distillation and attention transfer loss," DCASE2021 Challenge, Tech. Rep., June 2021.

[26] Y. Wang, J. Salamon, M. Cartwright, N. J. Bryan, and J. P. Bello, "Few-shot drum transcription in polyphonic music," *CoRR*, vol. abs/2008.02791, 2020. [Online]. Available: https://arxiv.org/abs/2008.02791

[27] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech 2019*, Sep 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680

[28] V. Lostanlen, J. Salamon, M. Cartwright, B. McFee, A. Farnsworth, S. Kelling, and J. P. Bello, "Per-channel energy normalization: Why and how," *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 39–43, 2019.

[29] P. Rodríguez, I. Laradji, A. Drouin, and A. Lacoste, "Embedding propagation: Smoother manifold for few-shot classification," in *European Conference on Computer Vision*. Springer, 2020, pp. 121–138.

ACTIVE LEARNING FOR SOUND EVENT CLASSIFICATION USING MONTE-CARLO DROPOUT AND PANN EMBEDDINGS

Stepan Shishkin[1]*, *Danilo Hollosi*[1], *Simon Doclo*[1,2], *Stefan Goetze*[3]

[1] Fraunhofer Institute for Digital Media Technology IDMT, Division Hearing, Speech and Audio Technology, Oldenburg, Germany,{stepan.shishkin, danilo.hollosi}@idmt.fraunhofer.de
[2] University of Oldenburg, Dept. of Medical Physics and Acoustics and Cluster of Excellence Hearing4all, Oldenburg, Germany, simon.doclo@uni-oldenburg.de
[3] The University of Sheffield, Dept. of Computer Science, Speech and Hearing (SPandH), Sheffield, United Kingdom, s.goetze@sheffield.ac.uk

ABSTRACT

Labeling audio material to train classifiers comes with a large amount of human labor. In this paper, we propose an active learning method for sound event classification, where a human annotator is asked to manually label sound segments up to a certain labeling budget. The sound event classifier is incrementally re-trained on pseudo-labeled sound segments and manually labeled segments. The segments to be labeled during the active learning process are selected based on the model uncertainty of the classifier, which we propose to estimate using Monte Carlo dropout, a technique for Bayesian inference in neural networks. Evaluation results on the UrbanSound8K dataset show that the proposed active learning method, which uses pre-trained audio neural network (PANN) embeddings as input features, outperforms two baseline methods based on medoid clustering, especially for low labeling budgets.

Index Terms— sound event classification, active learning, Monte Carlo dropout, self-training, transfer learning

1. INTRODUCTION

Sound event classification, being an important part of machine audition [1], aims at differentiating between situations or events based on their acoustic properties [2–4]. Some of its applications include acoustic scene classification [5], environmental noise classification [6], traffic surveillance [7], monitoring of patient health [8], wildlife sound classification [9] and music genre classification [10]. To train a sound event classifier, a corpus of labeled recordings is required. While recording a sufficiently large audio corpus can be time-consuming by itself, the subsequent manual labeling of the recordings typically requires even more effort and is usually the bottleneck in the data preparation process.

In active learning (AL) [11, 12], a human annotator is queried to manually label unlabeled data during the training process. AL is usually formulated as a process that iterates between re-training the classifier upon receiving new labels from the annotator, and selecting unlabeled data to be manually labeled next. For a given labeling budget, i.e. the maximum number of labels a human annotator is asked to provide within the AL process, the aim is to maximize the accuracy of the classifier. Hence, algorithms are typically designed to maximize the informativeness of the received labels. In the context of sound event classification, AL has been applied to train support vector machine (SVM) classifiers [13, 14], a random forest [15], and a combination of an SVM and a nearest-neighbor classifier [16].

Rather than fitting a single or a handful of classifiers, one can instead model a Bayesian distribution over hypotheses, e.g., using neural networks [17–21]. In [17] it was shown that variational Bayesian inference can be performed by training a neural network in which a dropout layer precedes every weight layer. This technique, known as Monte Carlo (MC) dropout, allows to sample hypotheses from an approximate Bayesian posterior by means of sampling dropout masks. Although MC dropout has been successfully employed to improve informativeness estimates in AL [22, 23], to the best of our knowledge it has not yet been applied to sound event classification. Our proposed method, MC dropout active learning (DAL), combines AL, self-training by generating pseudo-labels for unlabeled sound segments, and transfer learning by using pre-trained audio neural network (PANN) embeddings [24] as input features. Evaluation results on the UrbanSound8k dataset [25] show that the proposed DAL method yields a larger classification accuracy than two baseline methods, especially for low labeling budgets.

In Section 2, we formalize the underlying active learning problem. Baseline AL methods based on medoid clustering are described in Section 3. Section 4 describes the proposed MC dropout AL method. In Section 5, the evaluation procedure and the experimental results are presented.

2. PROBLEM DEFINITION

Given is a labeling budget N, a set of sound event classes C, and a partially labeled set of sound segments, where each segment contains sound events from exactly one class c in C. The i^{th} segment is represented by its corresponding feature vector $\mathbf{x}_i$. We define the *unlabeled* set $S_U = \{\mathbf{x}_i\}$, containing feature vectors $\mathbf{x}_i$ of unlabeled segments, and the *(manually) labeled* set $S_L = \{(\mathbf{x}_i, l_i)\}$, containing tuples of feature vectors $\mathbf{x}_i$ and labels l_i of labeled segments. Each label corresponds to exactly one class in C. In the following, we use the term "segment" to refer to the feature vector corresponding to a segment.

The goal is to fit a classifier that predicts the class label $\hat{l}$ of any segment $\mathbf{x}$ as accurately as possible. To train the classifier, we have access to the sets S_U and S_L, and we are allowed to request labels for up to $N - |S_L|$ unlabeled segments, with $|S_L|$ the cardinality of S_L. The choice of the unlabeled segments that are labeled within the AL process may have a large impact on the resulting classifier's accuracy.

*This work was partially funded by the German Ministry of Science and Education (BMBF) in the project KI-MUSIK4.0 - Universal microelectronic-based sensor interface for industry 4.0.

3. BASELINE METHODS

In this section we briefly review the medoid active learning (MAL) method for sound event classification proposed in [14] and a modified version using PANN embeddings [24], referred to as MAL-PANN.

In MAL, a fully unlabeled set of segments is first split into small clusters using k-medoid clustering. The inter-segment distance metric used for clustering is based on segment-wide statistics of mel frequency cepstral coefficients (MFCCs) and their first- and second-order time derivatives. Specifically, for each MFCC and each time derivative, a normal distribution is fitted, and the distance between segments is computed based on the Kullback-Leibler divergence between the respective normal distributions. Starting from the largest cluster, medoids are then selected for labeling, where a medoid's label is propagated to other segments in the respective cluster. Once the number of labeled medoids matches the labeling budget N, an SVM classifier is fitted on both manually assigned as well as propagated labels. Acoustic features used for training the SVM are minimum, maximum, median, mean, variance, skewness, kurtosis of MFCCs as well as mean and variance of the first- and second-order time derivatives.

MAL-PANN is our modification of the MAL method, where we replace the MFCC-based features with the recently proposed PANN embeddings [24], i.e. the activations in the penultimate layer of the CNN-14 model that was trained on the AudioSet dataset [26]. Employing these pre-trained features instead of the original arbitrarily chosen features makes for a more fair benchmark to compare the DAL method (see Section 4) against. The inter-segment distance metric $s(\mathbf{x}_1,\mathbf{x}_2)$ in MAL-PANN is based on the cosine similarity between PANN embeddings $\mathbf{x}_1$ and $\mathbf{x}_2$, i.e.

$$s(\mathbf{x}_1,\mathbf{x}_2)=1-\frac{\mathbf{x}_1^T\mathbf{x}_2}{||\mathbf{x}_1||\cdot||\mathbf{x}_2||}, \tag{1}$$

where $(\cdot)^T$ denotes transpose, and $||\cdot||$ denotes the L^2-norm of a vector.

4. MONTE-CARLO DROPOUT ACTIVE LEARNING (DAL)

Instead of only fitting the classifier once the labeling budget is depleted (as in MAL), in the proposed DAL method the classifier is incrementally re-trained during the AL process. To enhance the training process, self-training is applied to generate *pseudo-labels* for unlabeled segments, which act as additional training targets for the classifier. Furthermore, the selection of segments to be manually labeled is based on a so-called *acquisition function*, which estimates the informativeness of labeling a segment. The acquisition function employed is based on model uncertainty, i.e. on the disagreement between individual hypotheses in a Bayesian posterior. To draw hypotheses from the posterior, and to measure the disagreement between their predictions, we propose to employ Monte Carlo dropout. To this end, the classifier is designed as a neural network that contains a dropout layer followed by a dense layer. Section 4.1 describes the architecture of the neural network classifier. In Section 4.2 the proposed iterative AL algorithm is presented, where the classifier is incrementally re-trained on each iteration.

4.1. Classifier

Figure 1 depicts the architecture of the neural network classifier, which maps a 2048-dimensional PANN embedding $\mathbf{x}$ of a sound segment to the respective class. The neural network consists of a dense layer preceded by a dropout layer with 50% dropout probability, and followed by a softmax layer. The dropout layer is kept in stochastic mode both during training and during inference.

A single forward pass through the network results in the class probability distribution $P(c|\mathbf{x},\mathbf{d})$ where $\mathbf{d}$ is the randomly sampled dropout mask. This output can be interpreted as the prediction of a hypothesis about the class distribution associated with the segment $\mathbf{x}$. The posterior distribution over classes $P(c|\mathbf{x})$ can be computed via a Monte Carlo estimate by sampling multiple dropout masks and averaging the individual outputs [17], i.e.

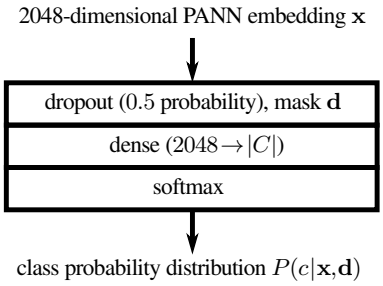


Figure 1: Neural network used in DAL for sound segment classification.

$$P(c|\mathbf{x})=\frac{1}{|D|}\sum_{\mathbf{d}\in D}P(c|\mathbf{x},\mathbf{d}), \tag{2}$$

where D denotes the set of sampled dropout masks. The number of sampled dropout masks $|D|$ is a parameter of DAL.

The classifier's predicted label for segment $\mathbf{x}$ corresponds to the class with the highest predicted probability, i.e.

$$\hat{l}(\mathbf{x})=\operatorname*{argmax}_{c\in C}P(c|\mathbf{x}). \tag{3}$$

4.2. Iterative active learning algorithm

In addition to the unlabeled set S_U and the (manually) labeled set S_L, DAL maintains a *set S_P of pseudo-labeled [27] sound segments*, which act as additional training targets for the classifier. The AL process starts with an initialization stage, and then iterates between stage I and stage II until the labeling budget N is depleted.

4.2.1. Initialization stage

DAL requires an initial set of labeled segments, on which the classifier is trained by minimizing the cross-entropy loss for a fixed number of gradient descent steps. The initial labeled set counts toward the labeling budget N.

4.2.2. Stage I: scanning S_U and generating S_P

For each unlabeled sound segment $\mathbf{x}\in S_U$, the *confidence* of the classifier is defined as as the highest class probability $P(\hat{l}(\mathbf{x})|\mathbf{x})$. If the confidence is larger than a certain threshold Θ, the tuple $(\mathbf{x},\hat{l}(\mathbf{x}))$ is copied into the pseudo-labeled set

$$S_P=\{(\mathbf{x},\hat{l}(\mathbf{x}))|\mathbf{x}\in S_U; P(\hat{l}(\mathbf{x})|\mathbf{x})>\Theta\}, \tag{4}$$

whereby the confidence threshold Θ is a parameter of DAL. Setting $\Theta=1$ corresponds to turning off pseudo-labeling, whereas $\Theta=0$ corresponds to assigning pseudo-labels to all unlabeled segments. It should be noted that S_P is generated anew in each iteration.

In addition, to estimate the informativeness of labeling a segment, for each unlabeled segment $\mathbf{x}\in S_U$ we compute the *acquisition function* value [22,28]. For that, each hypothesis sampled via MC dropout produces a single vote in favor of one class, resulting in the so-called vote distribution

$$\tilde{P}(c|\mathbf{x})=\frac{1}{|D|}\sum_{\mathbf{d}\in D}\delta_{c,\text{vote}(\mathbf{x},\mathbf{d})}, \tag{5}$$

with δ the Kronecker-delta and

$$\text{vote}(\mathbf{x},\mathbf{d}) = \underset{c\in C}{\text{argmax}} P(c|\mathbf{x},\mathbf{d}) \tag{6}$$

the class with the highest predicted probability when using the dropout mask $\mathbf{d}$. As acquisition function we use the vote entropy [29], i.e. the entropy of the vote distribution $\tilde{P}(c|\mathbf{x})$, i.e.

$$H_{\tilde{P}}(\mathbf{x}) = -\sum_{c\in C} \tilde{P}(c|\mathbf{x}) \cdot \log \tilde{P}(c|\mathbf{x}). \tag{7}$$

The acquisition function thus captures the model uncertainty, i.e. the degree of disagreement between predictions of the individual hypotheses. The unlabeled segment with the highest vote entropy $H_{\tilde{P}}$ is then presented to the annotator, removed from the unlabeled set S_U and added to the labeled set S_L along with the corresponding label. Each acquired label counts toward the labeling budget. It should be noted that in the first T_0 iterations no manual labels are requested, enabling the classifier to train on labeled and pseudo-labeled segments, without consuming the labeling budget.

4.2.3. Stage II: re-training the classifier

The classifier is re-trained on labeled segments in S_L and pseudo-labeled segments in S_P by minimizing the cross-entropy loss. Segments are sampled into minibatches such that a minibatch contains the same number B of segments for each class. It is well known that unconstrained training on pseudo-labeled data degrades model performance due to self-amplifying classification errors in the training dataset [30]. Hence, to reduce the impact of pseudo-labeled segments, for each class c we draw $B_{L,c}$ labeled and $B_{P,c}$ pseudo-labeled segments into a minibatch such that

$$B_{P,c} = \left\lfloor \alpha B \frac{|S_{P,c}|}{|S_{L,c}| + \alpha |S_{P,c}|} \right\rfloor, \tag{8}$$

$$B_{L,c} = B - B_{P,c}, \tag{9}$$

where $|S_{L,c}|$ and $|S_{P,c}|$ denote the number of labeled and pseudo-labeled segments belonging to class c, and α is a parameter of DAL. This effectively makes the chance of a pseudo-labeled segment to be drawn into the minibatch α^{-1} times smaller than the chance of a labeled segment. Setting $\alpha = 0$ prevents pseudo-labeled segments to be used for training, whereas for $\alpha = 1$ pseudo-labeled and labeled segments attain the same weight. Minibatch sampling and gradient descent are repeated a fixed number of times.

5. EVALUATION

In this section we evaluate the performance of the proposed DAL method and compare it with the baseline methods (MAL, MAL-PANN).

After presenting the used dataset and the performance metrics in Section 5.1, the default parameter values for DAL are discussed in Section 5.2. The experimental results are presented in Section 5.3.

5.1. Dataset and performance metrics

The performance of the considered AL methods is evaluated on the UrbanSound8K dataset [25], an environmental dataset containing 8732 short sound segments (up to 4 seconds). Each segment is weakly labeled with one of the following 10 classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music.

In the experiments, DAL is initialized with a labeled set S_L (see Section 4.2.1) which contains 3 randomly chosen segments for every class, i.e. 30 labeled segments in total. Manual labeling is simulated by revealing the ground truth label to an AL algorithm.

We assess the performance of an AL algorithm by means of the classification accuracy for different labeling budgets evaluated on the test split via 10-fold cross-validation. The accuracy is evaluated as the macro-averaged recall [31], which computes the percentage of correctly predicted ground-truth labels for each class, and averages these percentages over all classes. Depending on the computational cost of an experiment, we either conducted one or 10 experimental trials, i.e. repeated the experiment 10 times. For each experiment, 80% confidence intervals for the macro-averaged recall were computed using the bootstrap method. For the case of one experimental trial we treated each fold in the 10-fold cross-validation as an individual experiment when computing confidence intervals.

5.2. Default parameters

Table 1 summarizes default parameter values of the DAL method that were used in the experiments described in Section 5.3.

parameter	**value**
pseudo-labeling	
confidence threshold Θ in (4)	0.5
sampling weight α in (8) of pseudo-labeled segments	0.01
number T_0 of initial iterations without new acquisition	3
Monte Carlo dropout	
number of sampled dropout masks $\lvert D\rvert$ in (2) and (5)	128
optimization	
per-class minibatch size B in (8)	256
number of gradient descents per iteration	40 (1600 at initialization)
optimizer	Adam
learning rate	$1e{-}3$
weight decay	$1e{-}3$

Table 1: Parameter values for the DAL method.

5.3. Results

In Sections 5.3.1 and 5.3.2 we investigate the performance of the proposed DAL method while variating two important parameters: the confidence threshold Θ and the sampling weight α. It is worth noting that whenever one parameter was variated, the other was set to its default value (cf. Table 1). For the default values of all parameters as in Table 1, we then compare the performance of DAL with the baseline methods in Section 5.3.3.

5.3.1. DAL performance sensitivity to Θ

As discussed in Section 4.2, using pseudo-labels to train the classifier is an important aspect of DAL. Since the assignment of an unlabeled segment in the pseudo-labeled set S_P depends on the confidence threshold Θ in (4), it is important to understand the impact of this parameter on the overall performance.

Figure 2 depicts the performance of DAL for different values of the confidence threshold Θ for labeling budgets between 30 and 130. Studying and optimizing the performance for low labeling budgets is especially relevant for real-world applications. Results suggest that the best performance is achieved for a moderate value around $\Theta = 0.5$. As discussed in Section 4.2.2, setting $\Theta = 1$ corresponds to effectively turning off

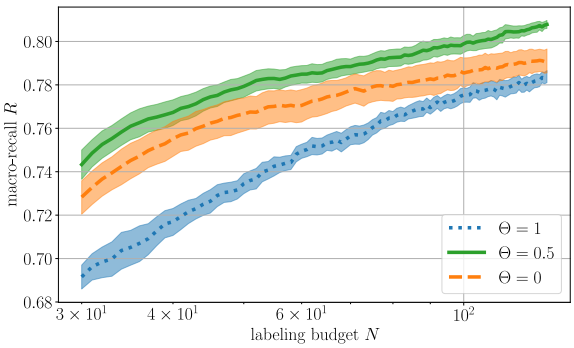


Figure 2: Macro-recall R over labeling budget N for different values of the confidence threshold Θ for assigning pseudo-labels in DAL. Confidence intervals are computed from 10 experimental trials.

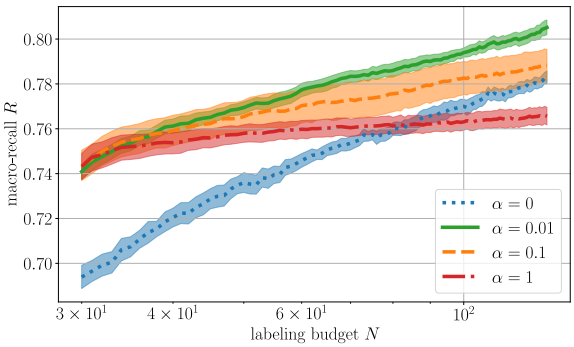


Figure 3: Macro-recall R over labeling budget N for different values of the sampling weight α of pseudo-labeled segments in DAL. Confidence intervals are computed from 10 experimental trials.

pseudo-labeling, resulting in worse performance, since DAL cannot benefit from unlabeled segments in this case. On the other hand, pseudo-labeling all unlabeled segments ($\Theta = 0$) also yields suboptimal performance, because segments are more likely to be assigned an incorrect pseudo-label.

5.3.2. *DAL performance sensitivity to α*

The impact of pseudo-labeled segments on the training depends on the value of α in (8), which regulates the amount of pseudo-labeled segments in a minibatch. Figure 3 depicts the performance of DAL for different values of α for labeling budget is between 30 and 130. It is evident that setting $\alpha = 0$ results in a suboptimal performance, since this prevents pseudo-labeled segments from appearing in a minibatch, as discussed in Section 4.2.3. In the case $\alpha = 1$ pseudo-labeled segments attain the same weight as labeled segments, which is known to degrade model accuracy due to mislabeled segments in the training dataset [23, 30]. In our experiments the value $\alpha = 0.01$ seemed to perform well, i.e. a pseudo-labeled segment is 100 times less likely to be drawn into a minibatch than a labeled segment with the same label. Given the large imbalance of data in favor of unlabeled segments it is reasonable that the sampling weight α of pseudo-labeled segments should be chosen small.

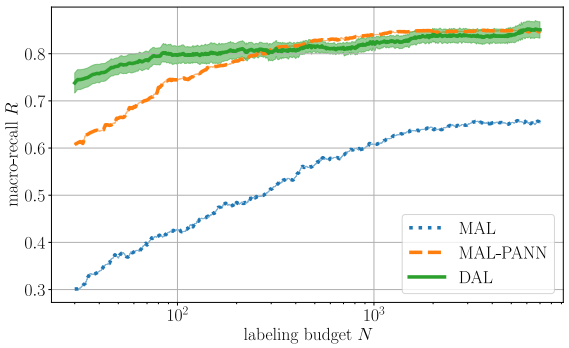


Figure 4: Macro-recall R over labeling budget N for baseline methods (MAL, MAL-PANN) and the proposed method (DAL). The confidence intervals for DAL are computed from 1 experimental trial whereby each cross-validation split is treated as an individual experiment. MAL and MAL-PANN are deterministic algorithms and their performance can be computed exactly.

5.3.3. *Performance of DAL vs baseline methods*

Using $\Theta = 0.5$ and $\alpha = 0.01$ determined in the previous experiments, Figure 4 depicts the performance of DAL against the labeling budget, now ranging from 30 to 7000. This figure also depicts the performance of the baseline methods (MAL, MAL-PANN).

First, it can be observed that simply switching from MFCC-based features as originally proposed in [14] to PANN embeddings greatly improves MAL performance, increasing the macro-recall for $N = 7000$ labels from about 65% (MAL) to about 85% (MAL-PANN). Second, we see that the proposed DAL method outperforms MAL for all considered labeling budgets and outperforms MAL-PANN (using the same features as DAL) for low labeling budgets (below 300), which is most relevant in practice.

6. CONCLUSION

In this paper, we proposed an active learning method for classifying sound segments that makes an efficient use of manual labels. The label-efficiency is established by a combination of active learning, self-training on pseudo-labels and transfer learning by means of using pre-trained embeddings.

The self-training aspect of DAL has a considerable influence on the classifier's accuracy. This is reflected in the performance sensitivity of DAL to the parameters controlling the pseudo-labeling policy and the pseudo-label weighting.

We have shown that the performance of the benchmark method, MAL, considerably improves when employing the same pre-trained PANN embeddings as in DAL, leading to a similar classification accuracy for larger labeling budgets. This indicates the importance of transfer learning that was applied in DAL.

In the experiments, the proposed method, DAL, outperforms benchmark methods especially for low labeling budgets.

In principle, DAL could be extended to the problem of multi-tagging, where a sound segment may have multiple class labels; this is a potential subject of future research. Furhtermore, a more complex strategy for assigning pseudo-labels could use adaptive confidence thresholds for each class to account for class imbalance.

The ability to perform approximate Bayesian inference via Monte Carlo dropout enables us to leverage model uncertainty and incorporate it into the AL process. Whether or not the employed acquisition function, vote entropy, is the best way of doing so, remains yet another open question.

7. REFERENCES

[1] W. Wang, *Machine Audition: Principles, Algorithms, and Systems*. Hershey, USA: Information Science Reference, 2011.

[2] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational Analysis of Sound Scenes and Events*. Springer, 2018.

[3] A. Mesaros, A. Diment, B. Elizalde, T. Heittola, E. Vincent, B. Raj, and T. Virtanen, "Sound Event Detection in the DCASE 2017 Challenge," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 27, no. 6, pp. 992–1006, 2019.

[4] J. Schröder, N. Moritz, J. Anemüller, S. Goetze, and B. Kollmeier, "Classifier Architectures for Acoustic Scenes and Events: Implications for DNNs, TDNNs, and Perceptual Features from DCASE 2016," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, no. 6, pp. 1304–1314, 2017.

[5] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Feature Learning With Matrix Factorization Applied to Acoustic Scene Classification," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, no. 6, pp. 1216–1229, 2017.

[6] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 279–283, 2017.

[7] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Audio Surveillance of Roads: A System for Detecting Anomalous Sounds," *IEEE Trans. Intell. Transport. Syst.*, vol. 17, no. 1, pp. 279–288, 2016.

[8] J. Laguarta, F. Hueto, and B. Subirana, "COVID-19 Artificial Intelligence Diagnosis Using Only Cough Recordings," *IEEE Open J. Eng. Med. Biol.*, vol. 1, pp. 275–281, 2020.

[9] S. Kahl, M. Clapp, W. Hopping, H. Goëau, H. Glotin, R. Planqué, W.-P. Vellinga, and A. Joly, "Overview of BirdCLEF 2020: Bird Sound Recognition in Complex Acoustic Environments," in *Proc. Conf. and Labs of the Evaluation Forum (CLEF)*, Thessaloniki, Greece, 2020.

[10] J. Salamon, E. Gomez, D. P. W. Ellis, and G. Richard, "Melody Extraction from Polyphonic Music Signals: Approaches, applications, and challenges," *IEEE Signal Process. Mag.*, vol. 31, no. 2, pp. 118–134, Mar. 2014.

[11] B. Settles, "Active Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[12] S. Budd, E. C. Robinson, and B. Kainz, "A survey on active learning and human-in-the-loop deep learning for medical image analysis," *Medical Image Analysis*, vol. 71, 2021.

[13] W. Han, E. Coutinho, H. Ruan, H. Li, B. Schuller, X. Yu, and X. Zhu, "Semi-Supervised Active Learning for Sound Classification in Hybrid Learning Environments," *PLoS ONE*, vol. 11, no. 9, 2016.

[14] Z. Shuyang, T. Heittola, and T. Virtanen, "Active learning for sound event classification by clustering unlabeled data," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 751–755.

[15] Y. Wang, A. E. Mendez Mendez, M. Cartwright, and J. P. Bello, "Active Learning for Efficient Audio Annotation and Classification with a Large Amount of Unlabeled Data," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, 2019, pp. 880–884.

[16] Z. Shuyang, T. Heittola, and T. Virtanen, "An Active Learning Method Using Clustering and Committee-Based Sample Selection for Sound Event Classification," in *Proc. Int. Workshop on Acoustic Signal Enhancement (IWAENC)*, Tokyo, Japan, 2018, pp. 116–120.

[17] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *Proc. Int. Conf. on Machine Learning (ICML)*, New York, USA, 2016, pp. 1050–1059.

[18] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," in *Proc. Int. Conf. on Machine Learning (ICML)*, Lille, France, 2015, pp. 1613–1622.

[19] D. P. Kingma, T. Salimans, and M. Welling, "Variational Dropout and the Local Reparameterization Trick," in *Proc. Int. Conf. on Neural Information Processing Systems (NeurIPS)*, Cambridge, USA, 2015, pp. 2575–2583.

[20] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational Dropout Sparsifies Deep Neural Networks," in *Proc. Int. Conf. on Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 2498–2507.

[21] C. Louizos, K. Ullrich, and M. Welling, "Bayesian Compression for Deep Learning," in *Proc. Int. Conf. on Neural Information Processing Systems (NeurIPS)*, New York, USA, 2017, pp. 3290–3300.

[22] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian Active Learning with Image Data," in *Proc. Int. Conf. on Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 1183–1192.

[23] M. Rottmann, K. Kahl, and H. Gottschalk, "Deep Bayesian Active Semi-Supervised Learning," in *Proc. IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, Orlando, USA, 2018, pp. 158–164.

[24] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 2880–2894, 2020.

[25] J. Salamon, C. Jacoby, and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research," in *Proc. ACM Int. Conf. on Multimedia (ACMMM)*, Orlando, USA, 2014, pp. 1041–1044.

[26] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 776–780.

[27] D.-H. Lee, "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks," *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 2013.

[28] W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler, "The Power of Ensembles for Active Learning in Image Classification," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, 2018, pp. 9368–9377.

[29] I. Dagan and S. P. Engelson, "Committee-Based Sampling For Training Probabilistic Classifiers," in *Proc. Int. Conf. on Machine Learning (ICML)*. San Francisco, USA: Elsevier, 1995, pp. 150–157.

[30] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020.

[31] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: An Overview," *ArXiv200805756 Cs Stat*, Aug. 2020.

MULTI-SCALE NETWORK BASED ON SPLIT ATTENTION FOR SEMI-SUPERVISED SOUND EVENT DETECTION

Xiujuan Zhu[1,3], *Ying Hu*[1,3*], *Xinghao Sun* [1,3], *Liang He*[1,2],

[1] School of Information Science and Engineering, Xinjiang University, Urumqi, China
{xiujuanzhu841}@gmail.com, {huying}@xju.edu.cn
[2] Department of Electronic Engineering, Tsinghua University, China
[3] Key Laboratory of Signal Detection and Processing in Xinjiang, China

ABSTRACT

Sound scene in real environment is generally composed of different types of sound events meanwhile the time-frequency scales of these events are diverse. Thus, it is important to design a proper mechanism to extract the multi-scale features for sound event detection (SED). In order to improve the discriminative ability of different types of sound events, we propose a multi-scale SED network based on split attention. We design a Multi-scale (MS) module to extract the fine-grained and the coarse-level features in parallel. A Channel Shuffle (CS) operation is introduced to enhance the cross-channel information communication among the features with different scales. Also, a Split Attention (SA) module is designed to learn several sub-features separately and an attention mechanism is followed to generate the corresponding importance coefficients for each sub-features. Experiments on DCASE2021 Task4 dataset demonstrate the effectiveness of our proposed multi-scale network.

Index Terms— sound event detection, multi-scale, channel shuffle, split attention

1. INTRODUCTION

The purpose of sound event detection (SED) is to identify the categories of sound events and detect the onset and offset of the target events in an audio sequence. Unlike audio classification task that it only needs to determine the event categories, detection task also needs to predict the temporal position of occurring events. Thus, SED is a more difficult task. SED has drawn great attention recently in a variety of applications, such as surveillance [1], smart cities and homes [2], [3], as well as multimedia information retrieval [4]. There are three kinds of learning approaches in SED: fully supervised SED, weakly supervised SED and semi-supervised SED. Following the baseline of DCASE2021 Task4 , this paper only focuses on semi-supervised SED based on mean teacher method [5].

Real-life SED is challenging since different sound events exhibit different time-frequency properties. For example, "Dog" and "Dishes" last shorter while "Running water" and "Blender" last longer in the time domain and cover a wider frequency range. If the model performs on a single resolution, it's hard to deal with the different types of sound events. Thus, how to obtain the multi-scale features and integrate the features with inconsistent scales is a key point in SED.

Multi-scale mechanism has drawn great attention in SED task. Zhang et al. [6] proposed Multi-Scale Time-Frequency Attention module to extract the information at multiple resolutions. Ding et al. [7] further proposed an multi-scale detection method based on Hourglass network. The mechanism of Feature pyramid [8] has proved to be useful to obtain multi-resolution features in SED [9] , [10]. Another way to get multi-scale features is to use dilated convolution. Li et al. [11] proposed a dilated convolution recurrent neural network (CRNN) to verify the effectiveness of different dilation rates in convolution layers. Drossos et al. [12] proposed to use dilated convolution instead of GRU to capture long temporal context. Different from the above mentioned methods, in this paper, the multi-scale is only reflected from the convolution kernels of different sizes, it is a relatively simple structure. Su et al. [13] proposed a channel shuffle module to promote cross-channel information communication between the high-level and low-level information. Zhang et al. [14] proposed the ResNeSt based on the split-attention and proved its effectiveness. The group learning mechanism in split-attention ensures the network only to learn sub-features in adjacent channels. Wang et al. [15] also showed that the channel features are mainly related to their adjacent channel features while little related to the remote channel features.

Inspired the above related works, we propose a multi-scale SED network based on split attention. The multi-scale module exploits convolution kernels of different sizes to learn the multi-scale features in parallel, which improves its ability to recognize sound events. Motivated by [13], the channel shuffle operation is adopted to enable the cross-channel information flowing among the features with different scales. Inspired by ResNeSt [14], we adopt split attention module based on group convolution to separately learn sub-features and also generate attention weights to re-weight these sub-features.

This paper is organized as follows. We introduce the proposed SED network in Section 2, describe the dataset and evaluation metrics in Section 3, analyze the experimental results in Section 4, and conclude the paper in Section 5.

2. PROPOSED METHOD

In this section, we firstly present the overall network structure. Then we separately introduce the proposed multi-scale module, channel shuffle operation and split attention module.

2.1. Network Architecture

As illustrated in Figure 1, the proposed network adopts CRNN as the backbone architecture. It mainly consists of three parts: multi-scale feature extraction part, bi-directional GRU (Bi-GRU) and lo-

*equal contribution with Xiujuan Zhu

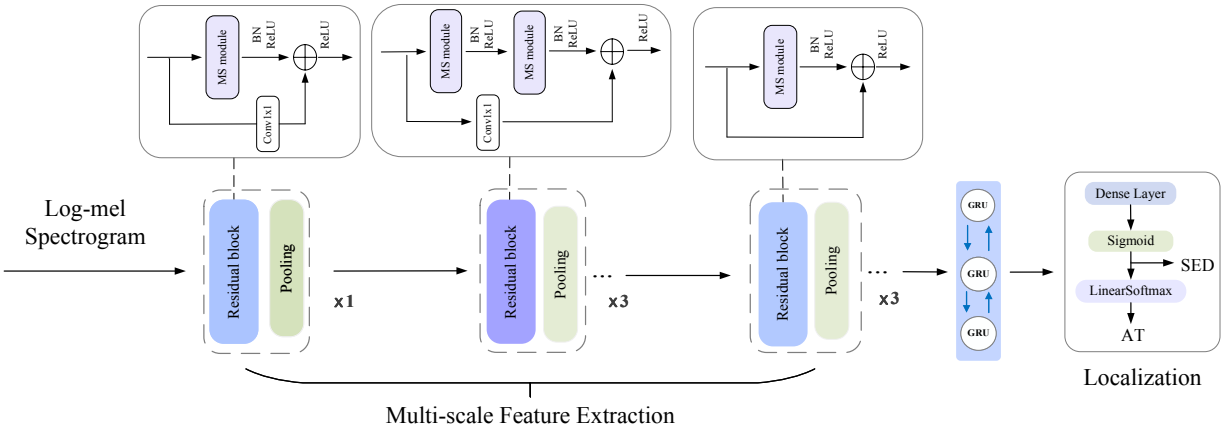


Figure 1: The overall architecture of the Multi-scale network based on Split Attention.

calization parts. The multi-scale feature extraction part is based on 7 residual blocks, each block followed by a pooling layer. In the first one and the last three residual blocks, each of them consists of one multi-scale (MS) module shown in Figure 2, and in the mid three residual blocks, each of them consists of two MS modules. Then a Bi-GRU is used to capture temporal information. The localization part produces frame-level predictions for SED and clip-level predictions for audio tagging (AT). Note that linear softmax [16] is introduced as an aggregation function to produce the clip-level predictions.

2.2. Multi-scale Module

In order to effectively model time-frequency context information, the multi-scale module exploits convolution kernels of different sizes to extract the features of different scales.

As shown in Figure 2, where a three-branch case is shown, each branch used to learn one single-scale feature map . Thus, multi-scale module can process the input feature at multiple scales in parallel. For a given feature map $X \in \mathbb{R}^{C\times H\times W}$, it firstly undergoes three kinds of scale transformations based on different kernel sizes k_i, thus $[X_1, X_2, X_3]$ are obtained. $X_i \in \mathbb{R}^{C\times H\times W}$ represents a specific scale feature can be generated as:

$$X_i = \mathrm{SA}(X, k_i), i = 1, 2, 3 \tag{1}$$

where SA denotes split attention module that is going to be described in details in Section 2.4 . k_i denotes the kernel size used in SA module. Then a pre-processed multi-scale feature $F' \in \mathbb{R}^{3C\times H\times W}$ is obtained by concatenating the multi-scale features X_i:

$$F' = \mathrm{Concat}([X_1, X_2, X_3]) \tag{2}$$

where Concat means the concatenation operation along the channel dimension.

In order to help the network learn a better multi-scale feature, a channel shuffle operation is applied to F' that it improves the information flowing among the features with different scales. A convolution layer with the kernel size of 1×1 is followed to change the channel numbers of output features. Thus, the final output features $F \in \mathbb{R}^{C\times H\times W}$ can be obtained by:

$$F = \mathrm{Conv1} \times 1(\mathrm{CS}(F')) \tag{3}$$

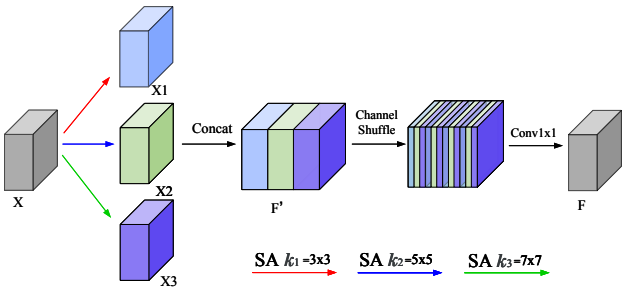


Figure 2: Illustration of our proposed multi-scale (MS) module. SA denotes split attention module.

Where CS denotes channel shuffle operation.

2.3. Channel Shuffle operation

In [17], channel shuffle operation can be used to improve the information flowing among the feature within different groups. In this paper, channel shuffle operation aims to enhance the cross-channel information communication among the features with different scales. A channel shuffle operation can be modeled as a process composed of "Reshape-Transpose-Reshape" operations. As shown in Figure 2 , the channel dimension of F' is reshaped to (g, c), where g is the number of groups, $c = 3C/g$. The channel dimension is further reshaped to (c, g) and then flatten back to $3C$. Through this operation, the channel information among different features can interact with each other.

2.4. Split Attention Module

As shown in Figure 3 , inspired by group convolution (GN) [18], SA module firstly adopts group convolution to learn different sub-features, which represent diverse semantic features such as different sound event patterns. Then, in order to measure the importance of different sub-features, a set of attention weights W_i corresponding to each sub-features are generated. This process can be abstracted into two parts: **Group**, **Attention**.

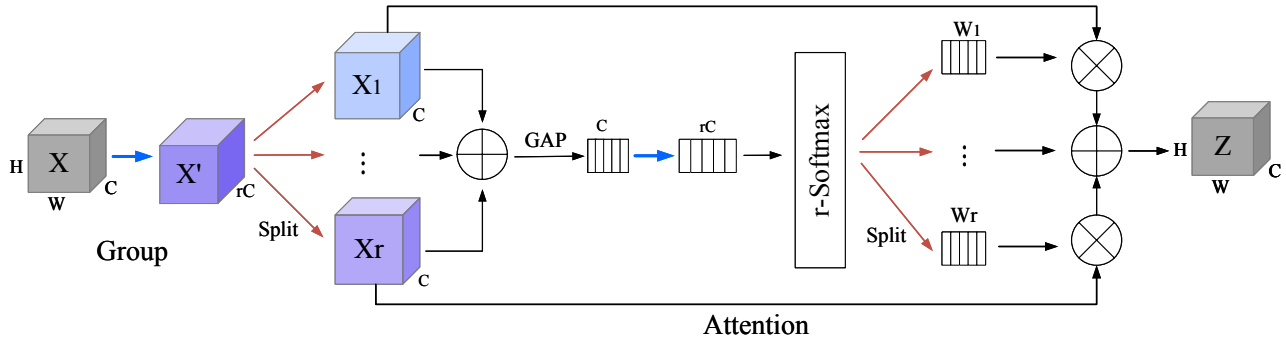


Figure 3: Illustration of the proposed split attention (SA) module. Blue arrow denotes group convolution operation, red arrow split operation along the channel dimension

Group: Assuming an input feature map $X \in \mathbb{R}^{C\times H\times W}$, we firstly adopt the group convolution to learn g sub-features in different groups separately. As a result, the C-channel feature map X is expanded into the rC-channel feature map $X' \in \mathbb{R}^{rC\times H\times W}$. Then the expanded feature map X' is split into r branches along the channel dimension that represented as $[X_1, ..., X_i..., X_r]$. $X_i \in \mathbb{R}^{C\times H\times W}$. $i \epsilon 1, 2, ..., r$. The number of group g and branch r will be discussed in the experiment.

Attention: Multiple sub-features $[X_1, ..., X_i..., X_r]$ are firstly fused via an element-wise summation $U = \sum_{i=1}^{r} X_i$. Then, global average pooling is calculated to squeeze the fused feature U into a channel-wise statistics $S \in \mathbb{R}^{C\times 1\times 1}$:

$$S = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} U(i', j') \tag{4}$$

Then, a simple attention mechanism with a Softmax function is performed on the channel-wise statistics S. The attention weight $W \in \mathbb{R}^{rC\times 1\times 1}$ can be obtained by:

$$W = r - Softmax((Conv(\delta(BN(Conv(S)))))) \tag{5}$$

Where δ means the ReLU activation function, BN the batch normalization, $Conv$ the group convolution with the kernel size of 1×1 and group number is g. The attention weight $W \in \mathbb{R}^{rC\times 1\times 1}$ is then splitted into a set of attention weights $W_i \in \mathbb{R}^{C\times 1\times 1}$.

Finally, by applying the weights W_i to the sub-features X_i, the output feature map of SA module $\mathrm{Z} \in \mathbb{R}^{C\times H\times W}$ is obtained by:

$$\mathbf{Z} = \sum_{i=1}^{r} S_i \times X_i \tag{6}$$

3. EXPERIMENTS

3.1. Dataset and Experimental setup

The audio samples in the DCASE2021 Task4 dataset are 10s clips recorded in domestic environment or synthesized to simulate a domestic environment. It contains 10 kinds of sound events. Three types data (i.e. the weakly labeled data (1578), unlabeled data (11412) and strong labeled data (10000)) are used for training. The ratio among them is 1:2:1 in each batch. The performance of the proposed method is evaluated on the validation data (1168).

Following the default experiment settings of DCASE2021 Task4 baseline, we also take the log-mel energies as input extracted with 128 mel-scale filters. The window length is 2048 with the hop size of 256. The audio is resampled to 16kHz. The training is set for 200 epochs using the Adam optimizer with an initial learning rate of 0.001. A learning rate exponential warmup [19] during the first 50 epochs is used. A detection threshold is fixed to 0.5 for each class. The binary SED predictions are further processed with a 7 frames median filter. For the 7 residual blocks in multi-scale extraction part, the number of channels for each residual block is [16, 32, 64, 128, 128, 128, 128], respectively and the pooling size is [[1, 2], [1, 2], [2, 2], [2, 2], [1, 2], [1, 2], [1, 2]], respectively. The dropout rate is 0.3. Note that due to the continuous pooling operation along the frequency dimension, its receptive field along the frequency dimension keeps increasing. Thus, in the last three residual blocks, the kernel sizes used in MS module are set to [[3, 3], [5, 3], [7,3]], that is the kernel sizes used in time dimension keep different values, while in frequency dimension the same.

3.2. Loss Function

The loss function for training the model is a sum of four loss components: two binary cross entropy (BCE) losses for supervised training and two mean square error (MSE) losses for consistency training, which are combined as follows:

$$\begin{aligned}\mathcal{L}(\theta) =& \mathcal{L}_{BCE}(sw_{out}, l_w) + \sigma(\lambda)\mathcal{L}_{MSE}(sw_{out}, tw_{out}) \\ & \mathcal{L}_{BCE}(ss_{out}, l_s) + \sigma(\lambda)\mathcal{L}_{MSE}(ss_{out}, ts_{out})\end{aligned} \tag{7}$$

Where sw_{out}, ss_{out} denote the AT output and SED output of the student model, respectively, tw_{out}, ts_{out} the AT output and SED output of the teacher model, l_w and l_s the weakly label and strong label of the labeled data.

3.3. Evaluation metrics

In DCASE2021 Task4, the evaluation metrics include PSDS-scenario1 (PSDS1), PSDS-scenario2 (PSDS2), Intersection-based F1 (IB-F1) and Collar-based F1 (CB-F1). The PSDS1 measures the model's capability of detecting the onset and offset of the event within an audio clip, and the PSDS2 measures that of avoiding confusion among the event classes. More details about PSDS evaluation metrics can refer to [20]. IB-F1 and CB-F1 are used as sup-

Table 1: Ablation experiments on multi-scale (MS) mechanism with different kernel sizes. We adopt vanilla convolution instead of SA module in MS module of all residual block in this experiment.

Network	PSDS1	PSDS2	IB-F1(%)	CB-F1(%)	Parameter
Base-2021	0.342	0.527	76.60	40.10	1.1M
MS-K=[3]	**0.358**	0.599	81.88	**44.48**	1.2M
MS-K=[3,5]	0.349	**0.602**	83.24	44.13	3.0M
MS-K=[3,5,7]	0.336	0.601	**83.50**	42.21	5.7M

Table 2: Ablation experiments on channel shuffle (CS) operation based on MS-K=[3,5] system. CS-g denotes the channel shuffle operation with g groups. g controls the fusion degree of features.

Network	PSDS1	PSDS2	IB-F1 (%)	CB-F1 (%)
MS-K=[3, 5]	0.349	0.602	**83.20**	44.13
+ CS-g=2	0.349	0.594	82.83	43.58
+ CS-g=4	**0.358**	**0.606**	82.98	**45.36**

plementary evaluation metrics to validate a model's performance in SED. For all these metrics, the value larger, the performance better.

4. RESULTS AND ANALYSIS

We separately investigate the contribution of each component to the overall network, including the multi-scale mechanism with different kernel sizes, the channel shuffle operation and the split attention module. All experiments are repeated 4 times and the average result of these experiments is reported.

Evaluations of MS mechanism

Table 1 shows the SED performance of the MS mechanism with difference kernel sizes. MS-K=[3] means there is only one branch with the kernel size of 3×3 in MS module, and MS-K=[3, 5] denotes there is two branches with the kernel sizes of 3×3 and 5×5. MS-K=[3, 5, 7] means exactly the processing depicted in Figure 2 but no channel shuffle operation. Experimental results show that our proposed MS network outperform the baseline of DCASE2021 Task4 [21] in terms of four evaluation metrics, demonstrating the effectiveness of the multi-scale mechanism for SED. However, compared with MS-K=[3], the performance of network applying two types of convolution kernels (MS-K=[3, 5]) or three types of convolution kernels (MS-K=[3, 5, 7]) has barely improved. The reason for this phenomenon may be that the network does not handle the features of different scales well.

Evaluations of CS operation

Table 2 lists the results of our proposed network with channel shuffle operation. In particular, compared with the network without CS operation denoted as MS-K=[3,5], the network applying CS operation with 4 groups achieve a better performance in terms of all evaluation metrics except IB-F1. This result demonstrates the effectiveness of channel shuffle operation.

Evaluations of SA module

Table 3 lists the results of network applied SA module. In this experiment, we only adopt vanilla convolution in MS module of the 1-th residual block, while SA module in MS module of the rest residual blocks. Compared with the results of first row, we can see that the network with split attention module achieves significantly improvement in terms of all evaluation metrics expect PSDS2 metric. The results demonstrate the effectiveness of SA module for SED. However, Table 3 shows that three are no significant difference among networks with different SA module on PSDS2 and IB-F1 metrics. Compared with the results between the second and fourth row or the third fifth row, we can see that the network applying group convolution with 2 group can achieve a better performance on PSDS1 and CB-F1 metrics than without applying group operation. This manifests that adopting group operation to learn sub-features is effective. Compared with the results between the fourth and fifth row, we can find that the performance of network splitting 2 sub-features (r=2) is better than without splitting operation in SA module. This indicates that generating attention weights to treat the learned sub-features differently is important. From the results of the last row, the performance of network applying channel shuffle get further improvements in terms of four metrics except PSDS1. It also shows the effectiveness of CS operation.

Table 3: Ablation experiments on split attention (SA) module based on MS-K=[3,5] system. SA(g, r) means the number of group is g, splitted sub-feaatures r in shuffle attention module.

Network	PSDS1	PSDS2	IB-F1(%)	CB-F1(%)	Parameter
MS-K=[3, 5]	0.349	0.602	83.24	44.13	3.0M
+ SA(1, 1)	0.354	0.598	**84.59**	47.99	3.2M
+ SA(1, 2)	0.350	0.602	84.40	46.64	5.5M
+ SA(2, 1)	0.367	**0.606**	83.80	48.59	1.9M
+ SA(2, 2)	**0.376**	0.599	83.63	49.02	3.3M
+ CS-g=4	0.373	0.602	83.98	**50.28**	3.3M

5. CONCLUSION

In this paper, we propose a multi-scale SED network based on split attention that it can deal with the short- or long- duration sound events. Multi-scale module can learn features with multiple scales in parallel. Specifically, channel shuffle operation is used to promote the cross-channel information flowing among the features with different scales. Split attention module can learn the different sub-features separately and generate attention used to weight the importance of sub-features. A set of experiments are conducted to verify their effective. The final results of the proposed network outperform the baseline of DCASE2021 Task4 significantly. In our future work, we would like to explore the issue that how to deal with the features with different scales in SED.

6. ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China (NSFC) (61761041,U1903213), Tianshan Innovation Team Plan Project of Xinjiang (202101642)

7. REFERENCES

[1] E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-based classification, search, and retrieval of audio," *IEEE Multim.*, vol. 3, pp. 27–36, 1996.

[2] J. P. Bello, C. Mydlarz, and J. Salamon, "Sound analysis in smart cities," in *Computational Analysis of Sound Scenes and Events*. Springer, 2018, pp. 373–397.

[3] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring activities of daily living in smart homes: Understanding human behavior," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 81–94, 2016.

[4] Q. Jin, P. Schulam, S. Rawat, S. Burger, D. Ding, and F. Metze, "Event-based video retrieval using audio," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[5] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NIPS*, 2017.

[6] J. Zhang, W. Ding, J. Kang, and L. He, "Multi-scale time-frequency attention for acoustic event detection," *arXiv preprint arXiv:1904.00063*, 2019.

[7] W. Ding and L. He, "Adaptive multi-scale detection of acoustic events," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 294–306, 2019.

[8] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[9] J. Yan, Y. Song, W. Guo, L. Dai, I. Mcloughlin, and L. Chen, "A region based attention method for weakly supervised sound event detection and classification," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 755–759, 2019.

[10] J. Yan, Y. Song, L. Dai, and I. Mcloughlin, "Task-aware mean teacher method for large scale weakly labeled semi-supervised sound event detection," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 326–330, 2020.

[11] Y. Li, M. Liu, K. Drossos, and T. Virtanen, "Sound event detection via dilated convolutional recurrent neural networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 286–290.

[12] K. Drossos, S. I. Mimilakis, S. Gharib, Y. Li, and T. Virtanen, "Sound event detection with depthwise separable and dilated convolutions," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[13] K. Su, D. Yu, Z. Xu, X. Geng, and C. Wang, "Multi-person pose estimation with enhanced channel-wise and spatial information," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5667–5675, 2019.

[14] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z.-L. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, "Resnest: Split-attention networks," *ArXiv*, vol. abs/2004.08955, 2020.

[15] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "Eca-net: Efficient channel attention for deep convolutional neural networks," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 531–11 539, 2020.

[16] Y. Wang, J. Li, and F. Metze, "A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 31–35.

[17] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84 – 90, 2012.

[19] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *ArXiv*, vol. abs/1706.02677, 2017.

[20] Ç. Bilen, G. Ferroni, F. Tuveri, J. Azcarreta, and S. Krstulović, "A framework for the robust evaluation of sound event detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 61–65.

[21] http://dcase.community/challenge2021/task-sound-event-detection-and-separation-in-domestic-environments.

LEVERAGING STATE-OF-THE-ART ASR TECHNIQUES TO AUDIO CAPTIONING

Chaitanya Narisetty[1], *Tomoki Hayashi*[2], *Ryunosuke Ishizaki*[2],
Shinji Watanabe[1], *Kazuya Takeda*[2]

[1] Carnegie Mellon University, Pittsburgh, USA,
cnariset@andrew.cmu.edu, shinjiw@ieee.org
[2] Nagoya University, Nagoya, Japan,
{hayashi.tomoki, ishizaki.ryunosuke}@g.sp.m.is.nagoya-u.ac.jp,
takeda@is.nagoya-u.ac.jp

ABSTRACT

This paper details our work towards leveraging state-of-the-art ASR techniques for the task of automated audio captioning. Our model architecture comprises of a convolution-augmented Transformer (Conformer) encoder and a Transformer decoder to generate natural language descriptions of acoustic signals in an end-to-end manner. To overcome the limited availability of captioned audio samples for model training, we incorporate the Audioset-tags and audio-embeddings obtained from pretrained audio neural networks (PANNs) as an auxiliary input to our model. We train our model over audio samples from Clotho & AudioCaps datasets, and test over Clotho dataset's validation and evaluation splits. Experimental results indicate that our trained models significantly outperform the baseline system from DCASE 2021 challenge task 6.

Index Terms— Automated Audio Captioning, Conformer, ESPNet, PANNs

1. INTRODUCTION

Automated audio captioning was first proposed by [1] as a task of generating descriptive captions for a give audio signal using the concepts of audio processing and natural language processing. Datasets for this task consist of audio samples mapped to at least one corresponding human-generated caption [2, 3]. To generate a caption of sufficient quality, it is essential that the training model distills meaningful audible representations from an audio signal.

Similar to established image caption generators [4], a typical audio captioning model also comprises of an encoder-decoder framework. The encoder computes an encoded representation of relevant acoustic features in an input audio sample, and the decoder outputs a sequence of tokens using the encoded representation to form a suitable descriptive caption [1, 5]. Popular and effective frameworks for audio captioning in literature comprise of CNN encoders and Transformer decoders. A 10-layer CNN encoder and a Transformer decoder with multi-head self-attention was proposed by [5], where the CNN encoder was first pretrained for a multi-label classification task. To overcome the issue of limited number of training samples, [6] used a mix-up based data augmentation to create training samples from convex combinations of two given audio samples and their word token embeddings. Reinforcement learning in the form of self-critical sequence training (SCST), introduced for image captioning [7], was also explored for audio captioning by [8] to directly optimize the evaluation metrics (BLEU, CIDEr etc.) instead of the cross-entropy loss during greedy decoding at test-time.

Our proposed method is based on state-of-the-art automatic speech recognition (ASR) techniques such as convolution-augmented Transformer (Conformer) [9] and the fusion of a language model, incorporated in the end-to-end speech processing toolkit ESPnet [10]. Furthermore, we utilize the pretrained audio tagging model PANNs [11] to extract auxiliary information (e.g., Audioset [12] tags and embedding vector) and integrate them with the ASR model, enabling us to generate consistent captioning results. The contributions of this paper are as follows:

- We apply an attention-based encoder-decoder with the Conformer architecture, which allows capturing both local and global contexts in the input sequence. We also employ the language model trained on the captions and integrate its score with shallow fusion, resulting in a more stable prediction.
- We also introduce a pretrained audio tagging model PANNs to extract the auxiliary information, including Audioset tags and embedding vectors, and then utilize them as the additional inputs for the encoder-decoder model.
- Experimental evaluation with DCASE 2021 Task 6 dataset [13] shows that the proposed framework significantly outperforms the baseline system. Our best trained model shows a SPIDEr score of 0.224 and 0.246 on the development-validation and development-evaluation sets, respectively.
- This work expands on our DCASE2021 challenge report [14] with detailed description of the proposed framework and key insights into the contributions of auxiliary input features and language model fusion.
- Towards supporting accessible and reproducible research, we intend to release our audio captioning system and pretrained models to the ESPNet toolkit[1].

2. PROPOSED METHODOLOGY

2.1. Overview

Fig. 1 illustrates an overview of the proposed method. Similar to other speech-related tasks, we use log-mel filterbank features as the primary input. Data augmentation is performed over these

[1] https://github.com/chintu619/espnet/tree/aac_wordtokens/egs/clotho/aac_word

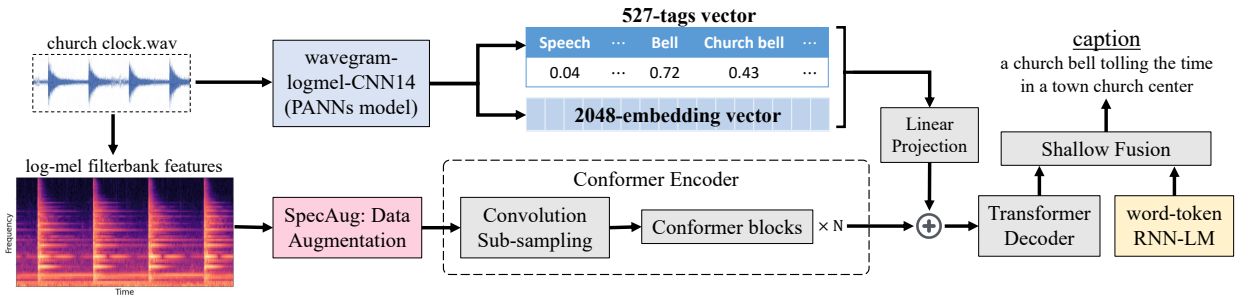


Figure 1: An overview of the proposed network architecture based on a Conformer encoder and a Transformer decoder. SpecAug based data augmentation is performed on the log-mel filterbank features. The pretrained wavegram-logmel-CNN14 PANNs model extracts the 527-tags vector and 2,048-embedding vector, and are fed as auxiliary inputs. Finally, a shallow fusion of decoder output and RNN-LM is performed to generate the output caption.

primary features to improve noise robustness. In addition to the primary input, we employ auxiliary inputs such as Audioset tags and an embedding vector, which are extracted with the pretrained audio tagging model PANNs [11]. Both inputs are fed into the attention-based encoder-decoder model. Inspired by the success of Conformer-based models for tasks like speech recognition, translation, and separation [15], our model uses a Conformer encoder for processing these audio features and a Transformer decoder to process words in a corresponding caption. To further improve the performance, we introduce the RNN-based language model and combine it with the encoder-decoder model in the decoding stage. The following subsections describe each of the components of our proposed Conformer model.

2.2. Encoder-Decoder Framework

The encoder incorporates a convolution sub-sampling layer and several Conformer blocks, where each block consists of a first feed-forward module (FFN), a multi-head self-attention (MHSA) module, a convolution module and a second feed-forward module in the aforementioned sequence. Similar to Transformer ASR models, a residual connection is added to the output of the feed-forward module followed by a layer normalization (LN) [16]. To regularize the network, the module employs dropout and Swish activation [17].

The self-attention module uses relative positional encoding in order to make the encoder robust to varying input length. This feature makes Conformer an ideal encoder for audio samples of varying length as seen in the present task. This module also employs dropout and a residual connection to regularize the network. For an input sequence $\mathbf{X} \in \mathbb{R}^{T \times d^{att}}$, where T is the number of time frames and d^{att} is the attention dimension, the positional encoding and regularization are computed according to Eq. 1. Finally the convolution module employs a point-wise convolution, a gated linear unit (GLU) activation [18], 1-dim depth-wise convolution layer, a batch normalization layer, Swish activation and a point-wise convolution. Both feed-forward modules employ a half-step scheme, and a residual connection and dropout are again used for regularization as shown in Eq. 2.

$$\mathbf{X} = \mathbf{X} + \text{Dropout}(\text{MHSA}(\text{LN}(\mathbf{X}))) \quad (1)$$

$$\mathbf{X} = \mathbf{X} + 0.5 \times \text{Dropout}(\text{FFN}(\text{LN}(\mathbf{X}))) \quad (2)$$

The decoder also incorporates several Transformer blocks, where each block consists of a multi-head self-attention layer, and a linear layer with ReLU activation sandwiched between two layer normalization layers.

2.3. Auxiliary Input Features

To improve the generalization ability of our model, we provide an auxiliary input to our encoder framework, similar to the use of robust audio embeddings in speaker recognition tasks [19]. For this purpose, we use CNN14 - one the PANNs models trained on the large scale Audioset dataset of over 5,000 hours of audio samples labeled with 527 audio tags. The CNN14 model is a wavegram-logmel-CNN system trained on 32kHz audio samples using 14 convolution layers. The model outputs a 527-tags vector, whose each element corresponds to the prediction of an audio tag. In addition to this 527-tags vector, we also extract a 2,048-embedding vector from each audio sample that is inputted to final classification layer.

The tags and/or embeddings obtained using PANNs are used as an auxiliary input to our model. When using both the tags and embeddings, the two feature vectors are simply concatenated to form a single column vector. These features are first L2 normalized and then passed through a feed-forward layer to be projected to the same size as that of the attention dimension. The projected features are finally added to the output of the Conformer encoder, before being sent as an input to the Transformer decoder.

2.4. Shallow Fusion with Language Model

We also separately train a word-token RNN language model (RNN-LM) using the captions in the training data and integrate it with the decoder using shallow fusion [20]. During inference, for each partial hypothesis h, the decoder combines its attention scores $\alpha_{att}(h)$ with the look-ahead word-token scores $\alpha_{lm}(h)$ provided by RNN-LM according to Eq. 3, where γ is a scaling factor.

$$\alpha(h) = \alpha_{att}(h) + \gamma \cdot \alpha_{lm}(h) \quad (3)$$

3. EXPERIMENTS

3.1. Data Preparation and Pre-processing

Our proposed model takes 16 kHz audio samples as input and computes 80 log-mel energies from each 64 ms frame, shifted every 32 ms. Accordingly, all the audio files in Clotho-v2 dataset were down-sampled from 44.1 kHz to 16 kHz. The overall development split of the Clotho-v2 dataset has 3,839 training samples, 1,045 validation samples and 1,045 evaluation samples. Each audio sample is 15-30

seconds long and contains 5 human generated captions with 8-20 words each. Since the Clotho-v2 dataset is relatively small to train large neural networks, we additionally augment the training data with roughly 46,000 single caption audio samples from the AudioCaps dataset [3]. Audio samples in this dataset are carefully chosen from the 2M samples in Audioset dataset [12]. Each audio sample is roughly 10 seconds long.

We perform input feature augmentation using SpecAug [21] consisting of three kinds of deformations - time warping, frequency masking and time masking. We set the maximum time warp parameter to $W = 5$, and randomly choose $w \in [0, W]$ such that the log-mel filterbank feature matrix is warped by w. Frequency and time masking are based on Cutout [22] regularization technique which masks a randomly chosen rectangular portion of the log-mel filterbank matrix. Dimensions of the mask were chosen randomly based on the maximum frequency and time masking parameters of $F_m = 30$ and $T_m = 40$ respectively.

3.2. Comparison Models and Training Parameters

3.2.1. Baseline System

The DCASE 2021 challenge task 6 provided a baseline encoder-decoder framework consisting of a 3 layer bi-directional GRU encoder, and a decoder with one GRU layer and one classification layer. The input acoustic features are extracted using 64 log-mel energies estimated over a 46ms frame, shifted every 23ms. Each encoder and decoder GRU layer has 256 bi-directional features, and the classification layer outputs the probability of 4637 unique words in each decoder iteration (time-step).

3.2.2. Proposed Model

The proposed Conformer model used in our experiments has 16 encoder layers and 4 decoder layers, each with 1,024 units along with 4 heads, $d^{att} = 256$ for attention layers and a depth-wise convolution with kernel size of 15. For better predictive performance through model ensemble, we also explored several variations in the dimensions of the proposed model. Decoding module for a model ensemble performs posterior averaging of the attention score output of constituent model decoders. A variation of the proposed Conformer model was trained with smaller encoder-decoder layers having 512 units each. Another variation was trained with a smaller attention framework having 128-dim layers with 2 heads. Final model variation was trained with above mentioned smaller attention framework, but with a larger kernel size of 31.

In addition to the log-mel energies, we extract a softmax vector of 527-tags and a 2,048-embedding vector from each audio sample using the CNN14 PANNs model [11] as detailed in Section 2.3. Each element of 527-tags vector represents the probability of a corresponding class-label in the Audioset ontology. All the proposed model variations employ shallow fusion using a 2-layer RNN-LM trained for 25 epochs with a batch-size of 64 and dropout of 0.5. Scaling factor γ for shallow fusion is set to 0.2.

3.2.3. Hyper-Parameters

During training, 64 audio-caption pairs were batched together and trained for 50 epochs with a learning-rate of 0.5, dropout of 0.1, cross-entropy loss function and *noam* optimizer [23]. To prevent exploding gradients, we set the gradient threshold to 5. Label smoothing [24] was set to 0.1 to avoid high confidence training predictions. Upon completion of training, we average the model parameters over the final-10 epochs and this averaged model was used for inference. During inference, beam search was performed with a beam-size of 10 and RNN-based language model weight of 0.2. We note that the above hyper-parameters are optimized based on our prior experience in tuning ASR systems.

3.3. Evaluation Metrics

Experimental evaluation for audio captioning is conducted using six metrics: BLEU-n [25], ROUGE-L [26], METEOR [27], CIDEr [28], SPICE [29] and SPIDEr [30]. Precision of output captions for 1,2,3,4-grams (contiguous sequence of n words) are evaluated by BLEU-n. F-measure between output and ground-truth captions is estimated by ROUGE-L by estimating their longest common subsequence. METEOR is a machine translation metric which computes a harmonic mean of 1-gram precision and recall between output and ground-truth captions. CIDEr computes the average cosine similarity of n-grams between output and ground-truth captions. SPICE evaluates the semantic similarity between output and ground-truth captions by first performing lemmatisation of captions and then computing the F-score between their scene graphs. Lemmatisation maps all the inflected forms of a word to its root form, and scene graphs are a semantic representation which encode the objects, attributes and relations present in captions. SPIDEr simply computes an average score of CIDEr and SPICE metrics.

4. RESULTS

The performance of our trained models were evaluated on both the development-validation and development-evaluation splits and are summarized in Table 1 and Table 2. All our proposed models outperform the DCASE 2021 baseline system by a significantly margin. Summarized results also show the contribution from various components of our proposed model: encoder-decoder, self-attention and auxiliary features.

4.1. Observations

We observe a slight degradation in performance when varying our model's architecture as compared to the baseline Conformer model. However these variations help to improve the performance of a model ensemble. Auxiliary input features of tags and embeddings were able to improve the scores of most metrics, especially over the development-validation split. We also observe that augmenting the training data with the development-evaluation split was indeed able to improve the proposed Conformer's performance over the development-validation split and vice-versa. Model ensemble was also performed over various combinations of our trained models, and was further able to increase the overall system performance.

4.2. Discussion

4.2.1. Understanding Auxiliary Features

We explore the individual contribution of extracted tags and embeddings towards the performance boost provided by the auxiliary input features. Table 3 details the performance of the proposed Conformer model when trained with only the extracted 2,048-embeddings and 527-tags as secondary inputs. Although the extracted tags provide sufficiently good CIDEr score, using both the tags and embeddings improves the SPICE score.

We additionally observed that the captions generated using Conformer model with auxiliary features for 520 samples ($\sim 25\%$), among the combined 2090 validation and evaluation samples, had

Method	BLEU-1,2,3,4				ROUGE-L	METEOR	CIDEr	SPICE	SPIDEr
Baseline	0.389	0.136	0.055	0.015	0.262	0.074	0.084	0.033	0.054
Conformer	0.512	0.317	0.205	0.131	0.336	0.148	0.310	0.100	0.205
smaller enc-dec	0.500	0.311	0.203	0.129	0.336	0.144	0.299	0.099	0.199
smaller attention	0.490	0.307	0.199	0.127	0.332	0.143	0.310	0.096	0.203
+ larger-kernel	0.496	0.307	0.198	0.124	0.336	0.143	0.297	0.098	0.198
+ auxiliary features	**0.521**	**0.330**	**0.217**	**0.138**	**0.345**	**0.154**	**0.323**	**0.107**	**0.215**
+ dev-eval split	0.515	0.321	0.207	0.131	0.340	0.149	0.314	0.101	0.208
Ensemble	0.533	0.343	0.226	0.146	0.355	0.154	0.341	0.106	0.224

Table 1: Scores of evaluation metrics for the development-validation split.

Method	BLEU-1,2,3,4				ROUGE-L	METEOR	CIDEr	SPICE	SPIDEr
Baseline	0.378	0.119	0.050	0.017	0.078	0.263	0.075	0.028	0.051
Conformer	0.534	0.343	**0.233**	**0.158**	0.354	0.157	0.351	0.106	0.228
smaller enc-dec	0.524	0.331	0.219	0.144	0.356	0.153	0.329	0.103	0.216
smaller attention	0.506	0.320	0.212	0.140	0.349	0.152	0.337	0.102	0.219
+ larger-kernel	0.518	0.330	0.224	0.150	0.355	0.154	0.340	0.105	0.223
+ auxiliary features	0.536	0.341	0.225	0.146	0.357	0.160	0.346	0.108	0.227
+ dev-val split	**0.541**	**0.346**	0.231	0.152	**0.356**	**0.161**	**0.362**	**0.110**	**0.236**
Ensemble	0.546	0.356	0.243	0.165	0.369	0.163	0.381	0.110	0.246

Table 2: Scores of evaluation metrics for the development-evaluation split.

a SPICE score of zero. Note that SPICE score is measured over all 5 ground-truth captions and these zero scores can imply a complete semantic mismatch for a significant portion of testing samples. Among these zero score samples, we also observe that extracted AudioSet tags (auxiliary features) are sometimes match very closely with the caption words. Consider '`18 Little Group.wav`', an audio sample from validation split with a ground-truth caption of '*sea animals make strange blips, groans and other vocalizations*'. Our generated caption is '*a cat is meowing and making noises*'. However, the top-2 AudioSet tags extracted for this audio sample are '*Whale vocalization*' and '*Animal*'. A potential improvement from this analysis would be to increase the weight of projected auxiliary features when mixing them with the encoder output. To better integrate the extracted tags and embeddings, it is also possible to use an additional pretrained encoder from the PANNs model, and fine-tune the auxiliary features during training.

Method	CIDEr	SPICE	SPIDEr
Conformer + auxiliary input	0.323	**0.107**	**0.215**
- 527-tags	**0.325**	0.102	0.214
- 2048-embeddings	0.315	0.098	0.207
Conformer + auxiliary input	**0.346**	**0.109**	**0.227**
- 527-tags	**0.346**	0.104	0.225
- 2048-embeddings	0.342	0.106	0.224

Table 3: Evaluating contributions of PANNs tags and embeddings towards model performance on development-validation split (top) and development-evaluation split (bottom).

4.2.2. Evaluating Shallow Fusion with RNN-LM

Shallow fusion with a pretrained language model is equivalent to a model ensemble approach where the scores of the acoustic model and the language model are combined. Table 4 shows the performance improvement, especially of CIDEr scores, provided by an RNN-LM optimized on the word sequences in the training dataset.

Method	CIDEr	SPICE	SPIDEr
Conformer	0.310	0.100	0.205
- RNN-LM	0.300	0.098	0.199
Conformer	0.351	0.106	0.228
- RNN-LM	0.344	0.105	0.225

Table 4: Evaluating contribution of RNN-LM towards model performance on development-validation split (top) and development-evaluation split (bottom).

5. CONCLUSION

This work provides a detailed description and analysis of our submission to 2021 DCASE challenge Task 6: automated audio captioning. The proposed methodology employs existing state-of-the-art ASR techniques including Conformer-encoder, Transformer-decoder, data augmentation, Audioset tags & embeddings as auxiliary inputs and shallow fusion with a pretrained RNN language model. Our experiments qualify the ability of ASR techniques for effective captioning of audio samples by significantly outperforming the DCASE baseline system. Leveraging ASR techniques for audio captioning opens potential research directions towards developing an integrated framework for joint modeling of ASR and captioning tasks, and will be tackled as part of our future work.

6. ACKNOWLEDGMENT

This work was supported in part by Sony Corporation, JHU HLTCOE, and Bridges PSC (TG-CIS210014).

7. REFERENCES

[1] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 374–378.

[2] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[3] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *NAACL-HLT*, 2019.

[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[5] K. Chen, Y. Wu, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on transformer and pre-trained cnn," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020). Tokyo, Japan*, 2020, pp. 21–25.

[6] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre-and post-processings for audio captioning," *arXiv preprint arXiv:2009.11436*, 2020.

[7] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7008–7024.

[8] X. Xu, H. Dinkel, M. Wu, and K. Yu, "A crnn-gru based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 225–229.

[9] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[10] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proceedings of Interspeech*, 2018, pp. 2207–2211.

[11] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[12] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[13] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020.

[14] C. Narisetty, T. Hayashi, R. Ishizaki, S. Watanabe, and K. Takeda, "Leveraging state-of-the-art ASR techniques to audio captioning," DCASE2021 Challenge, Tech. Rep., 2021.

[15] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi, *et al.*, "Recent developments on espnet toolkit boosted by conformer," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5874–5878.

[16] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[17] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[18] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*. PMLR, 2017, pp. 933–941.

[19] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[20] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm," in *Proceeding of Interspeech*, 2017, pp. 949–953.

[21] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[22] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[25] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[26] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004.

[27] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the second workshop on statistical machine translation*, 2007, pp. 228–231.

[28] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[29] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *European conference on computer vision*. Springer, 2016, pp. 382–398.

[30] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 873–881.

USING UMAP TO INSPECT AUDIO DATA FOR UNSUPERVISED ANOMALY DETECTION UNDER DOMAIN-SHIFT CONDITIONS

Andres Fernandez, Mark D. Plumbley

Centre for Vision, Speech and Signal Processing (CVSSP)
University of Surrey, UK
{andres.fernandez, m.plumbley}@surrey.ac.uk

ABSTRACT

The goal of Unsupervised Anomaly Detection (UAD) is to detect anomalous signals under the condition that only non-anomalous (*normal*) data is available beforehand. In UAD under Domain-Shift Conditions (UAD-S), data is further exposed to contextual changes that are usually unknown beforehand. Motivated by the difficulties encountered in the UAD-S task presented at the 2021 edition of the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge[1], we visually inspect Uniform Manifold Approximations and Projections (UMAPs) for log-STFT, log-mel and pretrained Look, Listen and Learn (L3) representations of the DCASE UAD-S dataset. In our exploratory investigation, we look for two qualities, *Separability (SEP)* and *Discriminative Support (DSUP)*, and formulate several hypotheses that could facilitate diagnosis and developement of further representation and detection approaches. Particularly, we hypothesize that input length and pretraining may regulate a relevant tradeoff between SEP and DSUP. Our code as well as the resulting UMAPs and plots are publicly available[2].

Index Terms— DCASE2021, Unsupervised Anomaly Detection, Domain Shift, UMAP, Interpretability

1. INTRODUCTION

The goal of Unsupervised Anomaly Detection (UAD) is to detect anomalous instances under the condition that only non-anomalous (i.e. *normal*) instances are available beforehand. This has relevance in monitoring applications where anomalous data is hard to collect whereas normal data is abundant. Unsupervised Anomaly Detection under Domain-Shift Conditions (UAD-S) presents an extra challenge: both normal and anomalous data can be exposed to *domain shifts*, i.e. changes in the environment that cause an impact in the data and are usually unknown beforehand. This can result in false negatives, if the detector is not sensitive enough, or false positives, if the detector does not tolerate or adapt to domain shifts.
In the audio domain, UAD has attracted attention as a promising Predictive Maintenance (PdM)[1] solution for Industrial Sound Analysis (ISA)[2]: Sound monitoring is non-invasive, is robust to occlusions, can be carried out during production, and anomalous sounds can signal issues long before critical faults occur. UAD-S is a natural extension to UAD, since even in controlled environments, new non-anomalous sources of sound can arise (e.g. due to maintenance work or upgrades). Given that it is difficult or undesirable to completely isolate the analyzed sound source from its environment, PdM-ISA solutions must be able to detect slight deviations (including short-duration events like clicks) while embracing stronger environmental changes.
The 2020 and 2021 editions of Detection and Classification of Acoustic Scenes and Events (DCASE) have incorporated UAD (2020, task 2) and UAD-S (2021, task 2) challenges. In both editions, a broad variety of approaches has been explored, but the results achieved in 2021 were significantly lower than in 2020 in terms of numeric performance. This may indicate a higher complexity of the 2021 task, independently of the choice of model and training scheme. In order to gain further insights, we propose to inspect the data distribution itself. Specifically, our proposed contributions are:

- We showcase a method for of UAD-S data exploration via visual inspection of Uniform Manifold Approximations and Projections (UMAPs) and assessment of 2 beneficial qualities: **Separability (SEP)** and **Discriminative Support (DSUP)**.
- We apply the proposed analysis procedure to the DCASE 2021 dataset, revealing insights on its macro- and microstructure.
- Based on the analysis and literature, we formulate a series of verifiable hypotheses that we believe can facilitate diagnosis and developement of further approaches.

Section 2 reviews UAD in DCASE. Section 3 describes our methodology. Section 4 describes our experiments. Section 5 presents and discusses some results. Section 6 concludes and proposes future work.

2. UAD IN DCASE

The DCASE 2020 dataset was a result of combining two recently curated datasets (ToyAdmos[3] and MIMII[4]), each featuring 10-second audio segments from different well-functioning *devices* (toy car, valve, fan, etc). Each segment was mixed with different background sounds to simulate real environments. The devices were then intentionally damaged/disrupted to provide anomalous data, which was only available for validation. The proposed models had to provide a real-valued anomaly score for each validation audio segment, and their performance was evaluated by ranking the Area Under ROC Curve (AUC) and Area Under Partial ROC Curve (pAUC) obtained across different devices[5].
The 10 best performing submissions in 2020 applied Deep Learning (DL), treating the developement dataset as training data. Some used different forms of data augmentation and additions from ex-

[1]DCASE website: http://dcase.community
[2]Online Resources:
Code: https://github.com/andres-fr/dcase2021_umaps
Webpage: https://ai4s.surrey.ac.uk/2021/dcase_uads

ternal datasets such as AudioSet[6] and Fraunhofer's IDMT-ISA-EE dataset[2]. For inference, most submissions directly applied the trained DL models, often via multi-task ensembles. The most popular alternative was to apply K-Nearest Neighbors (KNN) to learned embeddings of the training set[7, 8]. Most best performing models incorporated the information of the specific device upon training and evaluation. An exception was [9], which treated the data for all devices jointly. In general, a broad variety of models and training schemes achieved scores over 90%.

The 2021 UAD-S edition also combined two datasets (MIMII DUE[10] and ToyAdmos2[11]), extended in several aspects. Particularly, for each device, the 2021 dataset includes 7 *devices*, 6 *sections* and 2 *domains*, totalling 84 splits. Each device has 6 sections, which are balanced partitions of the data for evaluation purposes. Each section presents 2 domains: *source* and *target*, which differ in aspects like operating speed, machine load and environmental noise. As in 2020, the training data does not contain any anomalous sounds. The training data is also highly imbalanced: in all sections, only ∼0.3% of the training samples are on the *target* domain. Test data is balanced in terms of devices, splits and domains. The evaluation procedure is similar to 2020, but this time an overall *score* is given as the harmonic mean across all AUC and pAUC scores[12].

Out of 27 submissions for 2021, the autoencoder (AE) baseline ranked 21st with a *score* of ∼56.4% on the evaluation set. The 2021 winners[13] (∼66.8%) propose a particularly heterogeneous ensemble, combining different "*complementary*" representations, objectives and models, rather than "*relying on well-known domain adaptation techniques*". A related concept is the contrastive loss applied by [14], (9th place, ∼61%). Second place was achieved by [15] (∼65%) with a simpler setup based on applying non-parametric inference methods (Local Outlier Factor (LOF) and KNN) to trained embeddings. We note that, while it was observed that Representation Learning (RepL)-based methods generally underperform reconstruction-based ones for UAD[16], this does not seem to be the tendency here: reconstruction objectives are barely present in the top ranks, possibly due to sensitivity to domain shifts, and the emphasis is on representations, e.g. the importance of spectrogram hyperparameters noted by [13] and the implications and effectiveness of different embeddings analyzed by [17] (3rd place, ∼64.2%) which propose to use AdaCos[18]. An exception is [19] (4th place, ∼63.75%), which did propose a reconstruction-based method that compensates domain shift conditions.

Like in 2020, a variety of DL-related approaches were adopted, but the scores were substantially lower in the 2021 edition. Keeping in mind the small differences in the evaluation procedure, we argue that the emphasis on RepL-based methods, the relative success of non-parametric inference and the difficulty directly addressing domain shifts via well-known techniques point at the complexity of the task and the relevance of an adequate data representation, independently of the choice of model and training scheme. Therefore, in this exploratory work we propose to inspect the UAD-S data distribution itself. The goal is to gain further insights in order to facilitate diagnosis and developement of further approaches.

3. INSPECTING REPRESENTATIONS WITH UMAP

Generally, direct exploration of high-dimensional data like that encountered in the discussed approaches is difficult. Fortunately, when data is organized in lower-dimensional structures it can be possible to retain some of its structure while projecting the data onto as few as 2 dimensions, allowing for informative visual inspection.

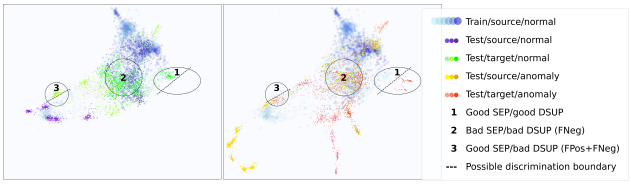


Figure 1: Excerpt from the device UMAP plot for *pump* with annotations in black. For example, region **1** presents good SEP and DSUP, since there is a simple boundary that clearly separates anomalies from normals and training data, and train/normal supports test/normal. *FPos* stands for false positives, and *FNeg* for false negatives. Each dot corresponds to 5 stacked log-mel frames. Color shades correspond to dataset *sections*. Training data is shown on both sides. Zoom to ∼1000% for detail.

UMAP[20] is a non-linear projection technique that has been shown to surpass alternatives like Principal Component Analysis (PCA) and t-SNE[21] in terms of speed, stability against reparametrizations and "*meaningfulness*" when applied on biological data[22]. Nevertheless, dimensionality reduction usually entails information loss, and artifacts arise: dense clusters may appear spread out and well-separated structures may collide when projected. For this reason, we restrict ourselves to the **assumption** that *if two regions appear separable on the UMAP projection, they are also separable on the original representation*. Crucially, *the opposite is not necessarily true*: two regions appearing mixed could be due to a projection artifact. With this in mind, we focus on two UAD-S properties:

- **Separability (SEP)**: In a projection with "good" SEP, a simple boundary can be drawn between anomalous and normal data with small error.
- **Discriminative Support (DSUP)**: If the training data provides set support for all normal data, and is separable from anomalous data, that set support can be directly used to discriminate anomalies. We consider that to be "good" DSUP.

Thus, by our assumption, if a given UMAP projection presents good SEP and DSUP, we infer that the corresponding high-dimensional representation has a simple boundary to separate normal from anomalous data (i.e. good SEP), and can be expressed using proximity to the training data (good DSUP). We argue that this is beneficial for the UAD-S task. Figure 1 illustrates this.

The concept of SEP could be quantified at a local level via e.g. Support Vector Machines (SVMs)[23]: given a set of data vectors $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, labeled with -1 or 1 as $(y_1, \ldots, y_N)$, the SVM objective is to find the hyperplane parametrized by β that creates the biggest margin between both classes, allowing for some error ε, via the following objective[24, ch. 12]:

$$\min \|\beta\| \quad \text{s.t.} \begin{cases} y_i(x_i^T\beta + \beta_o) \geq 1 - \xi_i \\ \xi_i \geq 0 \\ \sum_i \xi_i \leq \varepsilon \end{cases} \qquad \forall i \in \{1, \ldots, N\} \tag{1}$$

Good SEP would be then achieved when a simple boundary separates normal and anomalous data with low ε. DSUP could be similarly quantified by comparing training and anomalous data, provided training data supports all normal data. But here we propose a complementary approach: to **qualitatively** assess SEP and DSUP via visual inspection of UMAPs. To that end, we render a series of dual plots, showing **anomalous test data on the right and normal test data on the left**.

4. UMAP FOR DCASE 2021

For our data sources, we merged the *Development* and *Additional Training* datasets[12] from DCASE 2021, task 2. To illustrate the role of external datasets, we also incorporated the 10-second cut variant of Fraunhofer's IDMT-ISA-EE dataset[2], and a custom subset of AudioSet consisting of 10-second segments from ∼40k unique videos. All audio files were converted to mono 16kHz, and $(-1, 1)$ normalization was applied. Based on high-performing systems from 2020, we computed amplitude spectrograms via the square modulus of Short-Term Fourier Transforms (STFTs) with 1024 samples per window and 50% overlap for all datasets. We then converted amplitudes to dB, yielding the log-STFTs. From the STFT spectrograms, we also computed 128-bin melgrams[25] and converted them to dB, yielding the log-mels. We ended up with ∼300k frames per *source* split and 927 frames per *target* split, totaling ∼13 million for *source* and 39k for *target*. Our AudioSet subset had then ∼12.2 million frames, and Fraunhofer ∼223k. We used `librosa`[26] for the above audio computations. We also computed 512-dimensional Look, Listen and Learn (L3) embeddings with a hop size of 0.1 seconds using `openl3`[27]. L3 embeddings encode longer-term relationships and this results in less frames (e.g. ∼3.5 million for our AudioSet). STFTs from environmental AudioSet videos were used for L3 audio training.

To encode temporal relationships, we stacked consecutive frames. In this work we explored 3 stack sizes: 1, 5 and 10. We computed a set of 2D UMAP projections for each of the 3 computed representations and 3 stack sizes. To get resolution at different scales, we computed one UMAP per device plus a global UMAP, totalling 72 UMAPs. Due to hardware limitations, the full datasets couldn't be processed and random samples were taken: For the per-device projections with stack size 1 and 5, we took 20k random samples for each validation split, and a maximum of 10k for every other split (recall that *target* training splits have just 927 frames). For the global projections with stack size 1 and 5, we took 2k samples per validation split, a maximum of 1k per training split, 50k for AudioSet and 50k for Fraunhofer. We also computed the stack size 10 UMAPs with no external data. For any given representation and stack size, we developed 3 kinds of scatter plots to enable different levels of detail: Global plots like Figure 2 are based on the global UMAPs and show the full dataset, coloring the different devices. Device plots like Figure 3 are based on the per-device UMAPs and color the different sections and domains. Section plots like Figure 5 are based on the per-device UMAPs, but they show a single section and color the specific audio files.

5. DISCUSSION

In general, SEP patterns across different devices and sections could be observed, but regions with both good SEP and DSUP were very hard to find, the best example we found has been already presented in Figure 1. Distinctively anomalous patterns are also scarce and do not appear to follow any obvious repeating patterns. We also observe that data from the *target* domain generally overlaps with the *source* domain.

The difference between ToyAdmos2 and MIMII DUE datasets can be seen at multiple scales: the ToyCar and ToyTrain clusters are clearly distinguishable from all other devices (see Figure 2), and the internal structure for the ToyAdmos2 devices is also apparently simpler than for MIMII DUE devices (compare Figures 3 and 4).

Another distinctive feature is the "horn" shape formed by the AudioSet data samples (e.g. Figures 5 and 4). Since energy spectrograms are non-negative, they are confined to the first quadrant, which is a cone with a vertex on the zero-energy point. By checking the energies, we have observed that the lowest-energy samples are highly concentrated on all observed "horns" (a few outliers get projected elsewhere). This indicates that, accounting for the logarithmic conversion to dB, the representations conserve the conic geometry and the observed "horn" shape likely corresponds to the tip of the cone, giving a sense of origin that can aid interpretation.

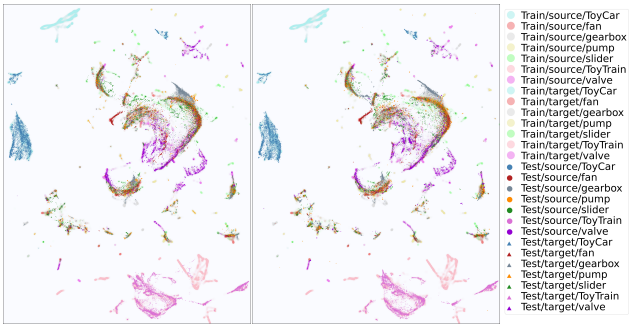


Figure 2: Global UMAP plot, sampled from the DCASE 2021 dataset. Each dot corresponds to 10 stacked log-STFT frames. Smaller dots correspond to anomalies on the right, and normal data on the left. Training data is shown on both sides. Zoom to ∼1000% for detail.

Interestingly, in all L3 device plots for *fan*, the training data appears almost completely separated from the test data (see e.g. Figure 4, the "shadows" have almost no test data on them). This means that a single L3 frame is enough to distinguish the *fan* training data from the test data fairly well. This should not be confused with a different phenomenon: some "shadow" clusters lack any overlying test data (e.g. the dark blue ones in Figure 3), but that is likely because those regions correspond to evaluation splits for which the challenge organizers did not release the test data. This is likely the case if the behavior is consistent across all representations.

Another particularity is that the AudioSet cluster appears to be smaller on the L3 representations, and the non-AudioSet data appears more scattered. This may be due to the fact that the L3 embeddings were trained on AudioSet and achieve a more compact representation there.

In the following we highlight several modelling **hypotheses** based on the above observations and the literature. We refer to our online resources for extensive results and code.

1. **Mixing ToyAdmos2 and MIMII DUE data may hinder performance**: Trivially distinguishable categories may lead to inefficient boundaries for anomaly discrimination. This was already proposed in [28].
2. **Temporal context and pretraining regulate a tradeoff between SEP and DSUP**: Generally, we observe that longer stack sizes provide better SEP. This makes sense because given enough length all audio files can be uniquely identified. But we also observed that this tends to scatter data apart and worsen DSUP. With pretrained embeddings, the observed tendency of concentrating the pretrained domain and scattering the rest may also entail a similar tradeoff. An ensemble with different tradeoff configurations may be beneficial. The complementary and contrastive approaches discussed in Section 2 may implicitly leverage this fact.

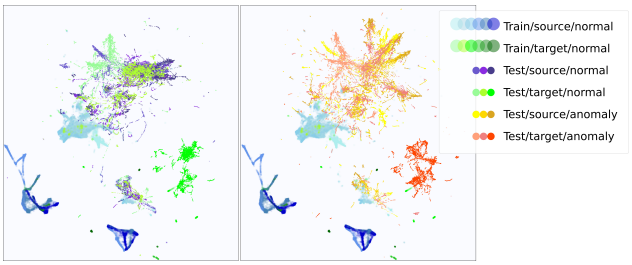


Figure 3: Device UMAP plot for *ToyCar*. Each dot corresponds to a single L3 frame. Color shades correspond to dataset *sections*. Training data is shown on both sides. Zoom to ∼1000% for detail.

3. **Normalization is a dominating factor**: If we interpret the data in Figure 5 as a cone with the vertex at the tip of the AudioSet "horn", renormalizing a frame would shift that frame roughly along the cone axis, which can greatly impact SEP and DSUP. The importance of proper normalization is supported by top-performing approaches like [9] and [17].
4. **Incorporating domain-related priors may help performance**: Bad DSUP only means that the the training data support can't be directly used for discrimination, but other kinds of prior knowledge still could be used to leverage the existing SEP. The 2021 dataset provides *domain*-related labels describing the domain shifts in the training data that could be used as priors. To the best of our knowledge, none of the participants made use of it, and could be a beneficial addition.

Lastly, we are aware of several methodological shortcomings:

1. We are only observing a subsample of the data, so extreme outliers are likely to be missed. Taking them into account may be crucial for successful analysis and detection.
2. As discussed in Section 3, data projections can only be used to confirm SEP and DSUP, not to discard them.
3. Qualitative, visual inspection may also be subject to perceptual biases, e.g. by color strength or shape consistency. Furthermore, plotting anomalous and normal data on different sides hinders the visual detection of slight differences.
4. Encoding temporal relations by stacking successive frames can lead to suboptimal representations due to e.g. conditional relations among frames, normalization and weighting issues.

Points 1 and 2 can be tackled by applying quantitative methods to the non-projected data, since the artifacts and size restrictions are imposed by the UMAP step. To overcome any issues related to high data volume and dimensionality, LOF and/or KNN-based methods like the ones used in [29, 7, 15] can be explored. Interactive exploration of the plots can help identifying small differences and overcoming perceptual biases. Representations that encode broader temporal context and other kinds of context can be explored to replace the frame stacks. Particularly, giving more weight to anomalous frames (or even ignoring very common frames) may help improving SEP and DSUP.

6. CONCLUSION AND FUTURE WORK

In this paper we performed an analysis of fixed and learned UAD-S data representations, based on the visual inspection of UMAPs

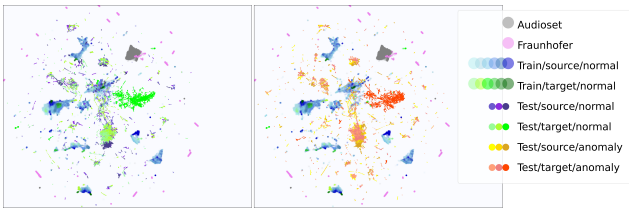


Figure 4: Device UMAP plot for *fan*. Each dot corresponds to a single L3 frame. Color shades correspond to dataset *sections*. Training data is shown on both sides. Zoom to ∼1000% for detail.

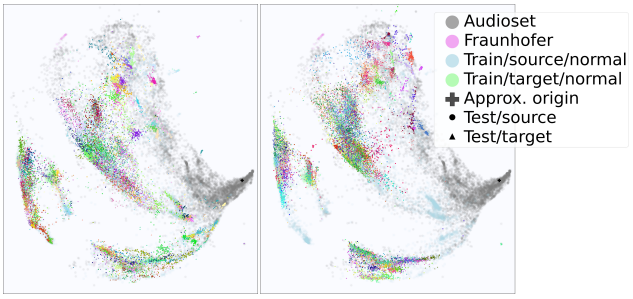


Figure 5: UMAP plot for *valve*, section 0. Each dot corresponds to a single log-mel frame. Smaller dots correspond anomalies on the right, and normal data on the left, and are colored by audio filename. After ignoring the 500 lowest-energy frames, the cross signals the average position of the following 100 ones. Zoom to ∼1000% for detail.

and assessment of *separability* (SEP) and *discriminative support* (DSUP). In line with the difficulties encountered at the DCASE challenge, we did not find consistently good SEP and DSUP in any of the observed representations. The representations helped to expose potential issues in connection with the literature, and ways to address them. Despite the discussed methodological shortcomings, we defend that visual UMAP inspection can complement well other quantitative forms of analysis, and we hope that the software we provide can become a useful tool in the context of UAD-S.
The analysis could be enhanced with interactive plots providing sonification (to better understand the topology by hearing it), and highlighting corresponding datapoints across different representations. Analysis of further representations and techniques like X-vectors and the Teager-Kaiser energy operator[13] may also be of interest, as well as the impact of different embedding objectives on SEP and DSUP. Another possible extension could be to visualize the actual predictions of a system, extending this analyisis framework to supervised scenarios.

7. ACKNOWLEDGMENTS

The authors would like to thank Helen Cooper for the support throughout the research process. This work was supported by grant EP/T019751/1 from the Engineering and Physical Sciences Research Council (EPSRC) and made use of time on Tier 2 HPC facility JADE2, funded by EPSRC grant EP/T022205/1.

8. REFERENCES

[1] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, 2019.

[2] S. Grollmisch, J. Abeßer, J. Liebetrau, and H. Lukashevich, "Sounding industry: Challenges and datasets for industrial sound analysis," in *EUSIPCO 2019*.

[3] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "ToyADMOS: A dataset of miniature-machine operating sounds for Anomalous Sound Detection," in *WASPAA 2019*.

[4] H. Purohit, R. Tanabe, T. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, "MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," in *DCASE 2019 Proceedings*.

[5] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *DCASE2020 Proceedings*, July 2020.

[6] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "AudioSet: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*.

[7] Q. Zhou, "ArcFace based sound MobileNets for DCASE2020 task 2," DCASE2020 Challenge, Tech. Rep., July 2020.

[8] Y. Sakamoto and N. Miyamoto, "Anomaly calculation for each components of sound data and its integration for DCASE 2020 challenge task2," DCASE2020 Challenge, Tech. Rep., July 2020.

[9] J. Lopez, L. Hong, P. Lopez-Meyer, L. Nachman, G. Stemmer, and J. Huang, "A speaker recognition approach to anomaly detection," DCASE2020 Challenge, Tech. Rep., July 2020.

[10] R. Tanabe, H. Purohit, K. Dohi, T. Endo, Y. Nikaido, T. Nakamura, and Y. Kawaguchi, "MIMII DUE: Sound dataset for malfunctioning industrial machine investigation and inspection with domain shifts due to changes in operational and environmental conditions," *arXiv:2006.05822, 1–4*, 2021.

[11] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, "ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions," *arXiv:2106.02369*, 2021.

[12] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and discussion on DCASE 2021 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring under domain shifted conditions," *arXiv:2106.04492, 1–5*, 2021.

[13] J. Lopez, G. Stemmer, and P. Lopez-Meyer, "Ensemble of complementary anomaly detectors under domain shifted conditions," DCASE2021 Challenge, Tech. Rep., July 2021.

[14] X. Cai, H. Dinkel, Z. Yan, Y. Wang, J. Zhang, and Y. Wang, "The small rice camera ready submission to the DCASE2021," DCASE2021 Challenge, Tech. Rep., July 2021.

[15] K. Morita, T. Yano, and K. Tran, "Anomalous sound detection using CNN-based features by self supervised learning," DCASE2021 Challenge, Tech. Rep., July 2021.

[16] S. Li, K. Tian, and R. Wang, "Unsupervised heart abnormality detection based on phonocardiogram analysis with beta variational auto-encoders," in *ICASSP 2021*.

[17] K. Wilkinghoff, "Utilizing sub-cluster AdaCos for anomalous sound detection under domain shifted conditions," DCASE2021 Challenge, Tech. Rep., July 2021.

[18] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[19] I. Kuroyanagi, T. Hayashi, Y. Adachi, T. Yoshimura, K. Takeda, and T. Toda, "Anomalous sound detection with ensemble of autoencoder and binary classification approaches," DCASE2021 Challenge, Tech. Rep., July 2021.

[20] L. McInnes and J. Healy, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv:1802.03426*, 2018.

[21] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[22] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. Kwok, L. G. Ng, F. Ginhoux, and E. Newell, "Dimensionality reduction for visualizing single-cell data using UMAP," *Nature Biotechnology*, vol. 37, 01 2019.

[23] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifier," *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, vol. 5, 08 1996.

[24] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[25] S. S. Stevens and J. Volkmann, "The relation of pitch to frequency: A revised scale," *American Journal of Psychology*, vol. 53, p. 329, 1940.

[26] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, vol. 8, 2015.

[27] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *ICASSP 2019*.

[28] P. Primus, "Reframing unsupervised machine condition monitoring as a supervised classification task with outlier-exposed classifiers," DCASE2020 Challenge, Tech. Rep., July 2020.

[29] K. Durkota, L. Michael, L. Martin, and T. Jan, "Neuron-net: Siamese network for anomaly detection," DCASE2020 Challenge, Tech. Rep., July 2020.

AUTOMATED AUDIO CAPTIONING BY FINE-TUNING BART WITH AUDIOSET TAGS

Félix Gontier, Romain Serizel, Christophe Cerisara

Université de Lorraine, CNRS, Inria, Loria, F-54000, France.
felix.gontier@inria.fr, {romain.serizel,christophe.cerisara}@loria.fr

ABSTRACT

Automated audio captioning is the multimodal task of describing environmental audio recordings with fluent natural language. Most current methods utilize pre-trained analysis models to extract relevant semantic content from the audio input. However, prior information on language modeling is rarely introduced, and corresponding architectures are limited in capacity due to data scarcity. In this paper, we present a method leveraging the linguistic information contained in BART, a large-scale conditional language model with general purpose pre-training. The caption generation is conditioned on sequences of textual AudioSet tags. This input is enriched with temporally aligned audio embeddings that allows the model to improve the sound event recognition. The full BART architecture is fine-tuned with few additional parameters. Experimental results demonstrate that, beyond the scaling properties of the architecture, language-only pre-training improves the text quality in the multimodal setting of audio captioning. The best model achieves state-of-the-art performance on AudioCaps with 46.5 SPIDEr.

Index Terms— Audio captioning, language models, transfer learning, BART, audio tagging

1. INTRODUCTION

The task of automated audio captioning [1] aims at improving the description of environmental sounds through the production of textual descriptions of input audio. This field of research has seen recent growth in interest within the audio community, with a recurring dedicated task introduced in 2020 to the DCASE challenge[1].

Audio captioning methods typically rely on sequence-to-sequence approaches, that encode audio features and produce sentences through a separate decoder [2]. With increasing focus on the format and vocabulary of captions [3, 4], recent advances have been achieved by encoding textual inputs in addition to audio representations. In particular, keyword prediction [5, 6] or similar captions retrieval [7] have been investigated as supplementary guidance material for captioning systems. Any such supplementary information must be inferred directly from the audio signal.

In terms of audio features, pre-trained embeddings such as VGGish [8] are commonly utilized. These embeddings are highly informative compared to other representations (eg. Mel spectrograms), which reduces the model capacity necessary to extract relevant semantic content as a result. In previous studies, however, text-based conditioning inputs are produced by a dedicated module trained on the captioning dataset. The small amounts of available data and the diversity of sound objects heavily limit the capabilities of such architectures. This often leads to poor accuracy in guidance inputs, and thus a lower semantic correctness of the resulting captions. For instance, Koizumi et al. [7] find that a model conditioned on ground truth similar captions in the dataset reaches near-human performances, whereas learned audio-based similar caption retrieval leads to significantly worse results.

Beyond textual input extraction, the language generation module in captioning systems is often trained from scratch. Thus, the model must learn to reproduce a fluent language structure with diverse vocabulary in addition to conveying semantic information from the input audio. Concurrently, many pre-trained models have been proposed in the natural language processing (NLP) community that efficiently model the syntax of natural language for representation learning [9] or generation [10]. Nevertheless, directly applying language models to multi-modal tasks such as captioning is not straightforward. Koizumi et al. [7] integrated a frozen GPT-2 [10] instance as the main language modeling part in their captioning system, and obtained results on par with the previous state of the art with fewer trainable parameters.

In this paper, we investigate scaling audio captioning architectures to the capacity of large-scale language models by utilizing audio and language pre-training. To do so, we present a novel method that adapts a transformer encoder-decoder with the BART general purpose pre-training [11] to produce captions by attending to both audio and text embeddings. This setting is illustrated in Figure 1. Thus, contrary to Koizumi et al. [7] no additional module combining audio and text information is learned from scratch. Furthermore, instead of learning guidance textual guidance inputs on the captioning dataset, we condition generation on AudioSet tags [12] obtained from a pre-trained model, YAMNet [13].

Specifically, the contributions of the present work are as follow:

- We propose a multi-modal conditioning scheme based on aligned temporal sequences of text and audio embeddings obtained from pre-trained models. The combination method relies on very few additional trainable parameters.
- We demonstrate that fine-tuning BART on these inputs results in high audio captioning performance, outperforming previous state-of-the-art systems.
- Through complementary experiments, we show the scaling properties of the BART architecture, as well as the potential of pre-training in tackling smaller captioning datasets.

To encourage the use of the proposed method in future work, the code for all presented experiments is made available[2].

This work was funded under the ANR project LEAUDS (Grant No. ANR-18-CE23-0020).

[1] https://dcase.community

[2] https://github.com/felixgontier/dcase2021aac

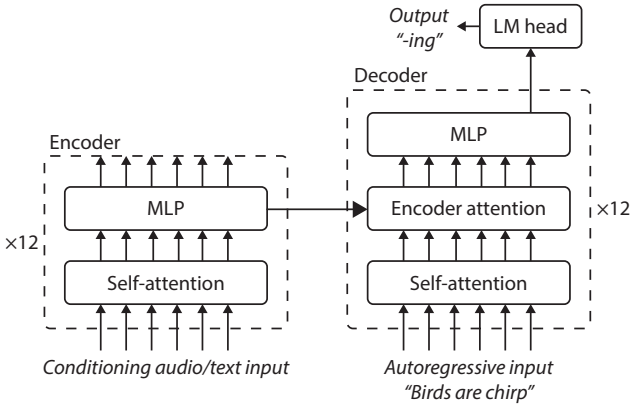


Figure 1: BART transformer architecture (residual connections and layer normalization omitted for clarity) and proposed setting in the captioning application.

2. METHODS

2.1. The BART model

The architecture associated with BART is a standard sequence-to-sequence transformer [14]. It is composed of a bi-directional encoder and an autoregressive decoder as illustrated in Figure 1. Encoder transformer blocks contain a multi-head self-attention layer followed by a multi-layer perceptron (MLP), as well as residual connections and layer normalization after each transformation. Decoder layers are further conditioned on the encoder output through an additional multi-head cross-attention layer after self-attention. Lastly, a dense layer outputs logits across all tokens in the data vocabulary. The main BART model comprises 12 layers in both the encoder and decoder with and internal dimension of 1024 in all hidden layers. The language tokenizer uses byte-pair encoding with a vocabulary of 50265 tokens. As a result, the architecture totals about 400 million parameters.

2.2. Text conditioning

In accordance with previous studies, we condition captioning with inputs describing the semantic content in text form. To do so, we propose to infer AudioSet tags from the audio input using YAMNet [13]. The YAMNet model achieves high tagging accuracy, and operates on 1 s audio frames. Contrary to other textual conditioning (e.g. keyword prediction), this results in a temporal sequence of identified sound objects in audio extracts. Such sequential detail is often found in ground truth captions, although at a coarser scale.

To obtain the conditioning input, YAMNet is applied to 1 s non-overlapping frames $\mathbf{x}_i$ of the audio input. This process is shown in Figure 2. Instead of selecting the AudioSet tag as the maximum of YAMNet logits, we sample tags from the output distribution at each iteration. The empirical motivation of this design is to increase the robustness of the model to YAMNet prediction errors, by randomly introducing incorrect yet plausible tags to the conditioning input. AudioSet tags are then utilized in their textual form (eg. *Chirp, tweet*). After application of the BART tokenizer, each tag typically results in a sub-sequence of one to six tokens. Conditioning sequences are produced by concatenating all sub-sequences with separator tokens. In our experiments, the choice of token (resp.

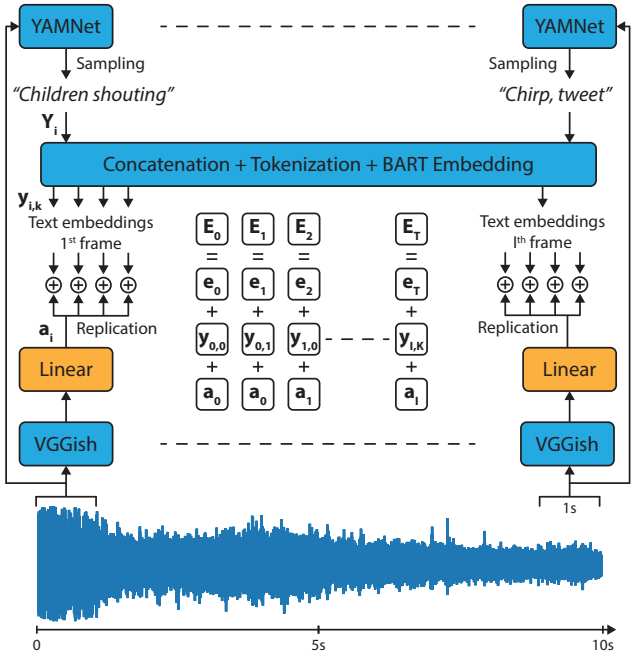


Figure 2: Conditioning input construction from pre-trained audio embeddings and textual tags. Blue and orange shading indicate frozen and learned modules respectively. Audio, textual, and positional embeddings (resp. $\mathbf{a_i}$, $\mathbf{y_{i,k}}$, and $\mathbf{e_t}$) are added to produce the input $\mathbf{E_t}$.

"."", "and", "then", "<mask>") did not have significant impact on the initial loss, convergence rate, or final captioning performance. Thus, all discussed models are trained with '. ' as the separator token. The pre-trained BART embedding layer then encodes tokens into real-valued vectors $\mathbf{y}_{i,k}$ of dimension 1024, where k denotes the token position within the i^{th} tag sub-sequence. Lastly, positional embeddings $\mathbf{e}_t$ are further added to $\mathbf{y}_{i,k}$ to disambiguate the sequential order of inputs in the model, where t is the token position in the model input sequence independently from the audio frame i.

2.3. Audio conditioning

Although conditioning the generation on text only is a very close setting to the pre-training objective of BART, YAMNet predictions may be erroneous for part of the audio extract. In order to help the model select correct semantic content from given tags, audio embeddings are added to the encoder input. Embeddings from the YAMNet model likely contain the information as the conditioning tags. Thus, we explore deep embeddings from the penultimate layer of two other tagging models: VGGish [8] and PANNs [15], specifically the *Wavegram-Logmel-CNN* variant.

VGGish is able to provide 128-dimensional embedding vectors $\mathbf{a}_i$ for 1 s audio frames. This matches the granularity of YAMNet predictions, and allows for the alignment of textual and audio sequences to condition the caption generation. Figure 2 illustrates the corresponding conditioning process. VGGish embeddings are replicated over all tokens of the corresponding YAMNet tag. Following Huang et al. [16], the embeddings are directly added to their text and positional equivalents (resp. $\mathbf{y}_{i,k}$ and $\mathbf{e}_t$). As the audio em-

bedding size is different from the internal dimension of BART, we introduce a trainable dense layer to perform the adaptation.

Alternatively, the PANNs model infers a single embedding vector of dimension 2048 for 10 s of audio. Compared to VGGish, the lower temporal detail in PANNs embeddings is compensated by greater semantic content, which translates to higher performance on AudioSet tagging. Conditioning on PANNs is also straightforward: embeddings are replicated over encoder input timesteps corresponding to 10 consecutive tags, mapped to 1024-dimensional vectors by a dense layer, then added to text and positional embeddings.

3. EXPERIMENTAL SETUP

3.1. Dataset

All the experiments are conducted on the AudioCaps dataset [17]. AudioCaps comprises training, validation, and evaluation splits of about 49000, 485 and 955 audio extracts. The dataset is a subset of AudioSet [12], thus most audio examples have a duration of 10 s. Training examples are associated with a single annotated caption, whereas validation and test splits contain 5 captions per audio file.

3.2. Evaluation metrics

We evaluate the quality of the generated captions on standard captioning metrics. We report BLEU-1 to BLEU-4, METEOR, ROUGE-L, and CIDEr [18], which are all based on n-gram matching. In addition, SPICE [19] is computed as an evaluation of the semantic quality of the generated captions. The overall performance is given by SPIDEr [20], the average of CIDEr and SPICE, which is the main metric of the DCASE challenge task 6 on captioning.

3.3. Baselines

We compare our approach against two baselines in the literature. First, the `TopDown-AlignedAtt` model in the AudioCaps dataset paper [17] achieves the best reported performance according to the SPIDEr metric. Secondly, the system presented by Koizumi et al. [7] is, to our knowledge, the first to include a pre-trained language model (frozen GPT-2) in an audio captioning framework. Lastly, the current state-of-the-art system on AudioCaps is described in [6].

3.4. Training procedure

Model parameters are trained to minimize the categorical cross-entropy loss over the 50265 classes in the BART tokenizer. Optimization is performed using the AdamW [21] optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and with a learning rate of 10^{-5}. In our experiments, we observed stable convergence without major overfitting. All models are trained for 4 epochs, or about 24000 iterations with a batch size of 8 examples. Validation is carried out after every 1000 iterations, and we retain the model checkpoint with the lowest validation loss.

3.5. Caption generation procedure

In order to preserve deterministic inference at evaluation, conditioning tags are selected as the maximum value over YAMNet logits as opposed to the sampling scheme applied during training. Captions are generated autoregressively via beam search, with a beam size of 4 and no constraint or penalization on the total caption length.

4. EXPERIMENTS

4.1. Conditioning evaluation

We investigate combinations of audio and textual encoder inputs, including text-only and audio-only captioning guidance. The model for each conditioning setting is trained 3 times with different random seeds. Table 1 details the main experimental results, with means and standard deviations of metrics reported over the 3 instances. The proposed approach is compared to baselines as well as human performance, which refers to the cross-validation of reference captions in the evaluation set [17]. This anchor reflects discrepancies in content and syntax among ground truth captions, and constitutes a reasonable upper bound on caption quality.

First, the model conditioned on PANNs embeddings, which only contain one vector for AudioCaps examples, fails to generate well-structured captions. VGGish embeddings perform significantly better, hinting that providing the model with information on the sequence of sound events is critical in audio-only conditioning designs. However, the variance in performance between training instances is very high compared to other settings.

Conditioning solely on YAMNet tags further improves both the fluency and faithfulness of captions. Tags directly provide vocabulary guidance to the model, whereas relevant terms must be inferred from audio embeddings. In addition, the model only operates on language in this case, thus the task setting is close to that of the BART pre-training scheme.

The relative increase in performance for settings combining text and audio conditioning suggests that the information in both inputs is complementary. Contrary to audio-only experiments, PANNs performs better than VGGish when paired with YAMNet tags. Because sequential detail is already contained in the input tokens, the temporal granularity of audio embeddings is less important than their semantic content. Empirical analysis reveals a higher - although weak - correlation between YAMNet tagging accuracy and SPICE scores in the text-only model compared to that combining tags and PANNs embeddings. This behavior may indicate that audio embeddings mitigate the appearance rate of incorrectly identified sound objects in produced captions.

4.2. System performance

The proposed approach achieves similar results to Eren et al. [6] on reported metrics, and outperforms other baselines. It is on par or better than human performance according to BLEU-1, BLEU-2, BLEU-3, and ROUGE-L. However, these metrics only evaluate matching n-grams, and do not correlate well with human evaluations of quality [22, 23]. On more advanced metrics for syntactic fluency and semantic correctness, respectively CIDEr and SPICE, the best model is below human performance by a large margin. Still, the low remaining gap in terms of SPICE score confirms that the high accuracy of YAMNet source recognition is well conveyed to output captions.

In the following subsections, we present complementary experiments on the properties of BART pre-training for audio captioning. All experiments retain the best conditioning setting of YAMNet tags combined with PANNs embeddings.

4.3. Interest of language pre-training

Because of the multi-modal nature of inputs in the captioning task, it is relevant to assess whether the system performance can be linked

Table 1: Evaluation of the proposed approach on AudioCaps. The displayed scores are means and standard deviations over three instances. The highest value for each metric is shown in bold.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE	SPIDEr
TopDown-AlignedAtt [17]	61.4	44.6	31.7	21.9	20.3	45.0	59.3	14.4	36.9
Koizumi et al. [7]	63.8	45.8	31.8	20.4	19.9	43.4	50.3	13.9	32.1
Eren et al. [6]	71.1	49.3	37.6	23.2	28.7	58.7	75.0	-	-
BART + PANNs	47.2 (1.4)	27.9 (0.6)	15.6 (0.9)	7.8 (0.9)	11.2 (0.6)	32.2 (0.8)	6.5 (2.5)	6.1 (0.9)	6.3 (1.7)
BART + VGGish	57.6 (2.9)	39.3 (3.2)	26.2 (2.9)	17.0 (2.6)	17.6 (1.8)	39.8 (1.9)	37.6 (8.0)	11.9 (1.3)	24.7 (4.6)
BART + YAMNet	61.1 (0.6)	43.3 (0.5)	30.5 (0.5)	20.8 (0.3)	19.8 (0.2)	43.6 (0.2)	54.7 (0.6)	14.1 (0.2)	34.4 (0.4)
BART + YAMNet + VGGish	65.8 (0.5)	48.7 (0.4)	35.4 (0.4)	25.3 (0.5)	21.9 (0.1)	46.5 (0.1)	63.9 (1.0)	15.9 (0.3)	39.9 (0.7)
BART + YAMNet + PANNs	**69.9 (0.5)**	**52.3 (0.7)**	**38.0 (0.8)**	26.6 (0.9)	24.1 (0.3)	49.3 (0.4)	75.3 (0.9)	17.6 (0.3)	46.5 (0.6)
Human	65.4	48.9	37.3	**29.1**	**28.8**	**49.6**	**91.3**	**21.6**	**56.5**

Table 2: Performance metrics for complementary experiments.

Variant	CIDEr	SPICE	SPIDEr
Reference	75.3	17.6	46.5
No BART pre-training	71.0	16.7	43.8
Frozen decoder param.	68.5	16.6	42.5
BART-XSum	71.8	17.3	44.5
BART-CNN	72.2	17.7	44.2
BART-CNN, frozen decoder param.	70.4	15.6	43.0
BART-base	73.1	16.8	45.0

to BART pre-training as opposed to its high-capacity architecture. We do so by evaluating a model without BART pre-training, i.e. randomly initialized. Discrepancies with the reference setting in Table 2 demonstrate that while catastrophic forgetting, i.e. BART overfitting to the target task and forgetting its pre-trained generic information about language, may occur due to multi-modal inputs, some information is retained from the denoising pre-training. Nonetheless, the large-scale transformer architecture is for a large part responsible for improvements over systems in the literature.

In a subsequent experiment, we freeze parameters of self-attention and MLP blocks in BART decoder layers. These parameters are expected to hold most of the knowledge on language modeling. Encoder parameters, as well as decoder cross-attention and layer normalization weights, remain freely fine-tuned. The number of trainable parameters is about 250 million, reduced from 408 million in the full model. This variant achieves higher SPIDEr than both baselines, which suggests that BART language modeling parameters can already produce high quality captions.

4.4. Pre-training task

Within the proposed conditioning setting, the captioning task solved by BART is related to summarization: the model is given about 10 often recurring AudioSet tags, whereas most captions in the dataset describe one to three sequential events. The authors of BART demonstrated the potential of its pre-training scheme when applied to the CNN/DM [24] and XSum [25] summarization datasets, with model parameters made available to the community.

We investigate the effect of summarization fine-tuning on our captioning method, by replacing regular BART parameters in the proposed model with BART-CNN and BART-XSum checkpoints at initialization. These setups result in slightly lower performance than the reference method in Table 2. However, we observed a significant decrease in the initial training loss in both cases. In addition, freezing decoder parameters (see Section 4.3) in the BART-CNN model produces better syntax compared to the equivalent setting with standard BART initialization, according to CIDEr. This indicates that the language modeling learned for summarization is better suited to captioning than that of denoising. Thus, using the BART-CNN decoder as the starting point may be preferable on smaller captioning datasets, if overfitting prevents training the full BART architecture.

4.5. Model capacity

We investigate the impact of model capacity on the quality of the generated captions. To do so, we replace the standard BART architecture with the BART-base variant provided by the authors. BART-base undergoes the same pre-training scheme with half as many encoder and decoder layers (6 instead of 12) as well as a internal dimension of 768 (from 1024). These modifications reduce the number of trainable parameters from 408 million to about 140 million.

Interestingly, the large reduction in model capacity does not translate to similarly important decrease in performance (see Table 2). We hypothesize that, even though the language modeling capabilities of BART-base are inferior to those of the standard BART, the highly formatted nature of captions requires less knowledge to model than general text in other tasks. This conjecture is in part supported by the fair syntactic quality of captions produced by smaller architectures in the literature. As a byproduct, it is also unlikely that further increasing model capacity from that of the standard BART architecture would yield appreciably higher performances.

5. CONCLUSION

In this paper, we presented an audio captioning scheme by fine-tuning BART with combined audio and textual conditioning. Our results demonstrate that transfer learning can be applied to scale captioning architectures to the size of state of the art NLP models, in spite of the limited data availability. Using pre-trained architectures to retrieve both audio and language guidance material removes the need for dedicated modules, and enables semantic conditioning with high accuracy and controlled generalization properties. We find that the proposed model can be scaled down or partly frozen with limited decreases in performance, hence diminishing the trade-off between high quality caption production and model capacity.

This study highlights interesting avenues for future research. In particular, upcoming work will explore methods to better utilize the knowledge of encoders with text pre-training in multi-modal downstream tasks. Determining the optimal granularity of temporal detail to reduce information redundancy in guidance inputs, or developing adaptive audio segmentation matching separate sound events, will also be investigated in future work.

6. REFERENCES

[1] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 374–378.

[2] K. Nguyen, K. Drossos, and T. Virtanen, "Temporal subsampling of audio feature sequences for automated audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, November 2020, pp. 110–114.

[3] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre- and post-processings for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, November 2020, pp. 190–194.

[4] E. Çakır, K. Drossos, and T. Virtanen, "Multi-task regularization based on infrequent classes for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, November 2020, pp. 6–10.

[5] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A Transformer-Based Audio Captioning Model with Keyword Estimation," in *Proc. Interspeech 2020*, 2020, pp. 1977–1981.

[6] A. O. Eren and M. Sert, "Audio captioning based on combined audio and semantic embeddings," in *2020 IEEE International Symposium on Multimedia (ISM)*, 2020, pp. 41–48.

[7] Y. Koizumi, Y. Ohishi, D. Niizumi, D. Takeuchi, and M. Yasuda. (2020) Audio captioning using pre-trained large-scale language model guided by audio-based similar caption retrieval. [Online]. Available: https://arxiv.org/abs/2012.07331

[8] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, June 2019, pp. 4171–4186.

[10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, July 2020, pp. 7871–7880.

[12] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audioset: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[13] M. Plakal and D. Ellis. Yamnet. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/audioset/yamnet

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[15] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[16] W.-C. Huang, C.-H. Wu, S.-B. Luo, K.-Y. Chen, H.-M. Wang, and T. Toda, "Speech recognition by simply fine-tuning bert," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7343–7347.

[17] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, June 2019, pp. 119–132.

[18] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[19] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *Computer Vision – ECCV 2016*, 2016, pp. 382–398.

[20] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," in *2017 IEEE International Conference on Computer Vision (ICCV)*, oct 2017, pp. 873–881.

[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[22] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Nov. 2016, pp. 2122–2132.

[23] J. Novikova, O. Dušek, A. Cercas Curry, and V. Rieser, "Why we need new evaluation metrics for NLG," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Sept. 2017, pp. 2241–2252.

[24] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gulçehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, Aug. 2016, pp. 280–290.

[25] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Oct.-Nov. 2018, pp. 1797–1807.

MICARRAYLIB: SOFTWARE FOR REPRODUCIBLE AGGREGATION, STANDARDIZATION, AND SIGNAL PROCESSING OF MICROPHONE ARRAY DATASETS

Iran R. Roman[1*], *Juan Pablo Bello*[1,2]

[1] Music and Audio Research Lab, New York University, NY, USA
[2] Center for Urban Science and Progress, New York University, NY, USA

ABSTRACT

`micarraylib` is a python library to load, standardize, and aggregate datasets collected with different microphone array hardware. The goal is to create larger datasets by aggregating existing and mostly incompatible microphone array data and encoding it into standard B-format ambisonics. These larger datasets can be used to develop novel sound event localization and detection (SELD) algorithms. `micarraylib` streamlines the download, load, resampling, aggregation, and signal processing of datasets collected with commonly-used and custom microphone array hardware. We provide an API to standardize the 3D coordinates of each microphone array capsule, visualize the placement of microphone arrays in specific spatial configurations, and encode time-series data collected with different microphone arrays into B-format ambisonics. Finally, we also show that the data aggregates can be used to reconstruct a microphone capsule's time-series data using the information from other capsules in the data aggregate. `micarraylib` will allow for the easy addition of more datasets and microphone array hardware as they become available in the future. All original software written for this paper is released with an open-source license.

Index Terms— sound event detection and localization, microphone arrays, spherical harmonics, ambisonics encoder, multichannel signal processing

1. INTRODUCTION

In the past few years, the field of machine listening has seen major advances in sound event detection (SED) algorithms [1, 2, 3]. These advances have been made possible by the introduction of large datasets with annotated sound events. In particular, the millions of annotated soundclips in AudioSet [4], totaling around 100 hours of data, have been critical for these developments.

In contrast, the development of sound event localization and detection (SELD) algorithms has been slower. This is not surprising, given that SELD datasets are much smaller than AudioSet, usually with only a few thousand sound events with annotations for both category and spatial localization (example datasets include those introduced by the DCASE SELD challenges in 2019 [5], 2020 [6], and 2021 [7], as well as the LOCATA challenge [8]).

To develop SELD algorithms that are as robust as existing SED ones, machine listening researchers will need access to large amounts of data collected with microphone arrays. A number of publicly available microphone array datasets exist (see Table 1), but these datasets have heterogeneous hardware parameters, thus complicating their aggregation.

*roman@nyu.edu

Table 1: Some publicly available microphone array datasets. The number of microphone arrays, total microphone capsules, length in hours, and presence of SELD annotations are tabulated.

dataset	no. arr	capsules	length	SELD
DCASE(3) 2019 [5]	1	4	8 Hr	Yes
DCASE(3) 2020 [6]	1	4	13 Hr	Yes
DCASE(3) 2021 [7]	1	4	13 Hr	Yes
LOCATA [8]	4	63	0.5 Hr	Yes
3D-MARCo [9]	7	71	0.2 Hr	No
EigenScape [10]	1	32	11 Hr	No

Two notorious differences between microphone array datasets include 1) the use of different microphone hardware and 2) conventions for SELD annotation (or complete lack of), including event start time, duration, and position in space. To standardize microphone array recordings across different hardware, some researchers encode them into the ambisonics B-format [11, 12], which uses the individual microphone capsule coordinates to compute a matrix of spherical harmonic coefficients. In its simplest form, the B-format is obtained by multiplying the pseudo-inverse of this matrix by the corresponding raw capsule recordings (also known as A-format) [13]. The ambisonics B-format captures specific spatial features (i.e. the first channel is equivalent to an omnidiectional microphone, the next three channels are fig-8 microphones aligned on the x, y, and z Cartesian coordinates, etc. see [14] for details on the spherical harmonics theory that results in B-format encoding). On the other hand, standardizing SELD annotations is possible if certain parameters (i.e. event start time, end time, and a position in space) are parsed to be consistent across datasets. Additionally, since not all microphone array datasets have SELD annotations, the use of unsupervised and self-supervised learning strategies to learn spatial representations will be necessary to use all available datasets.

Here we introduce `micarraylib`, a python library to download, standardize, and aggregate existing microphone array recordings. Using `micarraylib`, one can encode raw microphone array recordings across different datasets to be in the common ambisonics B-format. `micarraylib` also standardizes annotations to be in a common convention. Additionally, `micarraylib` organizes metadata (i.e. microphone capsule coordinates and hardware name) to be readily accessible. `micarraylib` is freely available at `https://github.com/micarraylib/micarraylib`.

In the next sections we describe `micarraylib`'s functional principles and show example applications, which include the aggregation of different SELD datasets, visualization of aggregated microphone coordinates, and data augmentation via interpolation of a

virtual capsule recording using data from neighboring capsules.

2. LIBRARY FUNCTION

We standardize three elements present in most microphone array datasets: 1) metadata, 2) SELD labels (if any), and 3) audio format.

2.1. Metadata processing

The most important piece of metadata accompanying any microphone array dataset is its microphone capsule coordinates. Microphone array datasets often use different microphone hardware. Designers publish the relative distance between microphone capsules and a reference point. These can be converted to 3D coordinates (Cartesian or polar). In some datasets the reference point is the microphone array's center, or a specific location in a physical space. `micarraylib` standardizes these distances and locations to be a common coordinate format (3D vectors, either Cartesian or polar). These coordinates can be used to visualize microphone arrays, to compute spherical harmonics, and more generally to develop algorithms that incorporate spatial information at the level of microphone capsule location.

Other pieces of metadata that `micarraylib` processes (if available) and makes available to the user include sound scene category, musical artist, and geographic location of the recording.

2.2. SELD label standardization

Because of their unique ability to capture spatial information, microphone arrays are a common hardware choice to collect SELD data. While several labeling conventions exist among datasets, annotations for sound events include at least a start time, but may also have an end time, and a position in space. `micarraylib` standardizes sound event labels across datasets to have the following format: start and end time (python tuple), object category (a unique integer or string), location coordinates (3D vector that may change over a fixed time-step if the event moves), and active time-steps (list of booleans if the event is transiently on or off). When one of these parameters does not exist for an event, `micarraylib` will indicate it with a `None`. By using `micarraylib`, researchers will access microphone array datasets with a standard format for spatial sound event labels.

2.3. Audio standardization

The ambisonics B-format allows for the standardization of microphone array recordings [11, 12]. A simple ambisonics encoder uses the individual microphone capsule coordinates to compute a matrix of spherical harmonics. The pseudo-inverse of this matrix then multiplies the raw capsule recordings to encode them into B-format channels.

While more complicated encoders are usually proprietary and include multiple effects that aid perceptual parameters [15], computing the spherical harmonics using microphone array coordinates is a straightforward operation if the microphone capsule coordinates are known. The Nth-order spherical harmonic matrix can be computed using equation 1.

$$Y_{n,l}(\theta,\phi) = X_{n,|l|}P_{n,|l|}\cos(\theta)\begin{cases}\sqrt{2}\sin(|l|\phi) & \text{if } l<0\\ 1 & \text{if } l=0\\ \sqrt{2}\cos(l\phi) & \text{if } l>0\end{cases} \quad (1)$$

This is the conventional equation used in the field of acoustics to compute the Laplace spherical harmonics [16]. n indexes the spherical harmonic order (i.e. 0th, 1st, 2nd, ..., Nth) and m indexes the degree (i.e. each order n has degrees $m \in [-n, n]$ degrees). $\theta \in [0, \pi]$ is the vertical angle advancing from top to bottom, $\phi \in [0, 2\pi]$ is the azimuth angle starting at the front of the microphone array and advancing counter-clockwise. $X_{n,|m|}$ is a normalization factor that ensures that spherical harmonics have unit magnitude [16] and $P_{n,|m|}$ is the Legendre function (without the Condon-Shortley phase) [17].

`micarraylib` converts raw capsule recordings into the common ambisonics B-format. This results in audio signals that have shared spatial characteristics across channels, independent of which microphone hardware was used to collect them.

3. LIBRARY ORGANIZATION AND COPYRIGHT

`micarraylib` is written in python and all its contents are open-source. The following subsections describe the organization of its file structure as seen from its root directory.

3.1. Micarrays

Directory that contains the `array_shapes_raw.py` and `array_directions_raw.py` files, which list the raw (as released by the manufacturer) shape (i.e. coordinates) and capsule directionality of each microphone array supported. The names that the manufacturer gave to each capsule in the microphone array are also included in these files. The `micarray.py` file in this directory defines a `micarray` object with attributes that summarize all the information provided by the manufacturer.

This directory also has files with functions that process the data from the `array_shapes_raw.py` and `array_directions_raw.py` files and standardize it to be in 3D Cartesian and/or polar coordinates.

3.2. Util

Directory that contains a `utils.py` file with basic functionalities for dataset standardization, such as functions to convert between polar and Cartesian coordinates, normalize units of length to meters and radians, and normalize time units to seconds (in SELD labels, for example). It also contains a `plotting.py` file with functions that tailor `matplotlib`'s plotting for microphone arrays.

3.3. Encoder

File defining the `encoder` object with its main attribute being a set of capsule coordinates (used to calculate the matrix of spherical harmonics). It also has an `encode` method that takes a numpy array with raw recordings and returns a simple encoding of these recordings in ambisonics B-format.

3.4. Dataset

File that defines the `dataset` object using the `soundata` API [18]. `soundata` is a new python library with tools to download and load common audio datasets with corresponding annotations and metadata. In addition to `soundata` attributes, the `dataset` object includes a list of microphone capsule coordinates used. The `soundata` API includes all methods to download and load the

original data. micarraylib standardizes the SELD annotations when they are not standardized by soundata.

3.5. Aggregator

File that defines the aggregate_datasets object, whose attributes include a list of dataset objects. Its default method standardizes all recordings across datasets to be the same number of channels in ambisonics B-format, and pairs individual recordings with their corresponding SELD labels.

It also defines the micarray_aggregate object, which aggregates coordinates and recordings across microphone arrays (useful when multiple pieces of microphone array hardware are used together in a single dataset, such as the 3D-MARCo or LOCATA datasets).

3.6. Augmentation

File that defines a data_augmentation object, which uses a neural network model to virtually add capsule recording data to a microphone array dataset at a coordinate defined by the user. Section 5 below describes the current functionality of this model (which is limited to reconstruction of channels within the EigenMike [19] hardware at the time of this writing, but we are working to expand its possibilities).

3.7. Copyright

micarraylib is released with a Creative Commons License. We also do not alter any dataset's license, as micarraylib only accesses data already hosted online (via soundata; as a result, datasets are not redistributed by micarraylib).

4. AGGREGATING DATASETS

micarraylib streamlines the aggregation of existing microphone array datasets. Figure 1 shows the code needed to standardize and aggregate the six different datasets in Table 1.

One at a time, micarraylib separately encodes each recording into a first-order ambisonics B-format (4 channels total; the first-order ambisonics limit is determined by the dataset with the lowest number of raw capsule recordings: 4 channels in the DCASE SELD datasets). After the simple encoding step, we have a total of 46 hours of audio data in a common ambisonics B-format. micarraylib also standardizes the SELD labels from the DCASE SELD and LOCATA datasets to have a start and end time, object category, spatial coordinates, and active time-steps. The 3D-MARCo and EigenScape datasets do not have SELD labels, and the resulting aggregate indicates this with None entries in the label attribute for those specific recordings. In the end, 34 hour of data in this dataset aggregate have labeled sound events.

4.1. Hardware considerations and next steps

Aggregating different datasets can result in SELD methods that confound elements that are different between datasets (i.e. hardware, events, and/or ambient). While our library encodes all datasets into the standard ambisonics B-format, it is important to keep in mind that the hardware differences between datasets could remain in the B-format. For this reason, we plan to continue fine-tuning our encoder to quantify and reduce differences between hardware. To better quantify these effects, datasets with scenes and events that

```python
import micarraylib as mc

datadir = '~/datasets'

datasets = [
    mc.datasets.dcase19(datadir),
    mc.datasets.dcase20(datadir),
    mc.datasets.dcase21(datadir),
    mc.datasets.locata(datadir),
    mc.datasets.marco(datadir),
    mc.datasets.eigenscape(datadir)
]

for dataset in datasets:
    dataset.load() # using the soundata API [18]

aggregate = mc.aggregators.aggregate_datasets(
    datasets,
    sr=24000,
)
```

Figure 1: Downloading, loading, and aggregating the six datasets in Table 1 using micarraylib.

.

are simultaneously collected with different hardware (i.e. the 3D-MARCo and LOCATA datasets) will be particularly useful.

5. DATA AUGMENTATION

An idealized spatial recording of a sound scene would record information at all locations in the space continuum. Since such idealized scenario is not possible with existing hardware, researchers must sample specific locations using microphone arrays with capsules at specific coordinates. micarraylib includes a model can be used to virtually add microphones to a dataset via interpolation from existing microphone capsule data.

5.1. Technical motivation

Aggregating microphone arrays can lead to denser spatial sampling of a sound scene. The resulting dense samplings of a sound scene are redundant [20]. Therefore, given a set of microphone capsules recording a common scene or source, it should be possible to interpolate, with some error, one of the capsule's time-series using the recordings collected with all other capsules. If this is possible, it should also be possible to virtually generate the recording of a microphone capsule outside but near the microphone array topology.

While a detailed empirical study of virtual microphone capsule time-series generation deserves a separate scientific report, micarraylib already includes some of this functionality. Here we describe a series of experiments that we carried out to design a model able to virtually add the recording of a missing microphone capsule using the recordings from other capsules in the EigenMike microphone array. These experiments also show the utility of micarraylib to aggregate datasets that can then be used for machine listening research.

In all experiments we ask the question: can the recording of a microphone capsule be reconstructed given the preceding 5 milliseconds of recordings with neighboring capsules? We hypothesize that such reconstruction is possible using both the recordings from neighboring capsules and their 3D spatial coordinates.

Table 2: Model performance for capsule interpolation experiments

Model	MSE (eval)
before training	0.9
exp 1	0.000039
exp 2	0.00013
exp 3	0.0016

5.2. Data

We used `micarraylib` to aggregate the EigenMike data from the EigenScape, 3D-MARCo, and LOCATA datasets. We skip the encoding step and keep the recordings in the raw format (A-format). We split each recording into development (the first 80% frames) and testing (the last 10% frames) subsets (the frames between the %80 and 90% of each recording length were left out to minimize the effects of temporal correlation between the development and testing data).

5.3. Methodology

The input features to train our model are an intermediate step between A-format and B-format that is computed as follows. We calculate the fourth order spherical harmonic matrix using the 3D coordinates for each EigenMike capsule and equation (1). The result is a matrix W that has $(N+1)^2$ rows ($N = 4$) and 32 columns, one corresponding to each EigenMike capsule. We multiply the sample recorded by a given capsule with the spherical harmonic coefficients in the corresponding column of matrix W. We stack 20 of these matrices into a tensor that corresponds to 5 consecutive milliseconds of samples in a recording. We refer to these concatenated matrices as tensor X.

In the experiments described below we remove some audio information from tensor X and we train a simple LeNet CNN [21] to use X to reconstruct the recording of a single capsule (whose audio information is missing from X) with a MSE loss function.

5.4. Experiments and results

We carried out three experiments: 1) Tensor X is only missing the audio information of the single capsule that we want to reconstruct. 2) The samples of five neighboring capsules are also removed from X. 3) X only has the audio information of the three capsules that would result in a tetrahedral geometry (4 capsules total) with respect to the capsule whose recording we want to reconstruct.

We trained a LeNet CNN model from scratch using the ADAM optimizer with a patience criterion of 1000 epochs. For each of these three experiments, the model's task was to reconstruct the missing samples of a single microphone capsule in the EigenMike using X as input. The average MSE across datapoints in the evaluation set was computed after training and is shown in Table 2.

As shown in Table 2, the evaluation MSE before training was close to 1. Experiment 1 reduced this MSE by 5 orders of magnitude, showing that a single capsule's samples can be reconstructed (with some error) using all other 31 capsule recordings in the EigenMike. Experiment 2 also reduced the MSE but only by 4 orders of magnitude, showing that reconstruction of the same capsule's recording is affected by the fact that audio information from neighboring capsules was missing. Experiment 3 only reduced the MSE by 3 orders of magnitude due to there only being information from

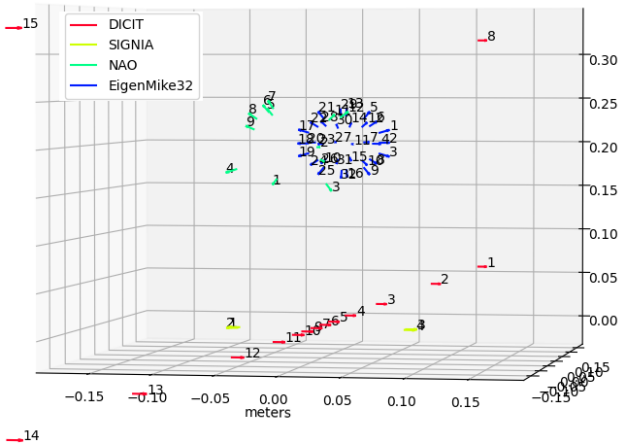


Figure 2: Interactive visualization with `micarraylib` of the location and directionality of microphone arrays used to collect the LOCATA dataset, one of which is the EigenMike. Using `matplotlib`'s 3D plotting, `micarraylib` includes methods to plot capsules in microphone array hardware.

3 other microphone capsules to reconstruct the capsule's recording. These results show that reconstructing a capsule's samples using other capsule's recordings is possible, but the interpolation is sensitive to the density and proximity of recordings available to carry out this reconstruction.

5.5. Next steps

Given that reconstructing a microphone's signal seems possible within EigenMike capsules, we will train a model able to use EigenMike data to recover the signal captured by a capsule or array outside the EigenMike geometry. For this purpose, we will use the microphone array aggregates from the 3D-MARCo and LOCATA datasets, which include data recorded by microphone capsules outside the EigenMike structure but very near to it (see Fig 2). Our goal is for `micarraylib` to ultimately allow for the virtual simulation of microphone capsules in arbitrary coordinates.

The idea of virtual microphone capsule generation has already been explored with supervised learning using neural network techniques like CNNs [22, 23] or autoencoders [24], via statistical interpolation with β-divergence [25] or with pure signal processing [26]. The full potential of the microphone array data augmentation that we propose will be achieved by using `micarraylib` in combination with audio augmentation libraries like MUDA [27] and Audiomentations (`https://github.com/iver56/audiomentations`).

6. CONCLUSIONS AND FUTURE DIRECTIONS

We have presented `micarraylib`, a library to aggregate microphone array datasets by standardizing microphone array coordinates, SELD labels, and recordings using a simple ambisonics B-format encoder. Our presentation of the library also included practical demonstrations that show both the utility of `micarraylib` and the need for it in the field of machine listening. Researchers will be able to use `micarraylib` to create large aggregates of standardized microphone array datasets.

We are also looking forward to seeing what other functionalities the machine listening research community believes should be added to `micarraylib`. We will continue adding new datasets to `micarraylib` in years to come, and we will also add support for more complex signal processing procedures that include completely custom array shapes, microphone polarity, methods for perceptually-plausible ambisonics encoding and decoding, as well as processing of motion-capture data for moving sound events and microphone arrays.

7. ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under grant no. IIS-1955357. The authors thank the founding source and their grant collaborators.

8. REFERENCES

[1] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.

[2] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General perception with iterative attention," *arXiv preprint arXiv:2103.03206*, 2021.

[3] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 126–130.

[4] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[5] S. Adavanne, A. Politis, and T. Virtanen, "A multi-room reverberant dataset for sound event localization and uetection," in *Submitted to Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019. [Online]. Available: https://arxiv.org/abs/1905.08546

[6] A. Politis, S. Adavanne, and T. Virtanen, "A dataset of reverberant spatial sound scenes with moving sources for sound event localization and detection," in *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, November 2020. [Online]. Available: https://arxiv.org/abs/2006.01919

[7] A. Politis, S. Adavanne, D. Krause, A. Deleforge, P. Srivastava, and T. Virtanen, "A dataset of dynamic reverberant sound scenes with directional interferers for sound event localization and detection," *arXiv preprint arXiv:2106.06999*, 2021. [Online]. Available: https://arxiv.org/abs/2106.06999

[8] H. W. Löllmann, C. Evers, A. Schmidt, H. Mellmann, H. Barfuss, P. A. Naylor, and W. Kellermann, "The LOCATA challenge data corpus for acoustic source localization and tracking," in *2018 IEEE 10th Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, 2018, pp. 410–414.

[9] H. Lee and D. Johnson, "An open-access database of 3D microphone array recordings," in *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

[10] M. C. Green and D. Murphy, "EigenScape: A database of spatial acoustic scene recordings," *Applied Sciences*, vol. 7, no. 11, p. 1204, 2017.

[11] J. Bamford and J. Vanderkooy, "Ambisonic sound for us (an analysis of imaging in ambisonics, stereo and dolby surround systems)," in *99th AES Convention*, 1995, pp. 6–9.

[12] F. Zotter, H. Pomberger, and M. Frank, "An alternative ambisonics formulation: Modal source strength matching and the effect of spatial aliasing," in *Audio Engineering Society Convention 126*. Audio Engineering Society, 2009.

[13] L. McCormack, S. Delikaris-Manias, A. Farina, D. Pinardi, and V. Pulkki, "Real-time conversion of sensor array signals into spherical harmonic signals with applications to spatially localized sub-band sound-field analysis," in *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

[14] K. Atkinson and W. Han, *Spherical harmonics and approximations on the unit sphere: an introduction*. Springer Science & Business Media, 2012, vol. 2044.

[15] M. Frank, F. Zotter, and A. Sontacchi, "Producing 3D audio in ambisonics," in *Audio Engineering Society Conference: 57th International Conference: The Future of Audio Entertainment Technology–Cinema, Television and the Internet*. Audio Engineering Society, 2015.

[16] E. G. Williams, *Fourier acoustics: sound radiation and nearfield acoustical holography*. Academic press, 1999.

[17] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. US Government printing office, 1948, vol. 55.

[18] M. Fuentes, J. Salamon, P. Zinemanas, M. Rocamora, G. Plaja, I. R. Román, R. Bittner, M. Miron, X. Serra, and J. P. Bello, "Soundata: A Python library for reproducible use of audio datasets," *http://arxiv.org/abs/2109.12690*, 2021.

[19] M. Acoustics, "EM32 Eigenmike microphone array release notes (v17. 0)," *25 Summit Ave, Summit, NJ 07901, USA*, 2013.

[20] Y. Huang, T. Z. Shabestary, A. Gruenstein, and L. Wan, "Multi-microphone adaptive noise cancellation for robust hotword detection," 2019.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] T. Ochiai, M. Delcroix, T. Nakatani, R. Ikeshita, K. Kinoshita, and S. Araki, "Neural network-based virtual microphone estimator," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6114–6118.

[23] K. Yamaoka, L. Li, N. Ono, S. Makino, and T. Yamada, "CNN-based virtual microphone signal estimation for MPDR beamforming in underdetermined situations," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[24] R. Takahashi, L. Li, S. Makino, and T. Yamada, "VMInNet: Interpolation of virtual microphones in optimal latent space explored by autoencoder."

[25] H. Katahira, N. Ono, S. Miyabe, T. Yamada, and S. Makino, "Generalized amplitude interpolation by β-divergence for virtual microphone array," in *2014 14th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2014, pp. 149–153.

[26] H. Katahira, N. Ono, S. Miyabe, T. Yamada, and S. Makino, "Virtually increasing microphone array elements by interpolation in complex-logarithmic domain," in *21st European Signal Processing Conference (EUSIPCO 2013)*. IEEE, 2013, pp. 1–5.

[27] B. McFee, E. J. Humphrey, and J. P. Bello, "A software framework for musical data augmentation." in *ISMIR*, vol. 2015, 2015, pp. 248–254.

IMPROVED STUDENT MODEL TRAINING FOR ACOUSTIC EVENT DETECTION MODELS

Anthea Cheung, Qingming Tang, Chieh-chi Kao, Ming Sun, Chao Wang

Amazon Alexa
101 Main St, Cambridge, MA 02142, USA
{antheach, qmtang, chiehchi, mingsun, wngcha}@amazon.com

ABSTRACT

We introduce several novel knowledge distillation techniques for training a single shallow model of three recurrent layers for acoustic event detection (AED). These techniques allow us to train a generic shallow student model without many convolutional layers, ensembling, or custom modules. Gradual incorporation of pseudolabeled data, using strong and weak pseudolabels to train our student model, event masking in the loss function, and a custom SpecAugment procedure with event-dependent time masking all contribute to a strong event-based F1-score of 42.7%, which matches the top submission score, compared to 34.7% when training with a generic knowledge distillation method. For comparison to state-of-the-art performance, we use the ensemble model of the top submission in the challenge as a fixed teacher model.

Index Terms— Acoustic event detection, knowledge distillation, pseudolabeling, SpecAugment

1. INTRODUCTION

Acoustic event detection (AED) is the task of predicting sound events and their time boundaries. It is an emerging area of research as the ability to correctly detect the start and end times of sound events has many useful practical applications in media indexing and retrieval [1], surveillance [2], enhancing smart home devices' ability to interpret the acoustic environment of the home [3]. Compared to audio tagging tasks, acoustic event detection remains a challenging area of research due to the difficulty of obtaining high-quality annotated clips which contain labels of onset and offset times.

We focused on task four of the 2019 edition of DCASE as it is solely focused on AED and provides a test dataset that can be compared with the top performances in the challenge [4]. In this task, the top-performing submissions are ensembles of models [5, 6] or comprised of multiple layers of convolutional layers [7, 6] and modules with custom architecture [5], which often consume significant memory and are less practical to use in resource-constrained mobile devices. Our focus is on using knowledge distillation techniques to achieve a single shallow model of three recurrent layers with a small degradation in accuracy. We used the ensemble model of the top submission in the challenge as a fixed teacher model.

Recent results on noisy student training explored promising techniques for an automatic speech recognition (ASR) task [8]. Firstly, they tried gradually introducing harder samples to the training of the student model by applying a score on each utterance-transcript pair and lowering the cutoff score for each generation of training. Curriculum learning applies a similar concept in slowly expanding the training set for the student model. Secondly, they performed SpecAugment [9] and increased the time masking length to produce harder samples for the student model. To our knowledge, neither techniques have been explored for AED before. We apply gradual incorporation of pseudolabeled data, strong and weak pseudolabels to train our student model, and a custom SpecAugment procedure with event-dependent time masking to achieve a strong event-based F1-score of 42.7%.

2. RELATED WORK

Recently, the use of deep learning models with convolutional neural network (CNN) [5, 10] and convolutional recurrent network network (CRNN) [4] architectures have yielded the best performance in AED tasks. More recently, custom solutions such as disentangled features [11] and independent component [12] modules have been added on top of CNN or CRNN architectures to further refine performance. As strongly-labeled AED datasets are relatively small, semi-supervised methods are used to take advantage of unlabeled and weakly-labeled sets, either by only using weak predictions [5, 13] or both strong and weak predictions [14, 4]. Knowledge distillation [15] has been studied for AED using only weak labels [16] or using weak and strong labels in two stages [17]. Mean-teacher models [18] use a similar concept in applying a consistency loss to student and teacher models with the same architecture.

The importance of different time scales of the present events are evident in the post-processing steps of several systems for AED [7, 19]. These improvements inspired us to mask the input features and predictions based on the labeled classes. Masking of time and frequency bands is used in SpecAugment [9], but most AED systems only use time and frequency masking not time warping [20, 21]. Partially masking the model outputs during gradient descent have been used for AED [22] and for localization tasks [23]. Curriculum learning [24] is a strategy of gradually adding more difficult samples during training, and has been used to train ASR [8], emotion recognition [25], and translation models [26]. Tonami et al [27] studied curriculum for AED but ranked samples' difficulty based on the presence of labeled classes.

3. METHODOLOGY

The DCASE 2019 task 4 dataset consists of 10 different sound events from domestic environments. The training dataset contains synthetic clips strongly-labeled with onset and offset times, weakly-labeled real recordings that contain event labels but no onset and offset times, and a large unlabeled dataset of real recordings. The validation and public test sets are both strongly-labeled datasets with real recordings.

3.1. Our model

Our student model $\mathcal{S}$ was designed to be a simple recurrent neural network (RNN) model, achieving close to state-of-the-art performance solely relying on data augmentation and knowledge distillation techniques without requiring hand-crafted features or custom architectures. We provided as input $\boldsymbol{X} \in R^{d \times T}$ log-mel features with dimension $d = 20$ and time steps $T = 500$. Our model uses a three-layer uni-directional LSTM architecture with $h = 256$ nodes in each layer to produce an embedding $g(\boldsymbol{X}) = \boldsymbol{H} \in R^{h \times T}$. To generate the onset and offset times of each predicted event c in the clip, we obtained a frame-level prediction $\boldsymbol{f} \in R^{C \times T}$ by passing $\boldsymbol{H}$ through a fully-connected layer $\boldsymbol{W}$ with C output classes and a sigmoid activation function.

We then used an attention-pooling mechanism to generate the audio tagging predictions. For each class c, the attention weights $\boldsymbol{z}_c \in R^T$ are obtained by:

$$\boldsymbol{z}_c = \frac{exp(\boldsymbol{a}_c \boldsymbol{H} + \boldsymbol{b}_c)}{\sum_{k=1}^{C} exp(\boldsymbol{a}_k \boldsymbol{H} + \boldsymbol{b}_k)}, \tag{1}$$

where $a_k, b_k \in R^h$ are the class weights and bias vectors for class k. The clip level audio tagging outputs are obtained by normalizing the frame outputs $\boldsymbol{f}_c$ with the attention weights $\boldsymbol{z}$. All frame predictions for events with a clip level prediction below the threshold of 0.5 were set to zero so that only clips above the threshold also had positive frame predictions.

3.2. Use of pseudolabels

We aim to understand the relative benefits of using pseudolabels, and whether or not progressively incorporating easier or harder samples yield better results. To that end, we used the teacher model $\mathcal{T}$ from the top-performing submission in the DCASE 2019 task 4 challenge [5], which is an ensemble of six CNNs with an attention pooling layer. The audio tagging and detection results from teacher are generated for the unlabeled set without applying post-processing steps, and used as targets for training the student model. After the end of an epoch, the pseudolabeled samples can be evaluated by the student model. At each generation of training, we deemed samples whose student model predictions more closely match the teacher model's soft targets as easier samples. We used the following heuristic to score the difficulty of each sample $\boldsymbol{X}$ using the weak predictions $\boldsymbol{t}, \boldsymbol{s} \in R^C$ of the teacher and student models, respectively:

$$\mu(\boldsymbol{t}, \boldsymbol{s}) = max_c(|\boldsymbol{t}_c - \boldsymbol{s}_c|), \tag{2}$$

which is the maximum difference in the teacher and student scores across all C classes. The maximum difference rather than mean is chosen to ensure that the score discrepancies across all events are below $\mu(\boldsymbol{t}, \boldsymbol{s})$.

3.3. Customized SpecAug procedure

Although the standard SpecAug [9] procedure applies a fixed length of time masking to each clip, the average duration of different sound events vary greatly. For example, the duration of dishes clanging is much shorter than that of vacuuming, so a model for vacuum sound should be robust to longer time masks compared to a model for dishes. Applying this principle, we devised a customized procedure that varies the length of the time mask. For clips from the unlabeled dataset, we used the weak soft targets generated by the teacher as labels; otherwise we use the original labels. We added random noise $\varepsilon_c \sim N(0, 1\mathrm{e}-6)$ for each event category to get noisy labels $\widetilde{\boldsymbol{Y}_c} = \boldsymbol{Y}_c + \varepsilon$. For the top K events in $\widetilde{\boldsymbol{Y}_c}$, we apply a time mask to $\boldsymbol{X}$ with the length given by $\gamma \cdot l_c$, where l_c is the median frame length of event c. We also masked frequency bands F times with fixed length L_f in the same way as the standard SpecAug procedure. Our tunable hyperparameters are F and L_f for frequency masking and K and γ for time masking.

4. TRAINING

We applied the same procedure as that of the original teacher model submission to produce 64-dimensional log-mel filterbank features. A window length of 40 ms, hop length of 20 ms, and 2048 number of fft components were used to produce 500 frames for each audio clip. For the student model, we used 20-dimensional log-mel filterbank features with the same window and hop lengths to produce 500 frames for each clip.

4.1. Loss function

Our dataset consists of: 1) a strongly-labeled synthetic dataset $\mathcal{L}^S$; 2) a weakly-labeled dataset $\mathcal{L}^W$; and 3) an unlabeled dataset $\mathcal{U}$. For each training sample $\boldsymbol{X}$ of the strongly-labeled synthetic dataset, we have strong audio detection labels $\boldsymbol{Y}^s$, and inferred weak labels $\boldsymbol{Y}^w$, where each class has $\boldsymbol{Y}_c^w = \max_t(\boldsymbol{Y}_{c,t}^s)$, while the weakly-labeled dataset only has weak labels $\boldsymbol{Y}^w$. For each sample in the unlabeled dataset, we denote the teacher strong and weak predictions by $\boldsymbol{t}^s$ and $\boldsymbol{t}^w$, respectively, and the student strong and weak predictions by $\boldsymbol{s}^s$ and $\boldsymbol{s}^w$. During training, the loss function is composed of the weak loss, strong loss, and unlabeled loss with hyperparameters λ_1 and λ_2 which are weights for the clip and frame level losses:

$$\mathcal{J} = J^w + J^s(\lambda_1, \lambda_2, \boldsymbol{M}) + J^u(\lambda_1, \lambda_2, \boldsymbol{M}) \tag{3}$$

where the weak loss J^w is the binary cross-entropy loss

$$J^w = \frac{1}{|\mathcal{L}^w|} \sum_{X \in \mathcal{L}^w} (\boldsymbol{Y}^w \log(\hat{\boldsymbol{Y}^w}) + (\boldsymbol{1} - \boldsymbol{Y}^w) \log(\boldsymbol{1} - \hat{\boldsymbol{Y}^w})) \tag{4}$$

and the strong loss is

$$J^s = \frac{\lambda_1}{|\mathcal{L}^s|} \sum_{\boldsymbol{X} \in \mathcal{L}^s} (\boldsymbol{Y}^w \log(\hat{\boldsymbol{Y}^w}) + (\boldsymbol{1} - \boldsymbol{Y}^w) \log(\boldsymbol{1} - \hat{\boldsymbol{Y}^w})) + \frac{\lambda_2}{|\mathcal{L}^s|} \sum_{\boldsymbol{X} \in \mathcal{L}^s} (\boldsymbol{Y}^s \log(\boldsymbol{M} \odot \hat{\boldsymbol{Y}^s}) + \boldsymbol{1} - \boldsymbol{Y}^s) \log(\boldsymbol{1} - \boldsymbol{M} \odot \hat{\boldsymbol{Y}^s})) \tag{5}$$

The loss J^s for strongly-labeled samples is a weighted sum of the clip-level and framewise binary cross-entropy losses. Since the labels in the dataset are sparse, the average framewise loss can be quite inefficient as the composition of the loss may be dominated by the cross-entropy loss of negative frames. Thus, we compared the results of three different types of masking: 1) no masking; 2) event masking; and 3) segment masking.

In the case of no masking, $\boldsymbol{M} \in R^{C \times T}$ is simply a matrix of ones. For event masking, we take the Hadamard product of the predictions $\hat{\boldsymbol{Y}^s}$ and the mask

$$\boldsymbol{M}_{ij} = \begin{cases} 1, & Y_i = 1 \\ 0, & Y_i = 0 \end{cases}, \tag{6}$$

Experiment	Curriculum	Pseudolabels	Masking	SpecAug	Best val F1	Best test F1	Mean±sd val F1	Mean±sd test F1
Lin_ICT_3	-	-	-	-	45.3	42.7	-	-
Lin_ICT_2	-	-	-	-	44.0	40.9	-	-
EF+SW+EM+CS	Easier	Strong+weak	Event	Custom	41.6	42.7	40.7 ± 0.6	41.3 ± 0.9
SW+EM+CS	All	Strong+weak	Event	Custom	41.3	42.5	40.3 ± 0.7	41.0 ± 1.1
HF+SW+EM+CS	Harder	Strong+weak	Event	Custom	40.7	42.2	40.3 ± 0.8	40.3 ± 1.2
SW+NM+NS	All	Strong+weak	None	None	34.1	34.7	33.1±0.7	33.5±0.9

Table 1: Comparison of pseudolabel scheduling (easier first, harder first, or adding all at once).

Experiment	Curriculum	Pseudolabels	Masking	SpecAug	Best val F1	Best test F1	Mean±sd val F1	Mean±sd test F1
EF+SW+EM+CS	Easier	Strong+weak	Event	Custom	41.6	42.7	40.7 ± 0.6	41.3 ± 0.9
EF+SW+EM+SS	Easier	Strong+weak	Event	Standard	40.7	41.6	40.0 ± 0.4	40.9 ± 0.6
EF+SW+EM+NS	Easier	Strong+weak	Event	None	40.2	41.2	39.5 ± 0.4	39.9 ± 0.7
EF+SW+EM+CS	Easier	Strong+weak	Event	Custom	41.6	42.7	40.7 ± 0.6	41.3 ± 0.9
EF+SW+SM+CS	Easier	Strong+weak	Segment	Custom	39.9	40.0	39.2 ± 0.4	39.1 ± 0.5
EF+SW+NM+CS	Easier	Strong+weak	None	Custom	35.7	34.5	33.9 ± 0.8	32.9 ± 0.8

Table 2: Comparison of data augmentation methods (custom SpecAug, standard SpecAug, and no augmentation) and different masking schemes.

so that only frames of present events contribute to the framewise loss. In the case of segment masking, we use the mask

$$\boldsymbol{M}_{ij} = \begin{cases} 1, & Y_{i+12} = 1 \text{ or } Y_{i-12} = 1 \text{ or } Y_i = 1 \\ 0, & \text{otherwise} \end{cases}, \quad (7)$$

that consists of 1's for corresponding onset and offset frames of each event with a 12-frame buffer before and after each segment (0.24 seconds before and after the onset and offset, respectively).

4.2. Post-processing

After obtaining the framewise outputs for the detection task, we applied the same post-processing step as the procedure in the teacher model. A median filter is applied to each event type, with a window size 1/3 the median number of frames for each event type in the synthetic labeled set.

5. RESULTS

We trained the following types of experiments:

1. Adding in all samples compared to adding easier or harder samples of the pseudolabeled dataset first;
2. Applying no data augmentation compared to using standard SpecAug and our custom SpecAug procedure;
3. Applying no masking, event masking, or segment masking to the loss function;
4. Using only weak or weak and strong pseudolabels on the unlabeled dataset.

Each experiment is trained with batch size 16 and learning rate 0.001 on an Adam optimizer with weights $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The macro event-based F1-score on the validation set is computed at the end of each epoch, and the weights of the best epoch is saved. As the number of training steps for the epochs of the experiments for may differ, in experiment 1, we trained each trial for 48,790 training steps and verified that the validation metric has converged for each experiment type.

5.1. Effect of adding pseudolabels in different stages

We compared the effect of adding all pseudolabels at once and adding easier or harder samples of the pseudolabeled dataset first. For all experiments, we trained in generations of five epochs each. In the first generation, only the labeled dataset was added. For our control group experiment, we addded all the pseudolabels; otherwise, we evaluated the difficulty of each pseudolabel by generating the scores as described in section 3.2. All the samples were ranked in order of their scores, where higher scores mean that the samples are harder for the student model as the student predictions are farther away from the teacher predictions. For the next two generations, the bottom 20% and bottom 40% scoring pseudolabel samples were added to the training set when incorporating easier samples first. We reversed this when incorporating harder samples first, i.e. the top 20% and top 40% scoring pseudolabel samples are added in generations two and three, respectively. Starting from the fourth generation, all pseudolabels were added.

For hyperparamter tuning, we tested each experiment type with multiple hyperparameter values for the loss weights $\lambda_1 \in \{0.5, 0.75, 1, 1.5, 2\}$ and $\lambda_2 \in \{0.5, 0.75, 1, 1.5, 2\}$. After finding the best hyperparameters for each experiment type, we repeated the training process with different random initializations to obtain a total of 10 different runs for each configuration.

The results are summarized in Table 1. We compared our results to the top submission in the challenge (Lin_ICT_3) and the top single model by the same team (Lin_ICT_2). Four models are compared: SW indicates that both strong and weak pseudolabels were used, EM indicates event masking, and CS indicates custom SpecAug procedure. The last experiment (SW+NM+NS) is a baseline knowledge distillation result that does not add masking or augmentation. All pseudolabels are added at once in SW+EM+CS, whereas easier samples are added first in EF+SW+EM+CS, and harder samples are added first in HF+SW+EM+CS. The results show that the best performance is attained by adding easier samples first, with a best event-based macro F1 score of 42.7%, on par with the best performing challenge submission. There is a mod-

Experiment	Curriculum	Pseudolabels	Masking	SpecAug	Best val F1	Best test F1	Mean±sd val F1	Mean±sd test F1
EF+SW+EM+CS	Easier	Strong+weak	Event	Custom	41.6	42.7	40.7 ± 0.6	41.3 ± 0.9
EF+W+EM+CS	Easier	Weak only	Event	Custom	28.4	28.6	26.0 ± 1.0	25.6 ± 1.8
EM+CS	N/A	None	Event	Custom	23.4	24.7	21.0 ± 1.6	21.2 ± 2.0

Table 3: Validation and test F1 scores for using weak, strong, and no pseudolabels.

Comparison	t-statistic	Statistically significant at		
		$\alpha = 0.2$	$\alpha = 0.05$	$\alpha = 0.01$
Easier first vs all	1.413	Y	N	N
Harder first vs all	0.143	N	N	N
Easier vs harder first	1.483	Y	N	N
Cust. SpecAug vs std.	2.802	Y	Y	N
Cust. SpecAug vs none	7.055	Y	Y	Y
Std. SpecAug vs none	2.989	Y	Y	Y
Event mask vs segment	6.800	Y	Y	Y
Segment mask vs none	19.875	Y	Y	Y
Event mask vs none	23.021	Y	Y	Y
Weak pseduolabels vs none	8.361	Y	Y	Y
Strong + weak vs weak	41.954	Y	Y	Y
Stong + weak vs none	36.011	Y	Y	Y

Table 4: Validation and test F1 scores for different pseudolabels.

est positive effect in adding easier samples first, but incorporating harder samples first does not have much benefit.

5.2. Effect of custom SpecAug procedure

Our best performing model is achieved by applying a custom time-masking SpecAug procedure randomly during training. We tried different values for the hyperparameter governing the length of the time masking $\gamma \in 0.25, 0.5, 1.0$. A final value of $\gamma = 0.25$ was fixed as the best time masking length. In the standard SpecAug experiment time masking was applied with fixed length of 16 time frames. For both procedures, we fixed the probability of applying SpecAug at $p = 0.5$ and apply 1 frequency mask of mask length 4 and 2 time masks. No time warping was applied in either procedure.

The effect of the SpecAug experiments are summarized in Table 2. We compared the performance of the overall top performing configuration (EF+SW+EM+CS, adding easier pseudolabels first) with applying standard SpecAug (EF+SW+EM+SS) and no data augmentation (EF+SW+EM+NS). CS, SS, and NS denote custom SpecAug, standard SpecAug, and no SpecAug, respectively. All other hyperparameters were kept fixed in the experiments, and each experiment is repeated to get ten trials with random initialization.

5.3. Effect of masking on the loss function

Additionally, we compared the effect of adding a segment and event masking matrix when computing the loss on strongly-labeled samples, as detailed in Eq 5. A comparison of ten trials for each masking type is shown in Table 2. The best results were achieved using event masking (EM), where only positive events were included in the calculation of the strong loss. Segment masking (SM) is noticeably worse than event masking but still performs much better than no masking (NM), suggesting that masking helps the student model learn which events are most important in the detection output, but focusing only on positive segments is too aggressive compared to simple event masking.

5.4. Effect of adding weak and strong pseudolabels

In our strongest model, both strong and weak predictions were used as pseudolabels on the unlabeled samples. The results are summarized in Table 3, where SW denotes adding both strong and weak pseudolabels and W denotes only adding weak pseudolabels. While adding weak pseudolabels does significantly boost the performance of the student model (EM+CS), the effect is the largest when comparing adding both strong and weak pseudolabels (EF+SW+EM+CS) with adding only weak pseudolabels (EF+W+EM+CS).

6. CONCLUSION

We have demonstrated that several techniques can be used to train a three-layer LSTM model on AED by using soft targets generated by a strong teacher model. In particular, progressively applying pseudolabeled samples, using variable-length time masking in SpecAug augmentation, and applying event masking to the loss function all contribute to a single model with a 42.7% macro event-based F1-score on the test set, matching state of the art performance of 42.7%. For each of the techniques, we perform a t-test on the means of the validation F1 score of two independent samples to test the statistical significance, summarized in Table 4. We find that adding easier samples first in the pseudolabeled dataset is statistically significant at the $\alpha = 0.2$ level, while the other techniques are significant at the $\alpha = 0.05$ level.

These techniques can be applied to AED models outside the teacher-student training context and can be further studied in more detail. Adding pseudolabeled data in different generations can help models learn more difficult samples over time. Further research can fine-tune these techniques in making the task harder in later generations. For example, adjusting time-masking techniques for data augmentation can be helpful for tasks with events of different average time-scales.

7. REFERENCES

[1] Q. Jin, P. Schulam, S. Rawat, S. Burger, D. Ding, and F. Metze, "Event-based video retrieval using audio," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[2] R. Radhakrishnan, A. Divakaran, and A. Smaragdis, "Audio analysis for surveillance applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.*, 2005, pp. 158–161.

[3] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring activities of daily living in smart homes: Understanding human behavior," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 81–94, 2016.

[4] N. Turpault, R. Serizel, A. Parag Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, New York City, United States, Oct. 2019. [Online]. Available: https://hal.inria.fr/hal-02160855

[5] L. Lin and X. Wang, "Guided learning convolution system for dcase 2019 task 4," Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, Tech. Rep., June 2019.

[6] Z. Shi, "Hodgepodge: Sound event detection based on ensemble of semi-supervised learning methods," Fujitsu Research and Development Center, Beijing, China, Tech. Rep., June 2019.

[7] L. Delphin-Poulat and C. Plapous, "Mean teacher with data augmentation for dcase 2019 task 4," Orange Labs Lannion, France, Tech. Rep., June 2019.

[8] D. S. Park, Y. Zhang, Y. Jia, W. Han, C.-C. Chiu, B. Li, Y. Wu, and Q. V. Le, "Improved noisy student training for automatic speech recognition," *Interspeech 2020*, Oct 2020. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-1470

[9] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech 2019*, Sep 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680

[10] T. K. Chan, C. S. Chin, and Y. Li, "Non-negative matrix factorization-convolutional neural network (NMF-CNN) for sound event detection," *CoRR*, vol. abs/2001.07874, 2020. [Online]. Available: https://arxiv.org/abs/2001.07874

[11] L. Lin, X. Wang, H. Liu, and Y. Qian, "Disentangled feature for weakly supervised multi-class sound event detection," *CoRR*, vol. abs/1905.10091, 2019. [Online]. Available: http://arxiv.org/abs/1905.10091

[12] X. Zheng, Y. Song, J. Yan, L.-R. Dai, I. McLoughlin, and L. Liu, "An effective perturbation based semi-supervised learning method for sound event detection." in *INTERSPEECH*, 2020, pp. 841–845.

[13] S. Adavanne, H. Fayek, and V. Tourbabin, "Sound event classification and detection with weakly labeled data," 2019.

[14] L. JiaKai, "Mean teacher convolution system for dcase 2018 task 4," DCASE2018 Challenge, Tech. Rep., September 2018.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[16] R. Shi, R. W. M. Ng, and P. Swietojanski, "Teacher-student training for acoustic event detection using audioset," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 875–879.

[17] K. Koutini, H. Eghbal-zadeh, and G. Widmer, "Iterative knowledge distillation in r-cnns for weakly-labeled semi-supervised sound event detection," DCASE2018 Challenge, Tech. Rep., September 2018.

[18] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1195–1204.

[19] L. Cances, T. Pellegrini, and P. Guyot, "Multi task learning and post processing optimization for sound event detection," IRIT, Université de Toulouse, CNRS, Toulouse, France, Tech. Rep., June 2019.

[20] H. Phan, L. Pham, P. Koch, N. Duong, I. McLoughlin, and A. Mertins, "Audio event detection and localization with multitask regression network," DCASE2020 Challenge, Tech. Rep., July 2020.

[21] W. Lim, "Specaugment for sound event detection in domestic environments using ensemble of convolutional recurrent neural networks," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, October 2019, pp. 129–133.

[22] W. Wang, C.-C. Kao, and C. Wang, "A simple model for detection of rare sound events," in *Proc. Interspeech 2018*, 2018, pp. 1344–1348. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-2338

[23] Y. Cao, Q. Kong, T. Iqbal, F. An, W. Wang, and M. D. Plumbley, "Polyphonic sound event detection and localization using a two-stage strategy," *arXiv preprint arXiv:1905.00268*, 2019.

[24] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML '09*, 2009.

[25] R. Lotfian and C. Busso, "Curriculum learning for speech emotion recognition from crowdsourced labels," vol. 27, no. 4, 2019.

[26] C. Wang, Y. Wu, S. Liu, M. Zhou, and Z. Yang, "Curriculum pre-training for end-to-end speech translation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 3728–3738. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.344

[27] N. Tonami, K. Imoto, Y. Okamoto, T. Fukumori, and Y. Yamashita, "Sound event detection based on curriculum learning considering learning difficulty of events," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 875–879.

DESCRIPTION AND DISCUSSION ON DCASE 2021 CHALLENGE TASK 2: UNSUPERVISED ANOMALOUS SOUND DETECTION FOR MACHINE CONDITION MONITORING UNDER DOMAIN SHIFTED CONDITIONS

Yohei Kawaguchi[1]*, Keisuke Imoto*[2]*, Yuma Koizumi*[3]*, Noboru Harada*[4]*, Daisuke Niizumi*[4]*, Kota Dohi*[1]*, Ryo Tanabe*[1]*, Harsh Purohit*[1]*, and Takashi Endo*[1]

[1] Hitachi, Ltd., Japan, yohei.kawaguchi.xk@hitachi.com
[2] Doshisha University, Japan, keisuke.imoto@ieee.org
[3] Google, Japan, koizumiyuma@google.com
[4] NTT Corporation, Japan, noboru.harada.pv@hco.ntt.co.jp

ABSTRACT

We present the task description and discussion on the results of the DCASE 2021 Challenge Task 2. In 2020, we organized an unsupervised anomalous sound detection (ASD) task, identifying whether a given sound was normal or anomalous without anomalous training data. In 2021, we organized an advanced unsupervised ASD task *under domain-shift conditions*, which focuses on the inevitable problem of the practical use of ASD systems. The main challenge of this task is to detect unknown anomalous sounds where the acoustic characteristics of the training and testing samples are different, i.e., domain-shifted. This problem frequently occurs due to changes in seasons, manufactured products, and/or environmental noise. We received 75 submissions from 26 teams, and several novel approaches have been developed in this challenge. On the basis of the analysis of the evaluation results, we found that there are two types of remarkable approaches that TOP-5 winning teams adopted: 1) ensemble approaches of "*outlier exposure*" (OE)-based detectors and "*inlier modeling*" (IM)-based detectors and 2) approaches based on IM-based detection for features learned in a machine-identification task.

Index Terms— anomaly detection, dataset, acoustic condition monitoring, domain shift, DCASE Challenge

1. INTRODUCTION

Anomalous sound detection (ASD) [1–7] is the task of identifying whether the sound emitted from a machine is normal or anomalous. Automatic detection of mechanical failure is an essential technology in the fourth industrial revolution, which includes artificial intelligence (AI)–based factory automation, and also prompt detection of machine anomalies by observing its sounds may be useful for machine condition monitoring.

We organized "*unsupervised ASD*" as Task 2 of the Detection and Classification of Acoustic Scenes and Events (DCASE) 2020 Challenge [8] for to connect academic tasks and real-world problems. The main challenge of this task was to detect unknown anomalous sounds under the condition that only normal sound samples have been provided as training data [1–7]. In real-world factories, actual anomalous sounds rarely occur but are highly diverse. Therefore, exhaustive patterns of anomalous sounds are impossible to collect. This means that we must detect unknown anomalous sounds that were not in the given training data. This unique and real-world oriented task attracted the interest of many participants, and resulted in 117 entries from 40 teams, which included several new approaches [9–12].

For the DCASE 2021 Challenge, we organized a follow-up unsupervised ASD task under domain-shift conditions, which simulates a more challenging issue in real-world applications. The main challenge of this task is that the acoustic characteristics of the training and testing phase are different due to changes in the normal condition such as motor speed and signal-to-noise ratio (SNR). A frequent real-world example of this problem is that the motor speed in a conveyor for transporting products varies in response to product demand; for a product whose demand changes with the seasons, training data recorded in the summer was 300–400 rotations per minute (RPM) (i.e. source domain), but the demand drops in the winter resulting in the motor speed decreasing to 100–200 RPM (i.e, target domain). Because a normal motor sound at 100 RPM is an unknown sound for the ASD system, it could incorrectly be detected as an anomalous sound. Therefore, methods to deal with such drift in normal conditions are required to accelerate the real-world application of ASD.

As the first benchmark task for domain-shift problems in ASD, we designed the DCASE Challenge 2021 Task 2 "*Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring under Domain-Shifted Conditions*." The scope includes differences in operating speed, machine load, environmental noise, and so on. After briefly introducing this task, we discuss remarkable approaches and their potential problems on the basis of the analysis of all 75 submissions from 26 teams.

2. UNSUPERVISED ANOMALOUS SOUND DETECTION UNDER DOMAIN-SHIFTED CONDITIONS

Let the L-sample time-domain observation $\boldsymbol{x} \in \mathbb{R}^L$ be an audio clip that includes a sound emitted from a machine. ASD is the determination of whether a machine is in a normal or anomalous state from $\boldsymbol{x}$. To determine the state of the machine, an anomaly score is calculated; it takes a large value when the machine is anomalous, and vice versa. To calculate the anomaly score, we have to prepare an anomaly score calculator $\mathcal{A}$ with parameter θ. The input of $\mathcal{A}$ is the audio clip $\boldsymbol{x}$ and additional information given by its file path, and one anomaly score $\mathcal{A}_\theta(\boldsymbol{x})$ is output. Then, the machine is determined to be anomalous when the anomaly score $\mathcal{A}_\theta(\boldsymbol{x})$ exceeds

a pre-defined threshold value ϕ as

$$\text{Decision} = \begin{cases} \text{Anomaly} & (\mathcal{A}_\theta(\boldsymbol{x}) > \phi) \\ \text{Normal} & (\text{otherwise}). \end{cases} \quad (1)$$

The primal difficulty in this task is to train $\mathcal{A}$ so that $\mathcal{A}_\theta(\boldsymbol{x})$ becomes a large value when the machine is anomalous, even though only normal sounds are available as training data.

In addition to the regular unsupervised ASD, we have to solve the domain-shift problem in real-world cases. As mentioned in Section 1, domain shifts refer to the difference in conditions between training and testing phases. The conditions differ in operating speed, machine load, viscosity, heating temperature, environmental noise, SNR, etc. The difference in conditions causes a gap in the sound characteristics, i.e., the distribution of the observation in the feature space changes. Here, two conditions are defined: **source domain** and **target domain**. The source domain refers to the original condition with a sufficient number of training clips, and the target domain refers to another state that has changed from the source domain. Let $\mathcal{D}_S$, $\mathcal{D}_T$, and $\mathcal{D}_{TA}$ be the distributions of $\boldsymbol{x}$ under the normal condition in the source domain, the normal condition in the target domain, and the anomalous condition in the target domain, respectively. The regular unsupervised ASD task is to determine whether $\boldsymbol{x}$ was generated from $\mathcal{D}_T$ or $\mathcal{D}_{TA}$ under the condition that clips from $\mathcal{D}_T (= \mathcal{D}_S)$ are available as training data, but clips from $\mathcal{D}_{TA}$ are not. On the other hand, in the domain shift scenario, detection must be performed under the condition that clips from $\mathcal{D}_T (\neq \mathcal{D}_S)$ are available as training data, but clips from $\mathcal{D}_{TA}$ are not. Note that only a few clips from $\mathcal{D}_T$ are provided as training data in 2021's task setting.

3. TASK SETUP

3.1. Dataset

The data used for this task comprises parts of the ToyADMOS2 [13] and MIMII DUE [14] datasets consisting of the normal/anomalous operating sounds of seven types of toy/real machines. We intentionally damaged machines to collect the anomalous sounds in these datasets. We provide the following types of machines: ToyCar and ToyTrain from ToyADMOS2, and fan, gearbox, pump, slide rail, and valve from MIMII DUE. To simplify the task, we use only the first channel of multichannel recordings; all recordings can be regarded as the single-channel recordings of a fixed microphone. Each recording is 10-sec-long audio that includes both the machine's operating sound and environmental noise. The sampling rate of all signals is 16 kHz. We mixed machine sounds with environmental noise, and only noisy recordings are available as training/test data. The environmental noise samples were recorded in several real factory environments. For the details of the recording procedure, please refer to the papers on ToyADMOS2 [13] and MIMII DUE [14].

In this task, we define two important terms: **machine type** and **section**.

Machine type refers to the type of machine, which can be one of seven in this task: fan, gearbox, pump, slide rail, ToyCar, ToyTrain, and valve.

Section is defined as a subset of the data within one machine type and consists of data from the source and target domains. A section is a unit for calculating performance metrics and is almost identical to "machine ID" in the 2020 version. In the 2020 version, there was a one-to-one correspondence between machine IDs and products, but in the 2021 version, machines of the same product appear in different sections (Sections 00–02 of the gearbox are the same product, and sections 03–04 of the gearbox are the same product.), and multiple products appear in the same section (Section 01 of the fan contains two products [14]).

We provide three datasets: **development dataset**, **additional training dataset**, and **evaluation dataset**.

Development dataset consists of three sections for each machine type (Sections 00, 01, and 02), and each section is a complete set of training and test data. For each section, this dataset provides (i) around 1,000 clips of normal sounds in a source domain for training, (ii) only three clips of normal sounds in a target domain for training, (iii) around 100 clips of both normal and anomalous sounds in the source domain for the test, and (iv) around 100 clips each of normal and anomalous sounds in the target domain for the test.

Additional training dataset provides the other three sections for each machine type (Sections 03, 04, and 05). Each section consists of (i) around 1,000 clips of normal sounds in a source domain for training and (ii) only three clips of normal sounds in a target domain for training.

Evaluation dataset provides test clips for the three sections (Sections 03, 04, and 05) identical to those in the additional training dataset. Each section consists of (i) test clips in the source domain and (ii) test clips in the target domain, none of which have a condition label (i.e., normal or anomaly). Note that the sections of the evaluation dataset (Sections 03, 04, and 05) are different from the development dataset (Sections 00, 01, and 02).

3.2. Evaluation metrics

The area under the curve (AUC) and partial-AUC (pAUC) for receiver operating characteristic (ROC) curves are used for evaluation as well as the 2020 edition [8]. The pAUC is an AUC calculated from a portion of the ROC curve over the pre-specified range of interest. In our metric, the pAUC is calculated as the AUC over a low false-positive-rate (FPR) range $[0, p]$. The AUC and pAUC for each machine type, section, and domain are defined as

$$\text{AUC}_{m,n,d} = \frac{1}{N_- N_+} \sum_{i=1}^{N_-} \sum_{j=1}^{N_+} \mathcal{H}(\mathcal{B}_{\theta,j,i}), \quad (2)$$

$$\text{pAUC}_{m,n,d} = \frac{1}{\lfloor pN_- \rfloor N_+} \sum_{i=1}^{\lfloor pN_- \rfloor} \sum_{j=1}^{N_+} \mathcal{H}(\mathcal{B}_{\theta,j,i}), \quad (3)$$

where $\mathcal{B}_{\theta,j,i} = \mathcal{A}_\theta(x_j^+) - \mathcal{A}_\theta(x_i^-)$, m represents the index of a machine type, n represents the index of a section, $d = \{\text{source}, \text{target}\}$ represents a domain, $\lfloor \cdot \rfloor$ is the flooring function, and $\mathcal{H}(x)$ returns 1 when $x > 0$ and 0 otherwise. $\{x_i^-\}_{i=1}^{N_-}$ and $\{x_j^+\}_{j=1}^{N_+}$ are normal and anomalous test clips in domain d in section n in machine type m, respectively, and they have been sorted so that their anomaly scores are in descending order. N_- and N_+ are the number of normal and anomalous test clips in domain d in section n in machine type m, respectively. The additional use of the pAUC is based on practical requirements. If an ASD system frequently gives false alarms, we cannot trust it. Therefore, it is important to increase the true-positive rate under low FPR conditions. In this task, we will use $p = 0.1$. The official score Ω for each submitted system is given by the harmonic mean of the AUC and pAUC

scores over all machine types, all sections, and both domains. As the aforementioned equations show, a threshold value does not need to be determined to calculate AUC, pAUC, or the official score because the threshold value is set to the anomaly score of a normal test clip.

3.3. Baseline systems and results

The task organizers provided two baseline systems.

Autoencoder-based baseline: The first baseline is an autoencoder (AE)-based anomaly score calculator and the same as the DCASE 2020 task 2. Details are described in the 2020 task description [8]. The anomaly score $\mathcal{A}_\theta$ is calculated as the mean square error of reconstruction for the observed sound. To obtain small anomaly scores for normal sounds, the AE is trained to minimize the reconstruction error of the normal training data. This method is based on the assumption that the AE cannot reconstruct sounds that are not used in training, that is, unknown anomalous sounds. If and only if A_θ for each test clip is greater than a threshold, the clip is judged to be anomalous.

"*Outlier exposure*"-based baseline using MobileNetV2: The second baseline is an anomaly score calculator obtained by using an approach called "*outlier exposure*" (OE) [15] for the machine-identification task. In DCASE 2020, 10 of the 40 teams used this approach [9, 11, 12, 16–22], and with four of them [9, 16, 19, 20] using the model of MobileNetV2 [23]. The models of this baseline are trained to identify from which section the observed signal was generated; it outputs the softmax value that is the predicted probability for each section. The anomaly score is calculated as the averaged negative logit of the predicted probabilities for the correct section. We first calculate the log-mel-spectrogram of the input $X = \{X_t\}_{t=1}^{T}$, where $X_t \in \mathbb{R}^F$, and F and T are the number of mel-filters and time-frames, respectively. Then, the acoustic feature (two-dimensional image) at t is obtained by concatenating consecutive frames of the log-mel-spectrogram as $\psi_t = (X_t, \cdots, X_{t+P-1}) \in \mathbb{R}^{P\times F}$. By shifting the context window by L frames, $B(= \lfloor \frac{T-P}{L} \rfloor)$ images are extracted. The frame size of the short-time Fourier transform (STFT) is 64 ms, and the hop size is 50 %. In addition, $F = 128$, $P = 64$, and $L = 8$. The Adam optimizer is used, and we fix the learning rate to 0.00001. We stop the training process after 20 epochs, and the batch size is 32. We train models independently for each machine type using normal clips from all sections of that machine type. The sections are used as classes to train the individual models. The anomaly score is calculated as:

$$\mathcal{A}_\theta(X) = \frac{1}{B}\sum_{b=1}^{B} \log\left(\frac{1 - p_\theta(\psi_{t(b)})}{p_\theta(\psi_{t(b)})}\right), \tag{4}$$

where $t(b)$ is the beginning frame index of the b-th image and p_θ is the softmax output by MobileNetV2 for the correct section.

Tables 1 and 2 show the AUC and pAUC scores for the two baselines, respectively. Because the results produced with a GPU are generally non-deterministic, the average and standard deviations from these five independent trials (training and testing) are shown.

Table 1: Results of the AE-based baseline

Section		AUC [%]		pAUC [%]	
		Source	Target	Source	Target
ToyCar	00	67.63 ± 1.21	54.50 ± 0.89	51.87 ± 0.50	50.52 ± 0.20
	01	61.97 ± 1.50	64.12 ± 1.07	51.82 ± 0.87	52.14 ± 0.80
	02	74.36 ± 0.82	56.57 ± 1.53	55.56 ± 0.83	52.61 ± 1.20
ToyTrain	00	72.67 ± 1.19	56.07 ± 0.80	69.38 ± 1.06	50.62 ± 0.68
	01	72.65 ± 0.32	51.13 ± 0.53	62.52 ± 0.88	48.60 ± 0.13
	02	69.91 ± 0.33	55.57 ± 1.07	47.48 ± 0.02	50.79 ± 0.93
Fan	00	66.69 ± 0.81	69.70 ± 0.32	57.08 ± 0.15	55.13 ± 0.34
	01	67.43 ± 1.12	49.99 ± 0.48	50.72 ± 0.42	48.49 ± 0.38
	02	64.21 ± 1.27	66.19 ± 1.23	53.12 ± 0.78	56.93 ± 1.37
Gearbox	00	56.03 ± 0.53	74.29 ± 0.51	51.59 ± 0.16	55.67 ± 0.97
	01	72.77 ± 0.72	72.12 ± 1.06	52.30 ± 0.18	51.78 ± 0.15
	02	58.96 ± 0.53	66.41 ± 0.72	51.82 ± 0.29	53.66 ± 0.57
Pump	00	67.48 ± 0.58	58.01 ± 0.57	61.83 ± 0.41	51.53 ± 0.27
	01	82.38 ± 0.27	47.35 ± 0.53	58.29 ± 0.77	49.65 ± 1.46
	02	63.93 ± 0.45	62.78 ± 0.70	55.44 ± 0.52	51.67 ± 0.35
Slide rail	00	74.09 ± 0.48	67.22 ± 0.45	52.45 ± 0.63	57.32 ± 0.52
	01	82.16 ± 0.35	66.94 ± 0.39	60.29 ± 0.30	53.08 ± 0.39
	02	78.34 ± 0.16	46.20 ± 0.77	65.16 ± 0.55	50.10 ± 0.31
Valve	00	50.34 ± 0.27	47.12 ± 0.18	50.82 ± 0.16	48.68 ± 0.09
	01	53.52 ± 0.33	56.39 ± 1.42	49.33 ± 0.10	53.88 ± 0.61
	02	59.91 ± 0.34	55.16 ± 0.22	51.96 ± 0.52	48.97 ± 0.04

Table 2: Results of the MobileNetV2-based baseline

Section		AUC [%]		pAUC [%]	
		Source	Target	Source	Target
ToyCar	00	66.56 ± 2.68	61.32 ± 5.94	66.47 ± 5.67	52.61 ± 2.41
	01	71.58 ± 5.54	72.48 ± 3.68	66.44 ± 2.84	63.99 ± 2.60
	02	40.37 ± 7.19	45.17 ± 3.36	47.48 ± 0.23	48.85 ± 0.94
ToyTrain	00	69.84 ± 4.39	46.28 ± 3.85	54.43 ± 1.65	51.27 ± 0.73
	01	64.79 ± 3.65	53.38 ± 2.47	54.09 ± 1.15	49.60 ± 0.88
	02	69.28 ± 6.73	51.42 ± 2.64	47.66 ± 0.40	53.40 ± 1.12
Fan	00	43.62 ± 2.35	53.34 ± 2.03	50.45 ± 1.15	56.01 ± 1.38
	01	78.33 ± 1.52	78.12 ± 4.25	78.37 ± 2.26	66.41 ± 7.16
	02	74.21 ± 3.85	60.35 ± 3.79	76.80 ± 0.78	60.97 ± 6.55
Gearbox	00	81.35 ± 1.59	75.02 ± 2.92	70.46 ± 3.67	64.77 ± 2.52
	01	60.74 ± 5.11	56.27 ± 8.27	53.88 ± 2.82	53.30 ± 2.97
	02	71.58 ± 7.16	64.45 ± 9.67	62.23 ± 6.67	55.58 ± 7.90
Pump	00	64.09 ± 4.34	59.09 ± 3.08	62.40 ± 1.90	53.96 ± 0.93
	01	86.27 ± 3.18	71.86 ± 5.97	66.66 ± 5.23	62.69 ± 2.33
	02	53.70 ± 4.99	50.16 ± 3.78	50.98 ± 1.23	51.69 ± 1.03
Slide rail	00	61.51 ± 4.92	51.96 ± 3.17	53.97 ± 2.03	51.96 ± 2.96
	01	79.97 ± 3.70	46.83 ± 10.65	55.62 ± 1.57	52.02 ± 4.17
	02	79.86 ± 1.41	55.61 ± 5.48	71.88 ± 4.64	55.71 ± 2.84
Valve	00	58.34 ± 4.01	52.19 ± 3.33	54.97 ± 4.43	51.54 ± 1.88
	01	53.57 ± 2.26	68.59 ± 2.84	50.09 ± 0.45	57.83 ± 2.49
	02	56.13 ± 1.96	53.58 ± 0.55	51.69 ± 0.32	50.86 ± 0.84

4. CHALLENGE RESULTS AND DISCUSSION

4.1. Results for evaluation dataset

We received 75 submissions from 26 teams, and 20 teams achieved better performance than the baseline systems. The harmonic means of the AUC scores of the top 10 teams [24–33] are shown in Fig. 1 for the source and target domains. As shown in the figure, the performance for which machine type is high or low varies greatly from team to team. However, the score on the target domain roughly correlates to the official ranking.

We find that there are two remarkable approaches in high-rank solutions: the first is an ensemble of OE-based detection and "*inlier modeling*" (IM)-based detection [24, 27, 28]. Here, IM refers to out-of-distribution (OOD) detection methods based on modeling a distribution of inlier samples, for example, AE, k-nearest neighbors (kNN), local outlier factor (LOF), Gaussian mixture models (GMM), normalizing flows (NF), interpolation deep neural network (IDNN) [6], and their conditional versions. The second approach is IM-based detection for features learned in a machine-identification task [25, 26]. We describe the details in the following sections.

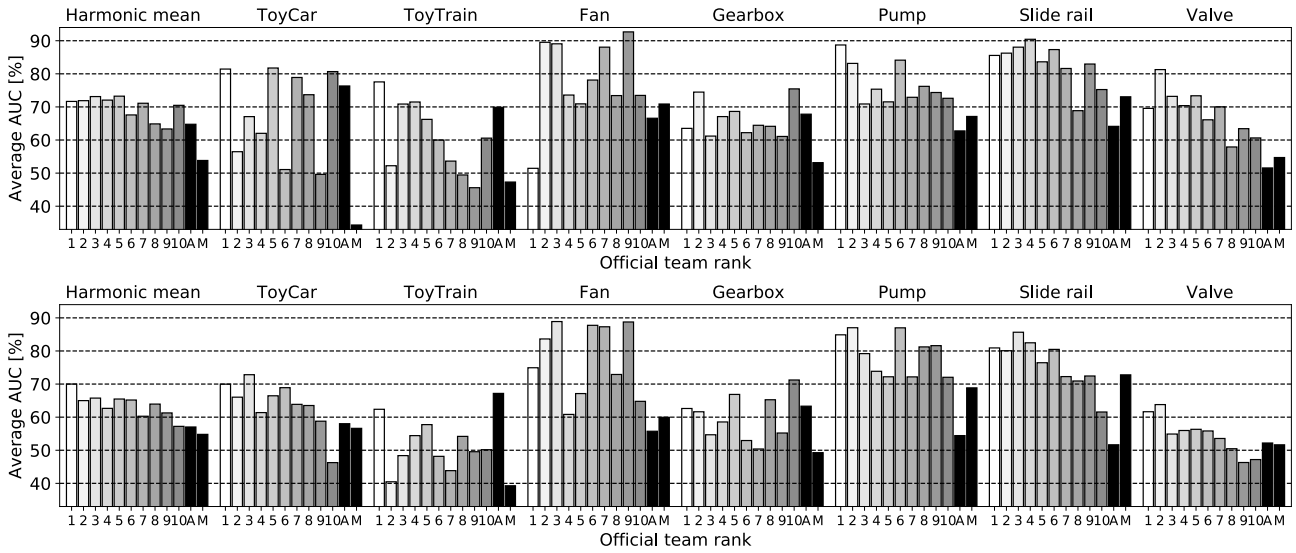


Figure 1: Evaluation results of top 10 teams in team ranking. AUC for source domains (top) and AUC for target domains (bottom). Label "A" and "M" on the x-axis means AE-based and MobileNetV2-based baselines, respectively.

4.2. Parallel-type hybrid approach: ensemble of OE-based and IM-based detectors

The first, fourth, and fifth-place teams [24, 27, 28] utilized this type of approach. OE-based detectors have a weakness in that their performance is severely degraded when the distributions of different sections (roughly, machine products) are too similar or too different [8, 34]; ensembles of OE and IM can reduce this OE weakness by leveraging the robustness of IM. The 2020 first-place team [9] used an ensemble of OE-based and IM-based detectors, using MobileNetV2 for OE and IDNN for IM. The 2021 first-place team [24] also used this type of approach, using multiple types of OE models and a conditional NF. Surprisingly, the technical report shows that this team did not perform any domain adaptation, but the performance of the target domain is outstanding. Taking into account the low performance of the individual subsystems of this team, we can guess that the ensemble gave them high generalization performance in both domains.

The fourth and fifth-place teams [27, 28] also took ensembles of OE and IM. However, unlike the first-place team, they prepared a model for each domain and performed domain adaptation, resulting in comparable AUC scores to the second and third-place teams in both domains. For example, the fifth-ranked team [28] is the only team to achieve AUC scores over 55% on all machine types in both domains. Although this ensemble-type approach tends to increase its model complexity, surprisingly, the model of the fifth-place team [28] is small. Further improving the performance of domain adaptation while maintaining the compactness of the model will continue to be a research problem.

4.3. Serial-type hybrid approach: IM-based detection for features learned in a machine-identification task

The second and third-place teams [25, 26] utilized this type of approach. They extracted features for the machine-identification task and performed IM-based detection on the extracted features. In training, the feature extraction model was first trained in the machine-identification task like OE-based methods, and then the IM-based detection model was trained. The aforementioned ensemble-type approach can be thought of as a parallel-type hybrid, whereas this approach can be thought of as a serial-type hybrid of OE and IM. This approach uses the powerful feature extraction of OE, but overcomes its aforementioned instability by taking advantage of the robustness of IM. In addition, this approach has the advantage of preventing the model complexity associated with ensembles.

Domain adaptation was performed only on IM-based detectors and not on feature extractors by the second and third-place teams [25, 26]. Such a domain adaptation method is less prone to overfitting because it fine-tunes only a limited range, and is considered effective when the number of training samples in the target domain is small. However, there is no guarantee that the features that are effective for machine identification will remain effective after the domain shift. In the future, it is desirable to verify how wide the effective range of this approach is and how far its performance for domain adaptation can be improved.

5. CONCLUSION

We presented an overview of the task and analysis of the solutions submitted to the DCASE 2021 Challenge Task 2. The main challenge of this task was to detect unknown anomalous sounds where the acoustic characteristics of the training and testing samples were different. We analyzed all evaluation results and submissions, and found that there are two types of remarkable approaches that TOP-5 winning teams adopted, i.e., 1) the parallel-type hybrid: ensemble approaches of OE-based and IM-based detectors and 2) the serial-type hybrid: approaches based on IM-based detection for features learned in the machine-identification task. Both two approaches are promising, but there is some room for performance improvement for domain adaptation. For the parallel-type hybrid, future work is to improve the performance of domain adaptation while maintaining the compactness of the model. Future work is needed for the serial-type hybrid to verify how wide this approach's effective range is and improve the domain adaptation performance.

6. REFERENCES

[1] Y. Koizumi, S. Saito, H. Uematsu, and N. Harada, "Optimizing acoustic feature extractor for anomalous sound detection based on Neyman-Pearson lemma," in *Proc. 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 698–702.

[2] Y. Kawaguchi and T. Endo, "How can we detect anomalies from subsampled audio signals?" in *Proc. 27th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017.

[3] Y. Koizumi, S. Saito, H. Uematsu, Y. Kawachi, and N. Harada, "Unsupervised detection of anomalous sound based on deep learning and the Neyman-Pearson lemma," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 212–224, Jan. 2019.

[4] Y. Kawaguchi, R. Tanabe, T. Endo, K. Ichige, and K. Hamada, "Anomaly detection based on an ensemble of dereverberation and anomalous sound extraction," in *Proc. 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 865–869.

[5] Y. Koizumi, S. Saito, M. Yamaguchi, S. Murata, and N. Harada, "Batch uniformization for minimizing maximum anomaly score of DNN-based anomaly detection in sounds," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019, pp. 6–10.

[6] K. Suefusa, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, "Anomalous sound detection based on interpolation deep neural network," in *Proc. 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 271–275.

[7] H. Purohit, R. Tanabe, T. Endo, K. Suefusa, Y. Nikaido, and Y. Kawaguchi, "Deep autoencoding GMM-based unsupervised anomaly detection in acoustic signals and its hyper-parameter optimization," in *Proc. 5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 175–179.

[8] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Proc. 5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 81–85.

[9] R. Giri, S. V. Tenneti, F. Cheng, K. Helwani, U. Isik, and A. Krishnaswamy, "Self-supervised classification for detecting anomalous sounds," in *Proc. 5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 46–50.

[10] S. Kapka, "ID-conditioned auto-encoder for unsupervised anomaly detection," in *Proc. 5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 71–75.

[11] P. Primus, V. Haunschmid, P. Praher, and G. Widmer, "Anomalous sound detection as a simple binary classification problem with careful selection of proxy outlier examples," in *Proc. 5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 170–174.

[12] T. Inoue, P. Vinayavekhin, S. Morikuni, S. Wang, T. H. Trong, D. Wood, M. Tatsubori, and R. Tachibana, "Detection of anomalous sounds for machine condition monitoring using classification confidence," in *Proc. 5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 66–70.

[13] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, "ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions," *arXiv preprint arXiv:2106.02369*, 2021.

[14] R. Tanabe, H. Purohit, K. Dohi, T. Endo, Y. Nikaido, T. Nakamura, and Y. Kawaguchi, "MIMII DUE: Sound dataset for malfunctioning industrial machine investigation and inspection with domain shifts due to changes in operational and environmental conditions," *arXiv preprint arXiv:2105.02702*, 2021.

[15] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," in *Proc. 7th International Conference for Learning Representations (ICLR)*, 2019.

[16] Q. Zhou, "ArcFace based sound MobileNets for DCASE 2020 task 2," DCASE2020 Challenge, Tech. Rep., 2020.

[17] J. Lopez, L. Hong, P. Lopez-Meyer, L. Nachman, G. Stemmer, and J. Huang, "A speaker recognition approach to anomaly detection," DCASE2020 Challenge, Tech. Rep., 2020.

[18] K. Wilkinghoff, "Anomalous sound detection with Look, Listen, and Learn embeddings," DCASE2020 Challenge, Tech. Rep., 2020.

[19] Z. Shinmura, "DCASE2020 task2 self-supervised learning solution," DCASE2020 Challenge, Tech. Rep., 2020.

[20] Q. Wei and Y. Liu, "Auto-encoder and metric-learning for anomalous sound detection task," DCASE2020 Challenge, Tech. Rep., 2020.

[21] F. Ahmed, P. Nguyen, and A. Courville, "An ensemble approach for detecting machine failure from sound," DCASE2020 Challenge, Tech. Rep., 2020.

[22] Y. Xiao, "Unsupervised detection of anomalous sounds technical report," DCASE2020 Challenge, Tech. Rep., 2020.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[24] J. Lopez, G. Stemmer, and P. Lopez-Meyer, "Ensemble of complementary anomaly detectors under domain shifted conditions," DCASE2021 Challenge, Tech. Rep., 2021.

[25] K. Morita, T. Yano, and K. Tran, "Anomalous sound detection using CNN-based features by self supervised learning," DCASE2021 Challenge, Tech. Rep., 2021.

[26] K. Wilkinghoff, "Utilizing sub-cluster AdaCos for anomalous sound detection under domain shifted conditions," DCASE2021 Challenge, Tech. Rep., 2021.

[27] I. Kuroyanagi, T. Hayashi, Y. Adachi, T. Yoshimura, K. Takeda, and T. Toda, "Anomalous sound detection with ensemble of autoencoder and binary classification approaches," DCASE2021 Challenge, Tech. Rep., 2021.

[28] Y. Sakamoto and N. Miyamoto, "Combine Mahalanobis distance, interpolation auto encoder and classification approach for anomaly detection," DCASE2021 Challenge, Tech. Rep., 2021.

[29] Q. Zhou, "Ensemble of ArcFace based systems for unsupervised anomalous sound detection under domain shift conditions," DCASE2021 Challenge, Tech. Rep., 2021.

[30] Y. Wang, Y. Zheng, Y. Zhang, and L. He, "Several approaches for anomaly detection from sound," DCASE2021 Challenge, Tech. Rep., 2021.

[31] J. Tozicka, D. Karel, and L. Michal, "Unsupervised anomalous sound detection by Siamese network and auto-encoder," DCASE2021 Challenge, Tech. Rep., 2021.

[32] X. Cai, H. Dinkel, Z. Yan, Y. Wang, J. Zhang, and Y. Wang, "The small rice camera ready submission to the dcase2021: Semi-supervised anomaly detection using contrastive learning," DCASE2021 Challenge, Tech. Rep., 2021.

[33] H. Narita and A. Tamamori, "Unsupervised anomalous sound detection using intermediate representation of trained models and metric learning based variational autoencoder," DCASE2021 Challenge, Tech. Rep., 2021.

[34] K. Dohi, T. Endo, H. Purohit, R. Tanabe, and Y. Kawaguchi, "Flow-based self-supervised density estimation for anomalous sound detection," in *Proc. 46th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 336–340.

MONYC: MUSIC OF NEW YORK CITY DATASET

Magdalena Fuentes[1]*, Danielle Zhao*[1]*, Vincent Lostanlen*[2]*,*
Mark Cartwright[3]*, Charlie Mydlarz*[1]*, Juan Pablo Bello*[1]*,*

[1] New York University, New York, NY, USA
[2] CNRS, Laboratoire des Sciences du Numérique de Nantes (LS2N), France
[3] New Jersey Institute of Technology, New Jersey, NY, USA

ABSTRACT

Music plays an important role in human cultures and constitutes an integral part of urban soundscapes. In order to make sense of these soundscapes, machine listening models should be able to detect and classify street music. Yet, the lack of well-curated resources for training and evaluating these models currently hinders their development. We present MONYC, an open dataset of 1.5k music clips as recorded by the sensors of the Sounds of New York City (SONYC) project. MONYC contains audio data and spatiotemporal metadata, i.e., coarse sensor location and timestamps. In addition, we provide multilabel genre tags from four annotators as well as four binary tags: whether the music is live or recorded; loud or quiet; single-instrument or multi-instrument; and whether non-musical sources are also present. The originality of MONYC is that it reveals how music manifests itself in a real-world setting among social interactions in an urban context. We perform a detailed qualitative analysis of MONYC, show its spatiotemporal trends, and discuss the scope of research questions that it can answer in the future.

Index Terms— Audio databases, environmental music, sound event detection, spatiotemporal context, street music, urban sound

1. INTRODUCTION

Although music is a fundamental component of urban life, our understanding of the musical soundscape of cities remains limited. Few prior sources address the detection of music in noisy environments [1, 2, 3], and even fewer the retrieval of music information. It is currently impossible to automate the indexation of street music, whether recorded by sensor networks or by smartphones. Such a limitation results from the lack of resources for training and evaluating dedicated machine listening models.

UrbanSound8k [2], now regarded as a standard benchmark for audio classifiers, was first in introducing a *street music* class as part of its taxonomy with the aim of detecting the presence of music. More recently, the SONYC-UST dataset [1] included *music* as one of the categories of its taxonomy. Yet, the musical samples in both of these datasets do not have detailed annotations; did not preserve the spatiotemporal distribution of street music; and were not representative in terms of acoustic content. In recent years, the field of audio event recognition has shifted its benchmarks to larger datasets, notably AudioSet [4] (derivative of YouTube) and FSD50k [5] (derivative of FreeSound). Although these datasets contain millions of audio clips, neither of them accommodates a *street music* category or provides any details about urban musical content. As a consequence, AudioSet-based classifiers such as YAMNet[1] cannot reliably identify urban music samples from YouTube or FreeSound, let alone from an acoustic sensor network. To the best of our knowledge, there is no open-source dataset for environmental music analysis that allows for developing models for understanding noisy music urban recordings.

The SONYC project [6] has recorded more than 700k hours of audio data from the streets of New York City. This data tells us a lot about the city's dynamics. The sounds of the city reflect its rhythm, and the major events that happened in the last years. A big part of the soundscape of the city is the music people listen to. People use music to manifest ideas or promote activities. Nightlife, festivals, social demonstrations, street celebrations, restaurants, bars and shops usually play music. Even music played loudly from the speakers of a car are manifestations of people's behaviour. Understanding music in an urban context gives us a deeper perspective on human behaviour, which is harder to obtain from other environmental sound events. From an acoustic viewpoint, the music recordings present in SONYC's archive are recorded in open spaces, in day-to-day conditions, and differ tremendously from commercial studio-recordings or artistic, close-field street recordings such as non-professional music videos. Recordings have low Signal-to-Noise Ratio (SNR), are picked up by the sensors at far-field distances (ranging from approximately ten to fifty feet), and they present big differences in their acoustic characteristics due to the different locations of the sensors: some are located in parks, some in commercial districts; some sensors have buildings close by, and others face towards open spaces. Additionally, the recordings have variant levels of noise from other sources present in the streets such as cars or people talking.

We present MONYC, a manually-annotated dataset of 10-second music clips recorded from the sensors of the SONYC project in the streets of New York City. MONYC was created using a combination of urban sound tagging; self-supervised learning; point process modeling; and human labeling. It conveys very rich metadata including timestamps and spatial location of clips, along with binary scene descriptors to assess models in different conditions (e.g. high interference of non-musical sources). By framing the detection and classification of musical events in the context of environmental acoustics, our goal is not to advance the state of the art in music technology; but rather, to discover relational, spatiotemporal, and behavioral trends in urban sounds at large.

2. DATA COLLECTION AND CURATION

As shown in Table 1, MONYC proceeds from SONYC by several stages of data filtering: from 250M audio clips acquired by SONYC sensors to 1.5k after agreement by multiple annotators. This section summarizes the process of data curation which led to MONYC.

Audio acquisition with SONYC sensors: SONYC sensors are

[1] https://www.tensorflow.org/hub/tutorials/yamnet

stage	SONYC	2017	subsampled	15 sensors	music	uniform	DPP	annotation	agreement
# clips	250M	30M	10M	5.8M	94k	30k	3k	1.7k	1.5k

Table 1: Number of clips at different stages of curation of MONYC.

placed on second-storey window ledges, at a typical height of seven meters [7]. They record street sounds under the form of 10-second audio clips at a sample rate of 48 kHz. For privacy reasons, the acquisition schedule of audio clips is randomized, with three clips per minute on average. Since May 2016, SONYC has acquired over 250M audio clips, making it one of the largest datasets of urban sounds worldwide.

Spatiotemporal filtering: We restrict our study to the year 2017 since it is the first year of the SONYC archive with complete data for the entire period, yielding 30M clips from 26 sensors. We subsample these 30M clips in time down to one clip per minute, yielding 10M clips. Then, we manually select 15 of the 26 sensors in diverse locations: e.g., on a small road, on a main avenue, in different corners of a park, next to a concert venue. Most of these sensors are located in Lower Manhattan, nearby Washington Square Park (WSP) and 5th Avenue; with a few other near Central Park and in downtown Brooklyn. Note that WSP is a well-known spot for street musicians while downtown Brooklyn hosts weekly outdoor concerts in the summer. Reducing the number of sensors from 26 to 15 yields 5.8M clips.

Urban sound tagging: We now proceed to retrieve street music within these 5.8M unlabeled clips. To this end, we run a deep learning model for urban sound tagging (UST) named SONYC-UST. SONYC-UST was trained on an open dataset of 19k SONYC clips from years 2016–2019 [1]. This dataset set was annotated by citizen scientists[2] and part of it was verified by experts. Among the 23 classes of the SONYC-UST taxonomy, three of them form the coarse category *music*. The SONYC-UST model has served as the baseline system for Task 5 of the DCASE 2020 challenge[3]. SONYC-UST does not take raw audio as input but relies on a pretrained feature extractor: Open-L^3 [8]. Open-L^3 is a deep convolutional network which was trained in a self-supervised way on unlabeled YouTube videos [8]. It is based on a mel-frequency spectrogram representation and produces 128-dimensional embeddings at a rate of 1 Hz, i.e. 10 embeddings per clip. SONYC-UST consists of two convolutional layers with a receptive field of 1 and ReLU nonlinearities, followed by AutoPool [9] to aggregate the 10 frame-level predictions.

We keep all clips where the model's output likelihood of *music* was above a threshold that corresponds to the validation set of [1]. This threshold is relatively low since at this stage we prioritized not discarding music clips that might be interesting but the model might not be confident with. At this point, we are down to 94k clips potentially containing street music.

Uniform spatiotemporal sampling: We compute the distribution of number of clips through the whole year for each sensor as a reference of the seasonal patterns spotted in that sensor. We keep this distribution as we downsize the number of clips for annotation in the following stage. Meanwhile, we select 2k samples per year per sensor at random (respecting the monthly distribution of music recordings), hence a total of 30k clips.

Diverse sampling with determinantal point processes: For our final sample we want not only to keep this seasonal distribution but to have as much diversity of acoustic conditions (e.g. instrumentation, genre, recording conditions) as we can within each sensor. For this, we use a determinantal point process (DPP).

Formerly known as fermionic processes, determinantal point processes (DPPs) are probabilistic models which initially arose in quantum physics to represent repulsive interactions between particles [10]. DPPs has gained attention in machine learning research [11] over the past decade, with the overarching goal of modeling the relative diversity of all possible subsets of a dataset. Since then, it has found many applications, e.g., text summarization [12], video recommendation [13], and news threading [14].

To curate the MONYC dataset, we consider a modified form of DPP known as K-DPP [15]. The key idea behind K-DPPs is to select a subset of K items from a larger collection Ω while striking a tradeoff between relevance and diversity. We use the DPP implementation from the DPPy package [16]. We use OpenL3 embeddings as the representation and the music likelihood output of the SONYC-UST model as relevance. We down-sampled each month of data per sensor following the yearly distribution explained below, for a total of 200 samples per sensor, and 3k samples for all sensors.

Annotations: Once we had the 3k music clips from the data-driven sampling, we performed the the manual annotation in four stages: 1) pre-selection of musical recordings by one annotator, 2) confirmation by three other annotators, 3) detailed annotation by the four annotators, and 4) annotation agreement and conflict solving. Annotating urban sound recording is a particularly hard endeavor, and street music clips are no exception. Many clips present low signal-to-noise ratio and sometimes music is faint. Other times it is distant and the interference of other sources makes it difficult to disambiguate if there is music or what instruments are present. Besides, the clips' duration of 10s is another challenging aspect, particularly for music annotations since the music can be captured at the least identifiable moment (e.g., an intro) making it hard to determine what music genre is being played. The four annotators are one student and three experienced machine listening researchers, all with musical training.

In a first stage, the student annotator curated all of the 3k clips. This annotator filtered out clips with no music or music too faint, leaving around 1.7k clips. Then each of the remaining annotators annotated one third of this data, with no overlap with each other. These annotators were asked again to confirm if there was music in the clip. Only clips where both annotators said there was music and no sensor faults were included in MONYC, for a total of 1587 clips.

Each annotator was then asked to provide for each clip multi-label free-form genre tags, and a set of binary indicators: whether the music is live or recorded, whether it is loud or quiet, whether the clip has a single instrument, and whether there is high interference from non-musical sources. To consider the uncertainty of annotating hard clips, the annotators used the label *unclear* for particularly hard examples. This free-form tagging was chosen to allow the discovery of music genres in urban music, and the process lead to a total of 114 tags, where the different annotators provided different level of granularity of sub-genres depending on their music preferences and knowledge. The annotators went then through a stage of agreement on a set of "sibling genres" that collapsed some of the annotated sub-

[2] https://www.zooniverse.org/projects/anaelisa24/sounds-of-new-york-city-sonyc

[3] https://github.com/sonyc-project/dcase2020task5-uststc-baseline

genres in a new set of agreed annotations. For instance, tags such as "ragtime", "bebop", "cool jazz", or "free jazz" were collapsed into "jazz". As many as 82 genres were collapsed to bigger categories, for a total of 41 genres in the agreed annotations taxonomy. After the agreement on this set of genres, recordings with conflicting annotations, either because annotators did not overlapped in any tag or they disagreed in the binary flags, were audited by at least two annotators to provide the final set of annotations for MONYC. One of the machine listening experts participated in all conflict solving to ensure consistency in the final annotations set. When there was no agreement among the two annotators, a third annotator was consulted. A total of 924 recordings were audited for agreement (58% of the total). Two sets of annotations are released: a set of agreed, audited annotations intended for developing machine listening models, and a set of pre-agreement annotations with more variance, to illustrate the difficulty and subjective nature of the data.

3. DATASET OVERVIEW

MONYC is an open source dataset of environmental music from the streets of New York City. The dataset taxonomy, all annotations and data are available online[4]. It consists of 1587 clips with manual annotations of multi-label free-form genre tags from four annotators, binary descriptors and non-exhaustive instrument annotations, as well as spatiotemporal metadata.

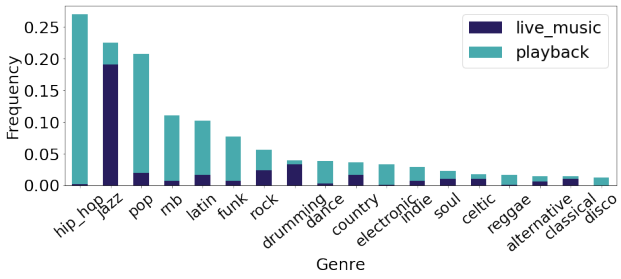


Figure 1: Distribution of 20 top genre labels.

Music genres in MONYC: The genre distribution of MONYC, as depicted in Figure 1, has several particularities. Firstly, unlike datasets such as AllMusic, Discorgs, Lastfm or Tagtraum (all part of AcousticBrainz [18]), where the genres with more appearances are rock or pop, the top genre in MONYC is hip hop, pop being the third one and rock just making it to the top ten. Looking at the binary indicator of whether the performance is live or playback, we can look at the relation between genre and live music in the context of street music. We see that most genres are playbacked, sometimes from cars passing, sometimes from shops, or speakers outside homes. The exception are two genres: jazz and drumming, which are both mostly live. Genres such as rock or country are more evenly spread between the two categories, being good candidates to asses the performance of models to identify live vs. playback music.

Spatiotemporal information: One of the unique features of MONYC is that each clip contains contextual information of where and when this clip was collected by the sensor network. To maintain privacy and following [1], we quantize the spatial information to the block level and the temporal information to the hour level. For the spatial information we provide borough and block identifiers,

[4]See https://magdalenafuentes.github.io/monyc/ and Soundata [17].

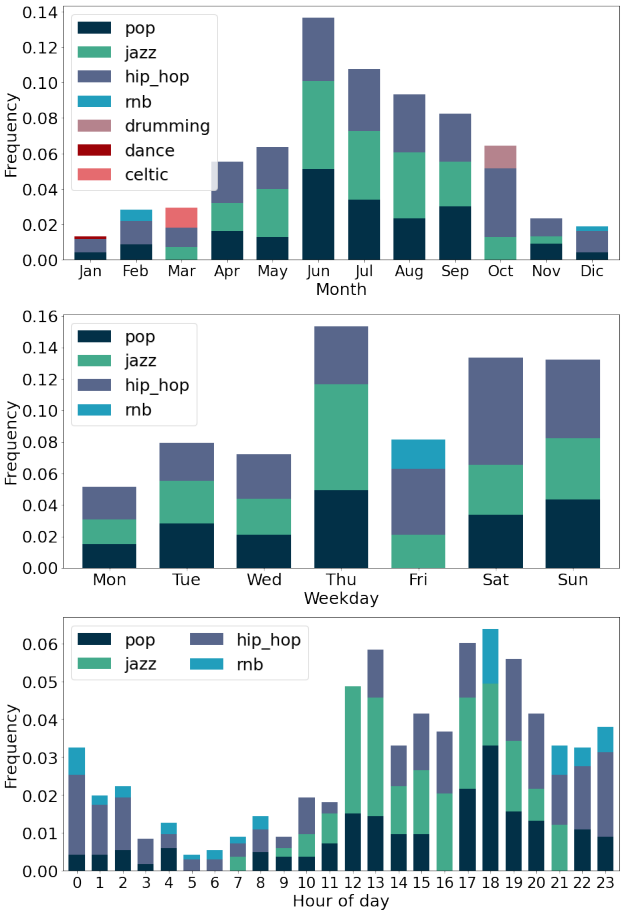


Figure 2: Temporal distribution of music clips at the month (top), weekday (middle) and hour (bottom) level.

as used in NYC's parcel number system known as Borough, Block, Lot (BBL) [19]. This is a common identifier used in NYC datasets, which facilitates the contrast of the sensor data to other open city data [20]. Figure 2 shows the temporal distribution of music clips at the month, day, and hour level broken down by the three top genres in each temporal scale. A first observation is that the amount of music clips increases towards the Summer months (June, July) an decreases considerably in Winter (November, December and January). This makes sense considering that the sensors are capturing environmental music, and in summer there are more concerts, more cars passing playing music with their windows open and more people in the street in general. The percentage of live music oscillates from less than 10% during Winter up to 35-40% in Summer, when it is at its highest. Music genres also change with seasons. As we saw previously, given its live nature, jazz has more presence in the streets in warmer months, with its peak being the summer. Hip hop and pop are season-less, being part of the City's music scene all year round. The appearance of other genres in the monthly top 3 is usually correlated with events that happened in those months. For example, the high presence of celtic music in March is highly explained by St. Patrick's day celebrations and parade on March 17th.

The weekday and hourly distributions also show interesting patterns. The first observation is that there is less street music at

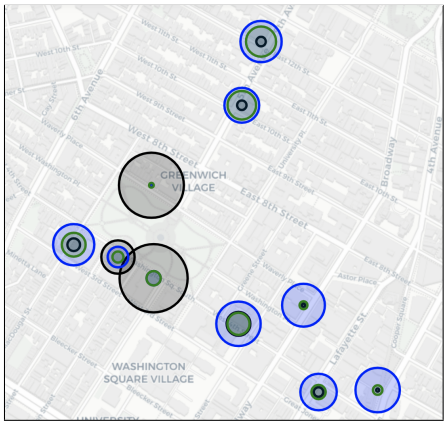


Figure 3: Spatial distribution of hip hop (blue), jazz (black) and pop (green). Circle sizes indicate number of recordings.

the beginning of the week, and an increase towards the weekend, with the exception of Thursdays which show a lot of music activity. When looking at this closer we noticed this is due to two sensors in Downtown Brooklyn which are facing towards a park, which in summer has live concerts every week on Thursdays[5], between noon and 2PM (which also explains the increase of music recordings around that time in the hourly distribution, with concerts having a big component of jazz as can be seen on the genre breakdown). Except for those two sensors, the rest follow a distribution that has more clips during weekends and late-afternoon/evenings. We included these two sensors in the dataset as they are a good example of how the location affect the observations in environmental music.

There are some genres that are constant through the week: pop, jazz and hip hop are present every day, but at different amounts per weekday. For instance, we observe that jazz has more presence on Thursdays, which can be related to the live concerts mentioned before. Hip hop is stable through the week with an increase towards the weekend. Rnb explains the increase of music clips on Fridays. More can be spotted by looking at e.g. the top five genres at different scales, we discussed the top three for better visualization.

An additional aspect of having the spatiotemporal data is that we can explore the spatial distribution of the different genres, i.e. where do we see more instances of one genre or another. An example of that is shown in Figure 3, which shows the spatial distribution of hip hop (blue), jazz (black) and pop (green). The map is zoomed in around Washington Square Park, where the density of sensors is the biggest for the SONYC network. Ten of the 15 sensors of MONYC are around the area of Greenwich Village. In the map we see the music events appearing in the same locations, which correspond to sensor deployments. The first observation is that hip hop and pop are more spread out across roads while jazz is more concentrated around the park. This makes sense considering that jazz is mostly live as in Figure 1, and it is being played in settings suitable for live performances such as a park. Hip hop and pop music are often played by passing cars or in gatherings outside homes, which agrees with the observations in the maps.

Music tagging in MONYC: We consider MONYC to be a challenging and interesting scenario to test tagging models, especially when focused on characterizing the music being played. This type of problem is not simply solved by using standard music taggers that where trained in high SNR recordings with no interfering sources, but requires models dedicated to environmental music. We include as a first example an experiment using the off-the-shelf music genre tagger *musicnn* [21], which is a convolutional neural network for out-of-the-box audio music tagging. We use the model trained with the Million Song Dataset (MSD) [22] since it is the one with bigger overlapping with MONYC's taxonomy out of the available models. We evaluate the model by computing the area under receiver operating characteristic curve (ROC-AUC) using the scikit-learn [23] implementation. We selected the subset of MONYC's clips that had genre annotations overlapping with the models' vocabulary, that is the 85% of the data. Twelve genres overlapped in MONYC's and MSD's taxonomies: *rock, pop, alternative, indie, dance, jazz, soul, electronica, folk, 90s, blues, hip hop, country, funk, and rnb*. The overall performance of the model in MONYC is considerably lower than in other datasets [24] (in the range of 90%), with a median ROC-AUC score of 50%. Figure 4 shows that the performance varies widely depending on the genre. Popular street genres such as hip hop, which are usually underrepresented when training music taggers, have very low performance. Looking at recordings from the three most common genres in the data (hip hop, jazz, pop), we noticed that the model performed 8-12% worse in average in those recordings labeled with high interference of sources. We hypothesize that this type of systematic error and the low overall performance could be corrected by re-training or fine tuning such models on MONYC data, which is out of the scope of this paper. This result presents a compelling first look at the type of errors such systems make in environmental music, and the steps we can now take towards making them more robust.

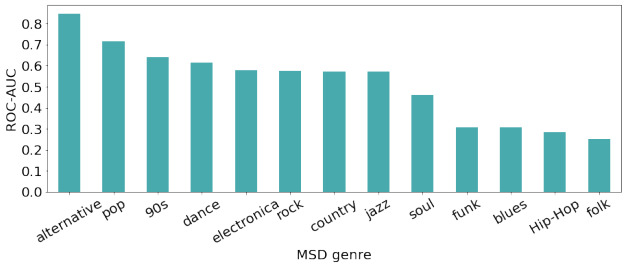


Figure 4: Median ROC tagging results of off-the-shelf music tagger breakdown per genre in MONYC.

4. CONCLUSIONS AND FUTURE WORK

We presented MONYC, the first-of-its-kind open dataset of music in urban settings. The dataset was created from the SONYC sensor network archive, delivering data-driven and self-supervised methods for sampling and curating a diverse set of music clips. It consists of a total of four hours of street music audio data along with highly rich annotations: multi-label genre tags from four annotators; spatiotemporal data consisting of location and timestamps of clips; and binary scene descriptors such as whether the music is live or recorded.

We hope this dataset provides the foundations for the development of machine listening models for environmental music, and we plan to expand the dataset with more recordings from the SONYC archive in the future by exploiting the current annotations, as well as similar data-driven methods.

[5]https://www.bam.org/media/9456156/Metrotech-2017_final.pdf

5. REFERENCES

[1] M. Cartwright, J. Cramer, A. E. M. Mendez, Y. Wang, H. Wu, V. Lostanlen, M. Fuentes, G. Dove, C. Mydlarz, J. Salamon, O. Nov, and J. P. Bello, "SONYC-UST-V2: An Urban Sound Tagging Dataset with Spatiotemporal Context," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, ser. DCASE, 2020.

[2] J. Salamon, C. Jacoby, and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1041–1044.

[3] E. Fonseca, J. Pons Puig, X. Favory, F. Font Corbera, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. Proceedings of the 18th ISMIR Conference; 2017 oct 23-27; Suzhou, China.[Canada]: International Society for Music Information Retrieval; 2017. p. 486-93.* International Society for Music Information Retrieval (ISMIR), 2017.

[4] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[5] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "FSD50K: an Open Dataset of Human-Labeled Sound Events," 2020.

[6] J. P. Bello, C. Silva, O. Nov, R. L. Dubois, A. Arora, J. Salamon, C. Mydlarz, and H. Doraiswamy, "Sonyc: A system for monitoring, analyzing, and mitigating urban noise pollution," *Communications of the ACM*, vol. 62, no. 2, pp. 68–77, 2019.

[7] C. Mydlarz, M. Sharma, Y. Lockerman, B. Steers, C. Silva, and J. P. Bello, "The Life of a New York City Noise Sensor Network," *Sensors*, vol. 19, no. 6, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/6/1415

[8] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3852–3856.

[9] B. McFee, J. Salamon, and J. P. Bello, "Adaptive Pooling Pperators for Weakly Labeled Sound Event Detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2180–2193, 2018.

[10] O. Macchi, "The Coincidence Approach to Stochastic Point Processes," *Advances in Applied Probability*, vol. 7, no. 1, pp. 83–122, 1975.

[11] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *Machine Learning*, vol. 5, no. 2-3, pp. 123–286, 2012.

[12] ——, "Learning Determinantal Point Processes," in *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 2011, pp. 419–427.

[13] M. Wilhelm, A. Ramanathan, A. Bonomo, S. Jain, E. H. Chi, and J. Gillenwater, "Practical Diversified Recommendations on YouTube with Determinantal Point Processes," in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2018, pp. 2165–2173.

[14] J. Gillenwater, A. Kulesza, and B. Taskar, "Discovering Diverse and Salient Threads in Document Collections," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*. Association for Computational Linguistics, 2012, pp. 710–720.

[15] A. Kulesza and B. Taskar, "k-DPPs: Fixed-size determinantal point processes," in *Proc. ICML*, 2011.

[16] G. Gautier, G. Polito, R. Bardenet, and M. Valko, "DPPy: DPP Sampling with Python," *Journal of Machine Learning Research - Machine Learning Open Source Software (JMLR-MLOSS)*, 2019, code at http://github.com/guilgautier/DPPy/ Documentation at http://dppy.readthedocs.io/. [Online]. Available: http://jmlr.org/papers/v20/19-179.html

[17] M. Fuentes, J. Salamon, P. Zinemanas, M. Rocamora, G. Plaja, I. R. Román, R. Bittner, M. Miron, X. Serra, and J. P. Bello, "Soundata: A Python library for reproducible use of audio datasets," *http://arxiv.org/abs/2109.12690*, 2021.

[18] D. Bogdanov, A. Porter, H. Schreiber, J. Urbano, and S. Oramas, "The AcousticBrainz Genre Dataset: Multi-source, Multi-level, Multi-label, and Large-scale," in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval (ISMIR 2019): 2019 Nov 4-8; Delft, The Netherlands.[Canada]: ISMIR; 2019.* International Society for Music Information Retrieval (ISMIR), 2019.

[19] "Borough, block, lot lookup." [Online]. Available: https://portal.311.nyc.gov/article/?kanumber=KA-01247

[20] "NYC Open Data." [Online]. Available: https://opendata.cityofnewyork.us/

[21] J. Pons and X. Serra, "musicnn: Pre-trained Convolutional Neural Networks for Music Audio Tagging," *arXiv preprint arXiv:1909.06654*, 2019.

[22] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," 2011.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of CNN-based Automatic Music Tagging Models," *arXiv preprint arXiv:2006.00751*, 2020.

CL4AC: A CONTRASTIVE LOSS FOR AUDIO CAPTIONING

Xubo Liu[1*], *Qiushi Huang*[2,3*], *Xinhao Mei*[1], *Tom Ko*[3], *H Lilian Tang*[2], *Mark D. Plumbley*[1], *Wenwu Wang*[1]

[1] Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, UK, {xubo.liu, x.mei, m.plumbley, w.wang}@surrey.ac.uk
[2] Department of Computer Science, University of Surrey, UK, {qiushi.huang, h.tang}@surrey.ac.uk
[3] Southern University of Science and Technology, Shenzhen, China, tomkocse@gmail.com

ABSTRACT

Automated Audio captioning (AAC) is a cross-modal translation task that aims to use natural language to describe the content of an audio clip. As shown in the submissions received for Task 6 of the DCASE 2021 Challenges, this problem has received increasing interest in the community. The existing AAC systems are usually based on an encoder-decoder architecture, where the audio signal is encoded into a latent representation, and aligned with its corresponding text descriptions, then a decoder is used to generate the captions. However, training of an AAC system often encounters the problem of data scarcity, which may lead to inaccurate representation and audio-text alignment. To address this problem, we propose a novel encoder-decoder framework called **C**ontrastive **L**oss for **A**udio **C**aptioning (CL4AC). In CL4AC, the self-supervision signals derived from the original audio-text paired data are used to exploit the correspondences between audio and texts by contrasting samples, which can improve the quality of latent representation and the alignment between audio and texts, while trained with limited data. Experiments are performed on the Clotho dataset to show the effectiveness of our proposed approach.

Index Terms— Audio captioning, cross-modal translation, contrastive loss, deep learning

1. INTRODUCTION

Automated Audio captioning (AAC) is a cross-modal translation task of generating a natural language description for an audio clip. It has various potential applications. For example, AAC can be used for generating subtitles for the audio content in a television program, or for generating text descriptions of audio to help the hearing impaired in accessing audio content. It can also be used by sound search engines to achieve more accurate retrieval and recommendation, or by a surveillance system to facilitate the detection of acoustic anomalies. The AAC problem has attracted increasing interest from the acoustic signal processing and machine learning communities in recent years.

Existing AAC systems are usually based on an encoder-decoder architecture [1, 2, 3, 4, 5]. The audio data is encoded into a latent representation and aligned with its corresponding text description. Then a decoder is used to generate the captions. Training of an AAC system often encounters the problem of data scarcity, which may lead to inaccurate representation and audio-text alignment. For example, Clotho [6] is a popular AAC dataset and was used for the DCASE challenge. However, it only contains 6974 audio samples, and each audio sample has five captions. To address this problem, information from keywords has been exploited for AAC [3, 7, 8]. The keywords of the caption are tagged firstly and then used to assist the generation of captions. However, due to the diversity of keywords, the tagging results of unseen audio clips may not be accurate in the inference stage. On the other hand, transfer learning techniques [9, 10] have been widely used in task 6 of the DCASE 2021 challenge, offering substantially improved performance. However, transfer learning relies heavily on large-scale external data [11] and pre-trained models [12].

Contrastive learning [13, 14] is a self-supervised paradigm that helps the model obtain high-quality representation. Inspired by the recent success of contrastive learning in computer vision (CV) [15] and natural language processing (NLP) [16, 17], we propose a novel encoder-decoder framework called **C**ontrastive **L**oss for **A**udio **C**aptioning (CL4AC). In CL4AC, the self-supervision signals derived from the original audio-text paired data are used to exploit the correspondences between audio and texts by contrasting samples. More precisely, we construct mismatched audio-text pairs as negative samples. Then, a contrastive learning objective is designed to maximize the difference between the representation of the matched audio-caption pair derived from the negative pairs. In this way, the quality of latent representation and the alignment between audio and texts can be improved without introducing large-scale external data, when they are trained with limited amount of data. To the best of our knowledge, contrastive learning approach has not been used for AAC in the literature.

The remainder of this paper are organised as follows. We introduce our proposed CL4AC in Section 2. Experiments are described in Section 3. Results are shown in Section 4. Finally, we conclude our work and discuss the future work in Section 5. The code of this work is made available on GitHub[1].

2. CONTRASTIVE LOSS FOR AUDIO CAPTIONING

In this section, we present our proposed contrastive learning framework for audio captioning (CL4AC). We first introduce the encoder-decoder architecture of CL4AC in Section 2.1. Then, we present the contrastive learning framework in Section 2.2.

2.1. Encoder-Decoder architecture

We first define the notations used in this section. The training data for AAC consists of paired audio and texts data. We denote a training set of N audio-text pairs by $D = \{(a_n, C_n)\}_{n=1}^{N}$, where $a \in \mathbb{R}^{H \times W}$

*The first two authors contributed equally to this work.

[1] https://github.com/liuxubo717/cl4ac

is the log mel-spectrogram of an audio clip with H and W being its height and width, respectively, $C = \{w_m\}_{m=1}^{M}$ is the token sequence of a caption where w_m is the m-th token in the caption C having M tokens, a_n is the log mel-spectrogram of the n-th audio clip in the dataset, and C_n is the token sequence of the n-th caption in the dataset.

The sequence-to-sequence architecture with Convolutional Neural Network (CNN) encoder and Transformer decoder are used as the basis of our proposed framework, as shown in Figure 1. This architecture was shown to offer the state-of-the-art performance [9, 10] in Task 6 of the DCASE 2021 challenge.

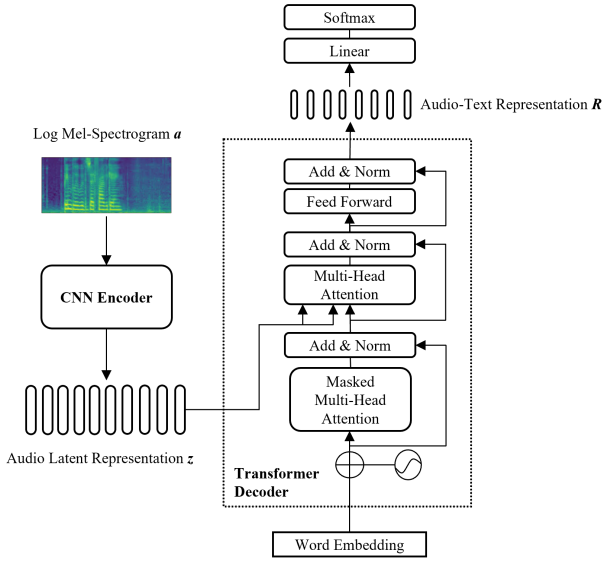


Figure 1: Sequence-to-sequence architecture with CNN encoder and Transformer decoder for audio captioning. The components in the dashed box indicate the Transformer decoder.

2.1.1. CNN encoder

Pre-trained audio neural networks (PANNs) [12] have demonstrated a powerful ability in extracting latent representation of audio signals for different downstream audio recognition tasks. To benefit from its high-quality audio representation, we choose PANNs as the encoder, which will be described in Section 3.3 in details. The PANNs encoder takes the log mel-spectrogram a of an audio clip as the input and extracts its latent representation $z \in \mathbb{R}^{H' \times W'}$. Formally:

$$z = \text{Encoder}(a). \tag{1}$$

2.1.2. Transformer decoder

The Transformer model has shown the state-of-the-art performance on language-related cross-modal task [18, 19], and is used as the decoder in our work. There are two main components in the decoder. Firstly, each token w_m in the input token sequence C is converted into a word embedding $e_m \in \mathbb{R}^{1 \times E}$, where E is the dimension of the word embedding, by the word2vec algorithm using Continuous Bag of Words Model (CBOW) [20] and Skip-Gram [21] model trained purely on the caption corpus. Then the word embedding of tokens are fed into the first self-attention layer to obtain their hidden states. The latent representation z of an audio clip extracted by the encoder is aligned and calculated with the hidden states of tokens, then the audio-text representation is obtained by the transformer decoder, denoted as $R \in \mathbb{R}^{M \times T}$, which consists of M vectors $\{r_m\}_{m=1}^{M}$, where the number of vectors is equal to the length of the input token sequence C and the dimension of each vector is T. The vector r_m of the audio-text representation R is calculated based on the word embeddings $\{e_1, ..., e_{m-1}\}$ and the audio latent representation. Hence, each r_m corresponds to the token w_m in the input token sequence C one-to-one, which can be used to predict the probability of the word over the vocabulary after it is passed through the final linear layer with softmax function. The transformer decoder predicts the m-th word w_m based on the previous tokens $\{w_1, ..., w_{m-1}\}$ and the audio latent representation z, as follows,

$$p(w_m|z, w_1, ..., w_{m-1}) = \text{Decoder}(z, w_1, ..., w_{m-1}). \tag{2}$$

The training objective is to optimize the cross entropy (CE) loss defined in terms of the predicted words as:

$$\text{Loss}_{\text{CE}} = -\mathbb{E}_{(a,C)\sim D} \log p(w_m|z, w_1, ..., w_{m-1}). \tag{3}$$

2.2. Contrastive learning framework

To obtain accurate audio-text representation R while the model is trained with limited data, we use the self-supervised signal derived from the audio-text training data by contrasting samples. First, we construct mismatched audio-text pairs as negative samples. Then, a contrasting auxiliary task is designed to maximize the difference between the representation R of the matched audio-text pair derived from negative pairs. The representations of the audio-text paired data are pulled together in the latent space while simultaneously pushing apart clusters of unpaired negative data by contrastive learning, as shown in Figure 2. In this way, the quality of audio-text representation and the alignment between audio and texts can be improved.

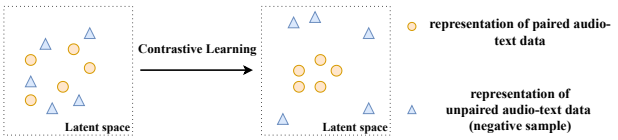


Figure 2: The representations of the audio-text paired data are pulled together in the latent space while simultaneously pushing apart clusters of unpaired negative data by Contrastive Learning (CL).

More specifically, for each anchor audio-text paired training data $x = (a, C)$, we replace the caption C by $C_{negative}$ which is a randomly selected caption unpaired with a in the training set D. Then, the mismatched audio-text pair as the negative training sample is constructed, denoted as $x_{negative} = (a, C_{negative})$. Table 1 shows the examples of x and $x_{negative}$ in the Clotho dataset. Since the last vector in the audio-text representation R is able to attend the context of all input tokens and the audio feature, the value of last vector of R is fed into a binary classifier $f(.)$ to predict whether the input audio and text data are paired ($y = 0$) or not ($y = 1$). The contrastive learning (CL) loss for this auxiliary task is defined as follows:

$$\text{Loss}_{\text{CL}} = -\mathbb{E}_{x' \sim D'} \log p(y|f(x')), \tag{4}$$

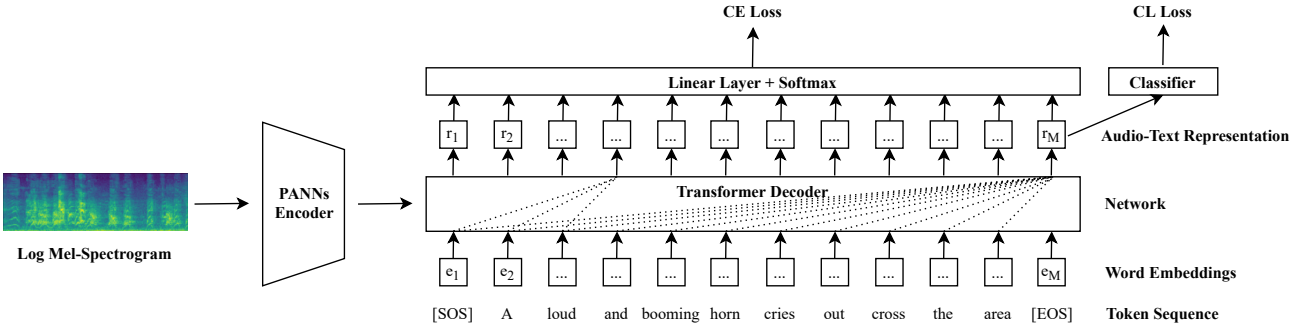


Figure 3: Contrastive loss for audio captioning (CL4AC) framework. The dashed lines indicate that the vector r_m of the audio-text representation R is calculated based on the word embeddings $\{e_1, ..., e_{m-1}\}$ and the audio latent representation obtained from PANNs. The last audio-text representation vector r_M is fed to the classifier $f(.)$ whose output is used to calculate the Contrastive Learning (CL) loss.

Example	paired caption C	unpaired caption $C_{negative}$
audio a	Something goes round that is playing its song	The Air is blowing some what fast outside
	At the fair, music is playing near a carousel through the speaker	A hand held sander was used as various speeds
	Chiming of bells, whistles and horns at a performance	A hard gravel ground is walked on by someone
	Fair kind music is being played at the circus grounds	A person using a hard object to tap and scrape glasses
	Polka or fair kind of music is being played	The wind is blowing and the waves are flowing

Table 1: Examples of paired audio-text training data $x = (a, C)$ and negative training sample $x_{negative} = (a, C_{negative})$. Examples are selected from the Clotho dataset, where each audio data has five corresponding captions.

.

where D' is the extended training set by merging the negative samples into the original training set D and x' is the audio-text pair drawn from D'. The full training objective of CL4AC is:

$$\text{Loss}_{\text{Training}} = (1 - y)\,\text{Loss}_{\text{CE}} + \text{Loss}_{\text{CL}}\,. \tag{5}$$

When the input is a negative audio-text pair, the gradient provided by the CE loss is meaningless, for this case, only CL loss is used for updating the model. The framework of CL4AC is shown in Figure 3.

3. EXPERIMENTS

3.1. Dataset

Clotho [6] is an AAC dataset whose sound clips are from the Freesound platform and annotated by Amazon Mechanical Turk. Clotho v2 was released for Task 6 of the DCASE 2021 Challenge, which contains 3839, 1045 and 1045 audio clips for the development, validation and evaluation split respectively. The sampling rate of all audio clips in Clotho dataset is 44 100 Hz. Each audio clip has five captions. Audio clips are of 15 to 30s duration and captions are eight to 20 words long. We merge the development and validation split, forming a new training set with 4884 audio clips. The performance of AAC system is evaluated on the evaluation split.

3.2. Data pre-processing

We use the original sampling rate to load audio data, and an 64-dimensional log mel-spectrogram is calculated using the short-time Fourier transform (STFT) with a frame size of 1024 samples, a hop size of 512 samples, and a Hanning window. SpecAugment [22] is used for data augmentation.

We transform all captions in the Clotho dataset to lower case with punctuation removed. Two special tokens "<sos>" and "<eos>" are added on the start and end of each caption. The vocabulary of the Clotho dataset contains 4367 words.

3.3. Model implementation

CNN-10 of PANNs [12] is used as the encoder to prevent over-fitting while trained with limited data. Specifically, the CNN-10 consists of four convolutional blocks where each has two convolutional layers with a kernel size of 3×3. Batch normalization and ReLU are used after each convolutional layer. The channels number of each block are 64, 128, 256 and 512, respectively. An average pooling layer with kernel size 2×2 is applied between them for down-sampling. Global average pooling is applied along the frequency axis after the last convolutional block followed by two fully connected layers to align the dimension of the output with the decoder input. Two transformer blocks with four heads and 128 hidden units are used as the decoder. The implementation for the encoder and decoder is the same as that in our DCASE 2021 Challenge system[2], which is the highest-scoring system without using model ensembles.

We trained the proposed model using Adam [23] optimizer with a batch size of 16. Warm-up is used in the first 5 epochs to increase the learning rate to the initial learning rate linearly. The learning rate is then decreased to $1/10$ of itself every 10 epochs. Dropout with a rate of 0.2 is applied in the proposed model to mitigate the over-fitting problem. We train the model for 30 epochs with an initial learning rate of 5×10^{-4} on the training set of the Clotho dataset.

[2] https://github.com/XinhaoMei/DCASE2021_task6_v2

Model	$BLEU_1$	$BLEU_2$	$BLEU_3$	$BLEU_4$	$ROUGE_L$	METERO	CIDEr	SPICE	SPIDEr
Baseline	0.550	0.345	0.222	0.139	0.372	0.169	0.356	0.115	0.235
CL4AC	0.553	0.349	0.226	0.143	0.374	0.168	0.368	0.115	0.242

Table 2: Performance of models is evaluated on the Clotho v2 evaluation set. Baseline: baseline system described in Section 3.4, which is similar to our DCASE submitted system but without transfer learning and reinforcement learning techniques. CL4AC: Proposed framework Contrastive Loss for Audio Captioning (CL4AC). During the inference stage, captions are generated using greedy search.

3.4. Baseline system

The baseline system is similar to our DCASE 2021 system which uses transfer learning (TL) from external dataset and reinforcement learning (RL) [9]. Our motivation is to mitigate the data scarcity problem for AAC without introducing external datasets, so we train the baseline without using the TL technique. Previous studies [24, 25] proved that although RL techniques can optimize neural networks towards non-differentiable metrics, they may generate syntactically incorrect and incomplete captions. Thus, RL is also removed in the baseline system. The hyper-parameters used for training the baseline system are similar to the proposed model (as described in Section 3.3), except that the training batch size is 32 and the initial learning rate is 1×10^{-3}.

3.5. Evaluation

During the inference stage, the mel-spectrogram of an audio clip along with the special token "<sos>" are fed into the encoder and decoder separately to generate the first token. Afterwards, the following tokens are predicted in terms of the previously generated tokens until the token "<eos>" or the maximum length (35 words in our experiments) is reached. The greedy search strategy is used to generate captions.

We evaluate the performance of the proposed framework using the same metrics adopted in Task 6 of the DCASE 2021 Challenge, including machine translation metrics: $BLEU_n$ [26], METEOR [27], $ROUGE_l$ [28] and captioning metrics: CIDEr [29], SPICE [30], SPIDEr [31]. $BLEU_n$ measures the quality of the generated text by calculating the precision of n-gram inside the text, which is an inexpensive metric to measure the correspondence between generated text and the ground truth. Generally, the higher $BLEU_n$ usually implies better precision and fluent text. The SPIDEr, a combination of SPICE and CIDEr, is designed for image captioning task measurement, which considers scene graph inside the generated caption and the term frequency-inverse document frequency (TF-IDF) of the n-gram. By considering the scene graph and the TF-IDF of n-gram, the metric will focus on the relationships among objects and the text's property, which ensures the semantic fidelity to the audio and the syntactical fluency of the language.

4. RESULTS

Table 2 shows the performance of our proposed method on the Clotho v2 evaluation set. By adopting the contrastive loss technique during the training process, all the metrics except METERO increased on the evaluation set. For $BLEU_1$, $BLEU_2$, $BLEU_3$, $BLEU_4$, the relative improvement percentages for contrastive loss are 0.55%, 1.16%, 1.80%, and 2.88%, respectively. The n in $BLEU_n$ means the n-grams matching between the predicted results and ground truths. The ascending increases of the relative improvement from $BLEU_1$ to $BLEU_4$ show that our proposed method generates more matching n-grams, demonstrating a more fluent and better quality captioning result. Besides, CIDEr and SPIDEr, the captioning metrics, obtained 3.37% and 2.98% relative improvement correspondingly. The better CIDEr and SPIDEr ensure the captions are better semantically faithful to the audio clip with the better language fluency. Numerical improvement of the machine translation and captioning metrics shows the effectiveness of CL4AC while trained with limited data.

5. CONCLUSIONS

This paper demonstrated the problem of data scarcity for AAC, which may lead to the inaccurate representation and audio-text alignment. To alleviate this issue, a novel encoder-decoder framework called **C**ontrastive **L**oss for **A**udio **C**aptioning (CL4AC) was proposed to learn a better cross-modal representation. In CL4AC, the self-supervision signals derived from the original audio-text data are used to exploit the correspondences between audio and text by contrasting samples in a limited dataset setting. Experiment results on $BELU_n$, CIDEr, and SPIDEr showed the effectiveness of the proposed approach with a relative improvement of up to 3.37%, compared to the baseline system. In future work, we will explore more contrastive learning approaches for AAC, such as Momentum Contrast (MoCo) [32] and SimCLR [15].

6. ACKNOWLEDGMENT

This work is partly supported by grant EP/T019751/1 from the Engineering and Physical Sciences Research Council (EPSRC), a Newton Institutional Links Award from the British Council, titled "Automated Captioning of Image and Audio for Visually and Hearing Impaired" (Grant number 623805725) and a Research Scholarship from the China Scholarship Council (CSC) No. 202006470010.

References

[1] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 374–378.

[2] K. Chen, Y. Wu, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on transformer and pre-trained cnn," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020). Tokyo, Japan*, 2020, pp. 21–25.

[3] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A transformer-based audio captioning model with keyword estimation," *arXiv preprint arXiv:2007.00222*, 2020.

[4] A. Tran, K. Drossos, and T. Virtanen, "WaveTransformer: A novel architecture for audio captioning based on learning temporal and time-frequency information," *arXiv preprint arXiv:2010.11098*, 2020.

[5] X. Mei, X. Liu, Q. Huang, M. D. Plumbley, and W. Wang, "Audio captioning transformer," *arXiv preprint arXiv:2107.09817*, 2021.

[6] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 736–740.

[7] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre-and post-processings for audio captioning," *arXiv preprint arXiv:2009.11436*, 2020.

[8] A. Ö. Eren and M. Sert, "Audio captioning based on combined audio and semantic embeddings," in *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2020, pp. 41–48.

[9] X. Mei, Q. Huang, X. Liu, G. Chen, J. Wu, Y. Wu, J. Zhao, S. Li, T. Ko, H. L. Tang, X. Shao, M. D. Plumbley, and W. Wang, "An encoder-decoder based audio captioning system with transfer and reinforcement learning for DCASE challenge 2021 task 6," DCASE2021 Challenge, Tech. Rep., July 2021.

[10] W. Yuan, Q. Han, D. Liu, X. Li, and Z. Yang, "The DCASE 2021 challenge task 6 system: Automated audio captioning with weakly supervised pre-traing and word selection methods," DCASE2021 Challenge, Tech. Rep., July 2021.

[11] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 119–132.

[12] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[13] A. V. D. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[14] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 18 661–18 673.

[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1597–1607.

[16] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," in *International Conference on Learning Representations*, 2021.

[17] Q. Huang, T. Ko, H. L. Tang, X. Liu, and B. Wu, "Token-level supervised contrastive learning for punctuation restoration," *arXiv preprint arXiv:2107.09099*, 2021.

[18] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, *et al.*, "Oscar: Object-semantics aligned pre-training for vision-language tasks," in *European Conference on Computer Vision*. Springer, 2020, pp. 121–137.

[19] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara, "Meshed-memory transformer for image captioning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 578–10 587.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[21] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 3111–3119.

[22] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] X. Mei, Q. Huang, X. Liu, G. Chen, J. Wu, Y. Wu, J. Zhao, S. Li, T. Ko, H. L. Tang, *et al.*, "An encoder-decoder based audio captioning system with transfer and reinforcement learning," *arXiv preprint arXiv:2108.02752*, 2021.

[25] Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan, "DialoGPT: Large-scale generative pre-training for conversational response generation," *arXiv preprint arXiv:1911.00536*, 2019.

[26] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[27] A. Lavie and A. Agarwal, "METEOR: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the Second Workshop on Statistical Machine Translation*, 2007, pp. 228–231.

[28] L. C. ROUGE, "A package for automatic evaluation of summaries," in *Proceedings of Workshop on Text Summarization of ACL, Spain*, 2004.

[29] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "CIDEr: Consensus-based image description evaluation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4566–4575.

[30] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "SPICE: Semantic propositional image caption evaluation," in *European Conference on Computer Vision*. Springer, 2016, pp. 382–398.

[31] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 873–881.

[32] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

ARCA23K: AN AUDIO DATASET FOR INVESTIGATING OPEN-SET LABEL NOISE

Turab Iqbal, Yin Cao, Andrew Bailey, Mark D. Plumbley, Wenwu Wang

Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, UK
{t.iqbal, yin.cao, andrew.bailey, m.plumbley, w.wang}@surrey.ac.uk

ABSTRACT

The availability of audio data on sound sharing platforms such as Freesound gives users access to large amounts of annotated audio. Utilising such data for training is becoming increasingly popular, but the problem of label noise that is often prevalent in such datasets requires further investigation. This paper introduces ARCA23K, an Automatically Retrieved and Curated Audio dataset comprised of over 23 000 labelled Freesound clips. Unlike past datasets such as FSDKaggle2018 and FSDnoisy18K, ARCA23K facilitates the study of label noise in a more controlled manner. We describe the entire process of creating the dataset such that it is fully reproducible, meaning researchers can extend our work with little effort. We show that the majority of labelling errors in ARCA23K are due to out-of-vocabulary audio clips, and we refer to this type of label noise as open-set label noise. Experiments are carried out in which we study the impact of label noise in terms of classification performance and representation learning.

Index Terms— Audio dataset, audio classification, label noise, machine learning, deep learning, neural networks

1. INTRODUCTION

Labelled audio data is a relatively scarce resource, yet it is vital for training audio classifiers in a supervised fashion. With the emergence of online sharing platforms such as Freesound [1] and YouTube [2], users now have access to massive amounts of annotated audio, and it is becoming increasingly popular to utilise this data for training. For classifying general sound events, early examples of web-sourced datasets include ESC-50 [3] and UrbanSound8K [4]. However, these datasets are relatively small, which is largely due to the high cost of manually verifying the data to ensure the sounds are relevant and the labels are correct. At the time of writing, the largest sound event dataset with thoroughly verified labels is FSD50K [5], which contains approximately 50 000 sounds and is the result of several years of crowdsourced labelling [1].

Given the cost of label verification, there has been interest in reducing or eliminating this aspect of dataset curation. AudioSet [2], for example, is a large-scale audio dataset comprised of over two million sounds across hundreds of classes. AudioSet classes belong to an ontology in which the classes share parent-child relationships. Although AudioSet clips have been manually verified by listeners, the process was not thorough, and many labelling errors remain [6]. The labels of other datasets, such as VGGSound [7], have not been verified at all. In the case of FSDKaggle2018 [8], FSDKaggle2019 [9], and FSDnoisy18k [10], only a small subset of the dataset has been manually verified. Nevertheless, these datasets are attractive because they are relatively large. The challenge is that the presence of labelling errors, or *label noise*, can significantly impact learning [10]. Hence, studying the effects of label noise is important.

Due to label noise, rather than the training examples being drawn from the true distribution, P, examples are drawn from a corrupted distribution, Q. In the literature, the noise process responsible for this corruption is typically assumed to be reversible, such that any incorrectly-labelled instance can be relabelled [11, 12]. This is *not* a realistic assumption when retrieving and labelling web data, as the sounds that are retrieved can be *out-of-vocabulary (OOV)* [10]. OOV sounds are sounds that do not belong to any of the classes of interest. We refer to this type of label noise as *open-set label noise*. There is currently a disconnect where much of the analysis and tools are for closed-set label noise, while open-set label noise has received little attention in this respect. While there are works that address datasets with open-set label noise [13, 14, 6, 9, 10], the analysis is limited by the lack of empirical insight.

In this paper, we introduce *ARCA23K*[1] (Automatically Retrieved and Curated Audio 23K), which is a dataset containing more than 51 hours of audio data across 23 727 Freesound clips and 70 classes taken from the AudioSet ontology. The clips comprising the training set have been retrieved and curated using an entirely automated process, while the validation set and test set are subsets of FSD50K. Given the absence of human verification, labelling errors are to be expected in the training set. In particular, many of the audio clips are out-of-vocabulary.

Our aim in creating ARCA23K is to facilitate the study of real-world, open-set label noise, including its effects on learning and how these effects can be mitigated. Unlike datasets such as FSDnoisy18k, ARCA23K allows studying label noise in a more controlled manner. Instead of manually verifying a subset of the dataset, we introduce another dataset called *ARCA23K-FSD*, which is a subset of FSD50K. ARCA23K-FSD is essentially a 'clean' counterpart of ARCA23K. Under certain assumptions, this setup allows controlling the amount of label noise by substituting clips from one dataset with clips from the other. A similar idea was proposed in the image domain [15].

The contributions of this paper are four-fold. First, we provide a detailed description of how the ARCA23K datasets were created and release the associated source code[2]. Our intention is to provide a method of dataset creation that is realistic while also being easily reproducible[3], such that anyone can adopt or improve our method for their own needs. Our second contribution is the release of ARCA23K itself (along with ARCA23K-FSD). As all the clips are available under a Creative Commons license, we are able to distribute the clips freely. Third, we characterise the label noise present in ARCA23K by running listening tests. Finally, we conduct experiments to examine the impact of open-set label noise on training audio classifiers, which includes comparisons to synthetic label noise and an evaluation of the representations that are learned.

[1] `https://zenodo.org/record/5117901`
[2] `https://github.com/tqbl/arca23k-dataset`
[3] Some clips on Freesound may be deleted, which we cannot control.

2. ARCA23K-FSD

In order to investigate open-set label noise, we propose two datasets: a clean dataset with training examples drawn from P and a noisy dataset with training examples from Q. The number of examples per class is set to be equal across the two datasets. By satisfying this requirement, we are able to emulate the noise process that corrupts P to give Q. More specifically, a training example drawn from the clean dataset is corrupted by substituting it with a training example of the same class drawn from the noisy dataset. The amount of label noise can then be controlled by substituting a proportionate number of training examples.

The clean dataset, ARCA23K-FSD, is a subset of FSD50K [5], which is currently the largest clean dataset of sound events available. FSD50K is comprised of more than 40 k training examples and 200 classes taken from the AudioSet ontology. In general, multiple labels are associated with each audio clip.

For simplicity, we reduced FSD50K to a single-label dataset. First, clips containing more than one type of sound were discarded. Next, to prevent class overlap, classes that were ancestors of other classes (according to the AudioSet ontology) were dropped, e.g. clips labelled as Guitar would be dropped because Acoustic guitar and Electric guitar are child classes. Finally, any sound class with an insufficient number of audio clips was removed from the dataset. The thresholds are 50 instances in the training set, 10 instances in the validation set, and 20 instances in the test set. A total of 77 classes were retained after this pruning process. Let $\mathcal{L}$ denote the set of AudioSet labels that remained.

3. ARCA23K

In this section, we describe how the clips in the ARCA23K dataset were retrieved and curated. This dataset only includes a training set, since the validation set and test set of the ARCA23K-FSD dataset are used for validation and testing, respectively. We use a keyword-based algorithm to retrieve relevant clips and label them accordingly. We will assume that we have access to the metadata of every clip in the Freesound database and that we can download the clips. The metadata includes a description of the clip and a set of *tags* that are intended to be search terms. As with FSD50K, we limit our search to clips that are between 0.3 and 30 seconds [5]. After curation, all clips are converted to 16-bit mono WAV files sampled at 44.1 kHz.

3.1. Retrieval Algorithm

The general framework for the retrieval algorithm is as follows. For every candidate Freesound clip, the tags and description are tokenised and preprocessed to give two word sequences, d_{tags} and d_{desc}, which we refer to as *documents*. For each label, $l \in \mathcal{L}$, a *query*, q_l is constructed, which also involves tokenisation and preprocessing. Given q_l, d_{tags}, and d_{desc}, a relevance score, $r(q_l, d_{\text{tags}}, d_{\text{desc}}) \in [0, 1]$, is computed, such that a higher score indicates a better match between the query and the two documents. By computing scores for each label, the most relevant label can be assigned to the clip:

$$l^* \coloneqq \arg\max_l r(q_l, d_{\text{tags}}, d_{\text{desc}}). \tag{1}$$

If $r(q_{l^*}, d_{\text{tags}}, d_{\text{desc}})$ is a low score, it indicates that none of the labels are a good match according to the algorithm. For this reason, clips for which $r(q_{l^*}, d_{\text{tags}}, d_{\text{desc}}) < \tau$ are discarded, where τ is a predefined threshold.

3.1.1. Tokenisation and Preprocessing

Tags, descriptions, and labels are tokenised and preprocessed using standard practices in information retrieval [16]. Tokenisation refers to converting a sequence of characters into a sequence of words. During this process, non-words such as punctuation and numbers are discarded. Tags are already assumed to be a sequence of words, hence tokenisation is not necessary for tags.

Preprocessing is carried out by first converting the words to lower-case so that retrieval can be case-insensitive. Following this, lemmatisation is applied to canonicalise words to their lemma form, e.g. 'guitars' would be reduced to 'guitar'. In cases where the lemma depends on which word class the word belongs to (e.g. verb, noun), the shortest lemma is chosen. For instance, 'clapping' would be reduced to 'clap' because, even though 'clapping' is the lemma for the noun, 'clap' is the lemma for the verb and is the shortest. After lemmatisation, stop words such as conjunctions and prepositions are removed. Finally, any duplicate words are also removed.

3.1.2. Query Construction

Given a label l, a query, q_l, is constructed as follows. The label is first tokenised and preprocessed as per Section 3.1.1. We will refer to the resulting output as a *root query* and denote it as $\bar{q}_l$. Next, a root query is constructed for every descendant label of l. For example, the label Bowed string instrument has several descendants, such as Cello and Double bass. After constructing the root queries, the final query q_l is constructed by concatenating all of them.

3.1.3. Computing Relevance Scores

In this section, we describe how relevance scores are computed. After creating a query for each label $l \in \mathcal{L}$, the vocabulary, V, can be defined as the concatenation of all the queries (after removing any duplicates). After constructing V, one can map any sequence of words, w, into a vector, $v(w) \in \{0, 1\}^{|V|}$, such that

$$v(w)_i \coloneqq \begin{cases} 1 & w_i = V_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The relevance score is then defined as

$$r(q, d_{\text{tags}}, d_{\text{desc}}) \coloneqq \bar{v}(q) \cdot [\bar{v}(d_{\text{tags}}) + \bar{v}(d_{\text{desc}})], \tag{3}$$

where $\bar{v}(w) \coloneqq v(w)/\|v(w)\|$. In other words, $r(q, d_{\text{tags}}, d_{\text{desc}})$ is the cosine similarity between $v(q)$ and $[v(d_{\text{tags}}) + v(d_{\text{desc}})]$.

3.1.4. Evaluation

The Freesound clips that are labelled by our retrieval algorithm include all Freesound clips that are between 0.3 and 30 seconds in duration. This means the clips that constitute the ARCA23K-FSD dataset are labelled by our algorithm too. It is therefore possible to compare the labels assigned by our algorithm to the ground truth labels of ARCA23K-FSD.

We used a threshold of $\tau = 0.5$, as it resulted in a reasonable compromise between precision and recall. Our algorithm retrieved 84.1 % of the ARCA23K-FSD clips and achieved an accuracy of 90.3 %. The accuracy was greater than 90 % for 51 out of 77 classes, and the average accuracy for these classes was found to be 96.4 %. For the other 26 classes, the average accuracy was found to be 67 %.

It should be noted that the ARCA23K-FSD clips are not an unbiased sample of the retrieved clips. The aim of the evaluation is

Table 1: Estimates of the proportion of ARCA23K clips that are PP/PNP/NP. The percentage of clips marked 'Unsure' is 1 %.

	PP	PNP	NP
IV	$(52.7\pm5.8)\%$	$(2.3\pm1.3)\%$	$(8.7\pm3.5)\%$
OOV	N/A	$(1.3\pm0.7)\%$	$(33.3\pm5.6)\%$

not to determine label accuracy in general but to demonstrate that it is comparable to approaches used for existing datasets. In Section 3.3, we evaluate the accuracy of the labels by manually verifying a subset of the ARCA23K dataset.

3.2. Curation

After labelling the candidate Freesound clips using the retrieval algorithm, we used a threshold of $\tau = 0.5$ to discard clips with a low relevance score. All clips belonging to the FSD50K dataset were also discarded to prevent any overlap. The number of retrieved clips at this point totalled almost 170 k. Next, the number of clips per class was reduced to match ARCA23K-FSD, since our aim is to create a dataset that mirrors ARCA23K-FSD. This was done by selecting a random sample of the correct size from each class. For seven of the classes, there was an insufficient number of clips to match the ARCA23K-FSD dataset, so the clips belonging to these classes were dropped altogether. The same was done for ARCA23K-FSD, resulting in 70 classes in total for both datasets.

3.3. Noise Rate Estimation

In this section, we describe how noise rates were estimated for the ARCA23K dataset and present the results. The noise rate is defined as the percentage of incorrectly labelled audio clips in the dataset. Similar to Fonseca et al. [5], we categorise clips as either 'Present and predominant' (PP), 'Present but not predominant' (PNP), 'Not present' (NP), and 'Unsure' (U). The reader is referred to the original work for detailed definitions [5]. PNP and NP are further split based on whether the other sounds are in-vocabulary (IV) or out-of-vocabulary (OOV). For example, NP/OOV means that at least one OOV sound can be identified in the clip.

The noise rate of the dataset was estimated by selecting a random subset of the dataset and performing listening tests. We selected 100 clips for the sample and repeated the experiment three times with replacement. Each sample was processed by a different listener, i.e. three listeners participated. The first three authors of this paper carried out the tests. They were trained by familiarising themselves with the classes, which involved reading the class descriptions and listening to example clips. They were also able to listen to example clips during the test and confer with each other[4].

The results are presented in Table 1. The noise rate can be calculated by excluding the sounds categorised as U. When PNP sounds are considered as incorrect, the noise rate was found to be $(46.4\pm4.8)\%$ (95% confidence interval). When PNP sounds are considered as correct, the noise rate was found to be $(42.4\pm4.1)\%$. Based on the results in Table 1, 75.9 % of incorrectly labelled clips are OOV. For many of the NP clips, we were able to identify them as NP from the tags and description alone[5], meaning that the labelling errors were the fault of the retrieval algorithm; some were simple mistakes, while others required understanding the context, which a keyword-based retrieval algorithm cannot infer. In other cases, the uploaders' annotations were misleading or incorrect. This was more prevalent with classes such as Whoosh, swoosh, swish, which are more open to interpretation without an agreed-upon definition. Finally, we observed that many of the OOV sounds were quite similar in sound to the IV classes. For example, 462351.wav, labelled as Acoustic guitar, contains sounds of a guitar string being strummed, but it is too distorted to belong to any of the guitar classes.

[4]In practice, listeners only conferred when they were unsure.

[5]All clips were listened to in their entirety nonetheless.

4. EXPERIMENTS

In this section, we describe the experiments that were carried out and present the results. Systems are evaluated using the accuracy and the mean average precision (mAP). The mAP is approximately equal to the area under the precision-recall curve; a higher value indicates better performance. We ran each experiment five times and provide 95% confidence intervals for the scores.

4.1. System

The machine learning model used in our experiments is an 11-layer convolutional neural network based on the VGG13 architecture [17]. Our model differs from VGG13 in that it uses batch normalisation [18] and only one fully-connected layer instead of three, as we found multiple fully-connected layers to be unhelpful.

The model was trained with mel-spectrogram inputs. Prior to computing the mel-spectrograms, the audio was downsampled from 44.1 kHz to 32 kHz, which reduced the audio's data rate without significantly affecting the results. The mel-spectrograms were then computed using a 32 ms frame length, a 16 ms hop length, and 64 mel bins per frame. Finally, the amplitudes of the mel-spectrograms were scaled logarithmically.

Since the audio clips in both datasets vary in duration, we padded the clips with silence. Instead of padding to a single fixed length, we used three different lengths: 5 seconds, 15 seconds, and 30 seconds. The least amount of padding was applied to each clip, e.g. a clip less than 5 seconds would be padded to 5 seconds. When selecting clips for a mini-batch, only clips of the same length were allowed. Without this multi-length approach, each clip would have to be padded to the maximum length, which would greatly increase training times.

The model was trained for 50 epochs using the cross-entropy loss function and the AdamW optimiser [19] with a learning rate of 0.0005, which was decayed by 10 % every two epochs. We used a batch size of 64, 32, and 16 for 5-, 15-, and 30-second clips, respectively. By using different batch sizes, and given the memory constraints, we were able to significantly improve training times and even the classification accuracy. During inference, we averaged the predictions of the top three epochs in order to reduce volatility.

4.2. Adding Noise

In addition to training with the ARCA23K datasets, we also added synthetic label noise to the ARCA23K-FSD dataset, which allows us to compare synthetic label noise to the real-world label noise present in ARCA23K. The synthetic label noise is closed-set rather than open-set. Let $k \in \{1, \ldots, K\}$ represent the class associated with an instance, where $K = 70$ is the number of classes. To add synthetic noise, we selected a proportion, ρ, of training examples and changed the class k of each selected example to

$$(k + i) \mod K, \tag{4}$$

Table 2: Model performance when using different training sets.

Dataset	Accuracy	mAP
ARCA23K	$(50.08\pm0.78)\%$	$(52.32\pm0.77)\%$
ARCA23K-FSD	$(61.16\pm0.41)\%$	$(66.28\pm0.59)\%$
Uniform Noise	$(38.12\pm1.83)\%$	$(35.76\pm2.10)\%$
Conditional Noise	$(38.35\pm0.56)\%$	$(36.82\pm0.62)\%$

where i is a random integer drawn from a suitable distribution. Two types of label noise were considered: uniform and class-conditional. In the case of uniform noise, i followed the uniform distribution, $U(1, K-1)$. In the case of class-conditional noise, the geometric distribution was used with $p = 0.5$. This distribution is concentrated over a small number of outcomes, which is realistic because only a small set of classes tend to be incorrectly attributed to a sound.

Finally, we ran experiments in which we replaced a proportion, ρ, of the ARCA23K-FSD training examples with ARCA23K training examples. Recall from Section 2 that this is equivalent to controlling the noise rate of ARCA23K. For each example that was replaced, the replacement example was restricted to be identically labelled. The noise rate of the resulting mixed dataset is a fraction of the noise rate estimated in Section 3.3. For example, $\rho = 0$ corresponds to a noise rate of 0, $\rho = 1$ corresponds to a noise rate of $46.4\,\%$, and $\rho = 0.5$ corresponds to a noise rate of $23.2\,\%$.

4.3. Representation Learning

As a final set of experiments, we examined how label noise affects the representations that are learned. To do this, we trained a linear classifier on embeddings derived from the output of the VGG model's penultimate layer and evaluated its performance. The VGG model was first trained as normal using a noisy dataset (either ARCA23K or ARCA23K-FSD with synthetic label noise). Next, using the output of the penultimate layer as input data, a linear classifier was trained on the (clean) ARCA23K-FSD dataset. In addition to training with the whole of ARCA23K-FSD, we trained the linear classifier with $10\,\%$, $20\,\%$, and $50\,\%$ of the dataset.

4.4. Results

The first group of results are presented in Table 2. In this table, we compare the performance of the system when trained using: (1) ARCA23K, (2) ARCA23K-FSD, (3) ARCA23K-FSD but with uniform label noise, and (4) ARCA23K-FSD but with conditional label noise. We set $\rho = 0.45$ for both (3) and (4). The results show that training with ARCA23K-FSD gives an mAP score of $66.28\,\%$, which is 14% higher than when training with ARCA23K, which suggests that the presence of open-set label noise has a considerable effect on training. However, it can be seen that the effect of synthetic label noise is much more severe, as the mAP drops below 40%.

We hypothesise that there are at least two reasons why real-world, open-set label noise has a milder effect on training. First, we believe that OOV clips are inherently less likely to harm performance compared to mislabelled IV clips, especially if the OOV clips sound very different to the IV clips. Recall that most of the incorrectly labelled clips in ARCA23K are OOV. Second, as observed in Section 3.3, a considerable number of OOV clips were found to be similar in sound to the IV clips. If these clips are labelled accordingly (e.g. 462351.wav labelled as Acoustic guitar), they can be considered as surrogates of the IV clips. Rather than being detrimental to learning, these surrogates are likely to be beneficial. Both of these hypotheses were verified to some extent in previous work [13].

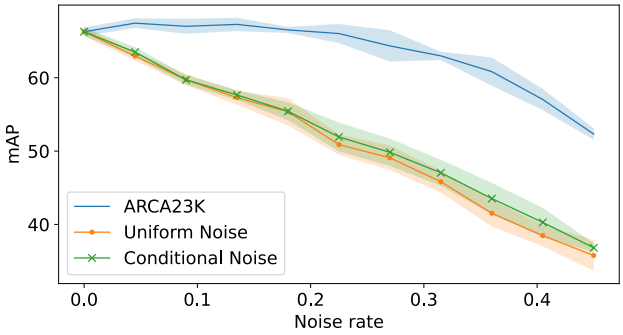


Figure 1: The mAP scores as ρ is varied from 0 to 0.45.

Table 3: mAP scores for the linear classifier. Columns indicate the percentage of ARCA23K-FSD clips used for training.

Dataset	$10\,\%$	$20\,\%$	$50\,\%$	$100\,\%$
ARCA23K	$55.27\,\%$	$58.94\,\%$	$58.91\,\%$	$59.82\,\%$
Uniform Noise	$30.86\,\%$	$37.11\,\%$	$42.29\,\%$	$45.52\,\%$
Conditional Noise	$41.15\,\%$	$46.06\,\%$	$48.11\,\%$	$50.09\,\%$
Random Weights	$7.93\,\%$	$10.11\,\%$	$13.12\,\%$	$16.23\,\%$

In Figure 1, we present the mAP scores as ρ (refer to Section 4.2) is varied from 0 to 0.45 at increments of 0.045. For all three types of noise, the performance generally decreases as ρ increases. However, while the plots appear linear for synthetic label noise, the plot for real-world, open-set label noise is non-linear. The performance decreases exponentially as the noise rate increases, albeit it is roughly the same until the noise rate exceeds $20\,\%$.

The results for the experiments described in Section 4.3 are presented in Table 3. We have also reported the performance when using a randomly-initialised VGG model to compute the embeddings. Similar to the results in Table 2, the performance is considerably worse when using synthetic label noise. When using ARCA23K to learn the representation, the performance of the linear classifier is relatively high even when training with $10\,\%$ of ARCA23K-FSD. On the other hand, the scores are still significantly lower than the score of $66.28\,\%$ in Table 2. These results show that label noise has a substantial effect on the quality of the learned representations.

5. CONCLUSION

In this paper, we introduced the ARCA23K dataset along with the companion ARCA23K-FSD dataset, which were created with the intention of studying open-set label noise. ARCA23K was created with minimal human labour by retrieving and curating clips from the Freesound database using an automated process, while ARCA23K-FSD was derived from FSD50K. We described the dataset creation process in detail and characterised the type of label noise present in ARCA23K via listening tests. Using these datasets, we were able to study the effect of label on learning in a controlled manner. We found that, while open-set label noise negatively affected performance, the impact was considerably milder than that of synthetic label noise. Furthermore, our experiments showed the extent to which label noise affects the learned representations of a model.

6. ACKNOWLEDGEMENT

This work was funded by the Engineering and Physical Sciences Research Council (EPSRC) Doctoral Training Partnership grants EP/N509772/1 and EP/R513350/1. It was also supported in part by EPSRC project EP/T019751/1 and by a Newton Institutional Links Award from the British Council with grant number 623805725.

7. REFERENCES

[1] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: A platform for the creation of open audio datasets," in *Proc. 18th Int. Society Music Information Retrieval Conf. (ISMIR)*, Suzhou, China, 2017, pp. 486–493.

[2] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 2017, pp. 776–780.

[3] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*, New York, NY, USA, 2015, pp. 1015–1018.

[4] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM International Conference on Multimedia*, Orlando, FL, USA, 2014, pp. 1041–1044.

[5] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "FSD50K: An open dataset of human-labeled sound events," *arXiv preprint arXiv:2010.00475*, Oct. 2020.

[6] B. Zhu, K. Xu, Q. Kong, H. Wang, and Y. Peng, "Audio tagging by cross filtering noisy labels," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2073–2083, Jul. 2020.

[7] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "VGGSound: A large-scale audio-visual dataset," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 721–725.

[8] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of Freesound audio with AudioSet labels: Task description, dataset, and baseline," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, Woking, UK, 2018, pp. 69–73.

[9] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, and X. Serra, "Audio tagging with noisy labels and minimal supervision," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York, NY, USA, 2019, pp. 69–73.

[10] E. Fonseca, M. Plakal, D. P. W. Ellis, F. Font, X. Favory, and X. Serra, "Learning sound event classifiers from web audio with noisy labels," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 21–25.

[11] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 2233–2241.

[12] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," in *International Conference on Learning Representations*, San Diego, CA, USA, 2015.

[13] T. Iqbal, Y. Cao, Q. Kong, M. D. Plumbley, and W. Wang, "Learning with out-of-distribution data for audio classification," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 636–640.

[14] A. Kumar, A. Shah, A. Hauptmann, and B. Raj, "Learning sound events from webly labeled data," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, Macao, China, 2019, pp. 2772–2778.

[15] L. Jiang, D. Huang, M. Liu, and W. Yang, "Beyond synthetic noise: Deep learning on controlled noisy labels," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, vol. 119, 2020, pp. 4804–4815.

[16] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, vol. 37, Lille, France, 2015, pp. 448–456.

[19] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.

AN ENCODER-DECODER BASED AUDIO CAPTIONING SYSTEM WITH TRANSFER AND REINFORCEMENT LEARNING

Xinhao Mei[1], *Qiushi Huang*[1], *Xubo Liu*[1], *Gengyun Chen*[2], *Jingqian Wu*[3*], *Yusong Wu*[3†], *Jinzheng Zhao*[1], *Shengchen Li*[3], *Tom Ko*[4], *H Lilian Tang*[1], *Xi Shao*[2], *Mark D. Plumbley*[1], *Wenwu Wang*[1]

[1] University of Surrey, Guildford, United Kingdom
[2] Nanjing University of Posts and Telecommunications, Nanjing, China
[3] Xi'an Jiaotong-Liverpool University, Suzhou, China
[4] Southern University of Science and Technology, Shenzhen, China

ABSTRACT

Automated audio captioning aims to use natural language to describe the content of audio data. This paper presents an audio captioning system with an encoder-decoder architecture, where the decoder predicts words based on audio features extracted by the encoder. To improve the proposed system, transfer learning from either an upstream audio-related task or a large in-domain dataset is introduced to mitigate the problem induced by data scarcity. Moreover, evaluation metrics are incorporated into the optimization of the model with reinforcement learning, which helps address the problem of "exposure bias" induced by "teacher forcing" training strategy and the mismatch between the evaluation metrics and the loss function. The resulting system was ranked 3rd in DCASE 2021 Task 6. Ablation studies are carried out to investigate how much each component in the proposed system can contribute to final performance. The results show that the proposed techniques significantly improve the scores of the evaluation metrics, however, reinforcement learning may impact adversely on the quality of the generated captions.

Index Terms— audio captioning, transfer learning, sequence-to-sequence model, cross-modal task

1. INTRODUCTION

An automated audio captioning (AAC) system describes an audio signal using natural language, which is a cross-modal translation task involving the technologies of audio processing and natural language processing [1]. Generating a meaningful description of an audio clip not only requires recognizing what audio events are presented, but also their properties, activities as well as spatial-temporal relationships. Audio captioning could be useful in several applications, such as subtitling for sound in a television program, assisting the hearing-impaired to understand environmental sounds, and analysing sounds in smart cities for security surveillance.

Drossos et al. [1] proposed the initial work in audio captioning, where they introduced an encoder-decoder architecture based on recurrent neural networks (RNNs) on a commercial sound effects library, ProSound Effects. After that, with the release of two freely available datasets AudioCaps [2] and Clotho [3], and a new audio captioning task in DCASE challenges, this field has received increasing attention. Almost all researchers investigating audio captioning have utilised an encoder-decoder architecture based on deep neural networks (DNNs). For the encoder, recurrent neural networks (RNNs) [1, 2, 4], convolutional neural networks (CNNs) [5, 6], or their combinations, i.e. convolutional recurrent neural network (CRNN) [7], have been used to model the temporal, or temporal-spectral relationship between audio features. For the decoder, recurrent neural network (RNN) has been widely used to generate captions by decoding audio features to text descriptions [1, 2, 4, 7]. To align the cross-modal representation between audio and language, attention mechanisms with different implementation methods have been used between the encoder and decoder [2, 8]. With the popularity of Transformer in natural language processing (NLP) and computer vision (CV), some researchers try to use Transformer as the decoder [5, 9, 10]. In addition, keywords and semantic information predicted from the input audio are introduced to guide caption generation [4, 9]. The encoder-decoder architecture with "CNN-Transformer" was shown to offer better performance in the DCASE 2020 challenge [5], which is chosen as the baseline system in our work.

As the availability of data in the audio captioning task is limited, training an end-to-end cross-modal audio captioning system from scratch becomes even more difficult. Transfer learning has been widely used to solve this data scarcity problem, where pre-trained audio models from an upstream audio processing task (i.e. audio tagging and sound event detection) are utilized as the audio encoder [5, 11]. As using pre-trained audio models can only transfer knowledge in the audio modality, we also pre-train the whole network on the AudioCaps dataset [2] in order to transfer knowledge in both audio and language modalities. Both transfer learning strategies are adapted and compared in the proposed system.

Another problem is the mismatch between the evaluation metrics and the loss function used for text generation. The evaluation metrics are discrete and non-differentiable, thus cannot be optimized directly by back-propagation. Previous works use reinforcement learning by incorporating the evaluation metrics into the optimisation of the learning system [7, 12]. We analyze the effects of reinforcement learning on audio captioning system. The results show that even though reinforcement learning can improve the score of evaluation metrics, it may impact adversely on the quality of the generated captions, in the sense that some redundant words are introduced in the captions generated. This finding indicates that existing metrics used for caption evaluation do not correlate well with human judgment. Our resulting system[1] was ranked in the 3rd place in DCASE

*Jingqian Wu is currently with Wake Forest University, USA
†Yusong Wu is currently with University of Montreal, Canada

[1] https://github.com/XinhaoMei/DCASE2021_task6_v2.git

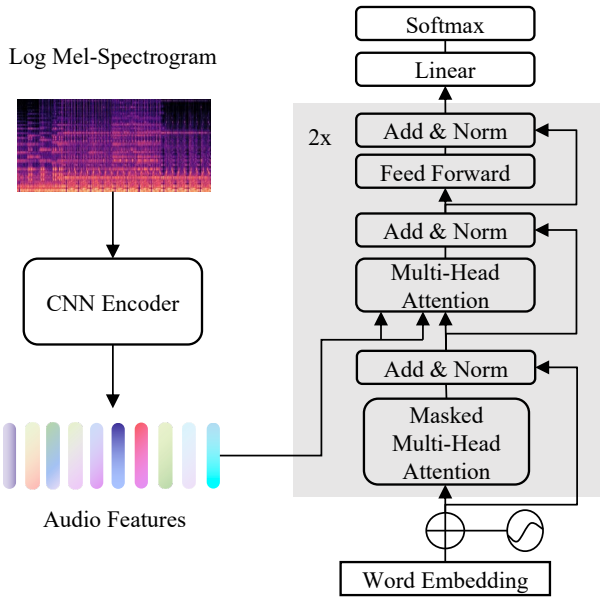


Figure 1: Architecture of the proposed model.

2021 Task 6 and was the highest scoring system without using an ensemble technique.

The remaining sections of this paper are organised as follows. In Section 2, the proposed model and methods are described in detail. Experimental setup is described in Section 3. Results are shown in Section 4. Finally, we conclude our work in Section 5.

2. PROPOSED METHOD

The proposed model consists of a CNN encoder and a Transformer decoder. The encoder takes the log mel-spectrogram $X \in \mathbb{R}^{T \times D}$ of an audio clip as input and produces audio feature vector $v \in \mathbb{R}^{T' \times D'}$. The decoder predicts the posterior probability of the n-th word w_n based on the feature vector v produced by the encoder and previously generated words w_0 to w_{n-1}. Mathematically,

$$v = \mathrm{Enc}(X) \tag{1}$$

$$p(w_n|v, w_0, ..., w_{n-1}) = \mathrm{Dec}(v, w_0, ..., w_{n-1}) \tag{2}$$

The diagram of the proposed system is shown in Fig. 1.

2.1. Model architecture

2.1.1. Encoder

Convolutional neural networks (CNNs) have been used widely in audio processing related works and have shown powerful ability in extracting audio features. A relatively simple 10-layer CNN proposed in the pre-trained audio neural networks (PANNs) [13] is used as the encoder to mitigate the over-fitting problem. The 10-layer CNN consists of four convolutional blocks where each has two convolutional layers with a kernel size of 3×3. The number of channels in each block is 64, 128, 256 and 512 respectively, and an average pooling layer with a kernel size of 2×2 is applied between them for down-sampling. After each convolutional layer, batch normalization and ReLU nonlinearity are used. Global average pooling is applied along the frequency axis after the last convolutional block and two fully connected layers are followed to further increase the representation ability and to ensure the dimension of the output is compatible with that of the decoder.

2.1.2. Decoder

The decoder consists of three parts, i.e., a word embedding layer, a standard Transformer decoder and a linear layer. Each input word is firstly encoded to a vector of fixed dimension through the word embedding layer. The word embedding layer can be regarded as an embedding look-up matrix of size $V \times d$, where V is the size of the vocabulary and d is the dimension of the word vector. This layer is randomly initialized and kept frozen during the training stage.

Transformer is designed to handle sequential data and shows state-of-the-art performance in generation tasks in the area of natural language processing [14]. The Transformer decoder is used as the multi-modal decoder here. Word embeddings from the word embedding layer together with the audio features obtained from the encoder are passed to the transformer decoder and are incorporated through a multi-head attention mechanism. As the captions in the datasets are mostly short in length, the decoder used only consists of two transformer decoder blocks with four heads. The dimension of the hidden layer is 128. A linear layer is used at last to output a probability distribution along the vocabulary.

2.2. Transfer learning

Transfer learning aims to transfer knowledge from the source domain to the target domain in order to solve the problem caused by insufficient training data and improve the generalization ability of the model. Transfer learning is mostly used in tasks involving single modality. For this cross-modal (i.e. audio to text) translation task, we introduce two transfer learning methods, where the first is transferring from an upstream task while the second is from an in-domain dataset.

The encoder extracts audio features from an audio clip and is a relatively separate component in the whole model, thus pre-trained audio models can be adapted as the encoder. Different pre-trained audio models have recently been published which can extract generalized audio patterns. PANNs [13] are the models pre-trained on the AudioSet dataset for an audio tagging task which have achieved state-of-the-art performance in many downstream audio pattern recognition tasks. One of the PANNs is used to initialize the parameters in the encoder in order to overcome the data scarcity problem and extract generalized audio features.

There are many powerful pre-trained language models for text generation tasks [15]. However, since pre-trained language models do not have encoder-decoder attention modules, it is not feasible to directly use a pre-trained language model as the cross-modal decoder here. In order to transfer knowledge in both modalities, AudioCaps, the largest audio captioning dataset currently available, is introduced to pre-train the proposed model, which allows transfer learning to be applied in both audio and language modalities.

2.3. Reinforcement learning

The training objective of audio captioning systems is usually to optimize the cross-entropy (CE) loss. That is, the model parameters

θ are trained to minimize

$$\mathcal{L}_{\mathrm{CE}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\log p(y_t|y_{1:t-1}, v, \theta) \quad (3)$$

where y_t is the ground-truth word at time step t. The model is trained with "teacher forcing" strategy, i.e. each word to be predicted is conditioned on previous ground-truth words in the training stage, while it is conditioned on previous output words in the test stage. This discrepancy leads to error accumulation during text generation in the test stage and is known as "exposure bias" [16]. Another problem is the mismatch between the training objective and the evaluation metrics. The performance of captioning systems is evaluated by discrete metrics, which are non-differentiable and cannot be directly optimized by back-propagation. To address these two problems, reinforcement learning with the policy gradient (PG) method is used to optimize the evaluation metrics considered and to directly improve the scores in terms of these metrics [7, 16, 12].

Reinforcement learning makes it possible to directly back-propagate the evaluation metrics in the form of a reward, which is computed by an evaluation metric. In our work, the model is trained to minimize the negative expected reward:

$$\mathcal{L}_{\mathrm{RL}}(\theta) = -\mathbb{E}_{w^s\sim p_\theta}[r(w^s)], \quad (4)$$

where $w^s = (w_1^s, ..., w_T^s)$ and w_t^s is the word sampled from the model at time step t. To compute the gradient of the negative reward, we choose the self-critical sequence training (SCST) method [16], which directly optimizes the true, sequence-level evaluation metric, but avoids learning an estimate of expected future rewards as a baseline. The expected gradient with a single sample $w^s \sim p_\theta$ can be approximated as:

$$\nabla_\theta \mathcal{L}_{\mathrm{RL}}(\theta) \approx -(r(w^s) - r(\hat{w}))\nabla_\theta \log p_\theta(w^s), \quad (5)$$

where $r(\hat{w})$ is the reward computed by the current model using a greedy inference algorithm.

3. EXPERIMENTS

3.1. Datasets

3.1.1. Clotho

Clotho [3] is an audio captioning dataset whose audio clips are all collected from the Freesound archive. To encourage caption diversity, each audio clip is provided with five captions annotated by different annotators. The duration of the audio clips ranges uniformly from 15 to 30 seconds. All the captions contain eight to 20 words.

Clotho v2 contains 3839 audio clips with 19 200 captions in the development split, and 1045 audio clips with 5225 captions in the validation and evaluation split, respectively. We merge the training and validation split together, which gives a new training set with 4884 audio clips.

During training, each audio clip is combined with one of its five captions as a training sample. During evaluation, all five ground-truth captions of an audio clip are used as references and compared with the predicted caption for metric computation.

3.1.2. AudioCaps

AudioCaps [2] is the largest audio captioning dataset currently available, which contains around 50k audio clips sourced from AudioSet with a duration of 10 seconds. AudioCaps is divided into three splits with 49 274 audio clips in the training set, 494 and 957 audio clips in the validation and test set, respectively. Each audio clip contains one caption in the training set, while each contains five captions in the validation and test sets. The length of the captions varies, with some containing only three words while some having more than 20 words.

3.2. Data pre-processing

The input features we used are 64-dimensional log mel-spectrograms obtained using a 1024-point Hanning window with a hop size of 512-points. SpecAugment [17] is used to augment data during training, which operates on the log mel-spectrogram of an audio clip using frequency masking and time masking.

All captions in the two datasets are transformed to lower case with punctuation removed. Two special tokens "<sos>" and "<eos>" are padded at the beginning and end of each caption. The vocabulary of the Clotho dataset contains 4367 words. As Clotho and AudioCaps have distinct vocabularies, for transfer learning from AudioCaps to Clotho, these two vocabularies are merged together which give a vocabulary containing 6636 words.

3.3. Experimental setups

The whole model is trained using Adam [19] optimizer with a batch size of 32. Warm-up is used in the first 5 epochs to linearly increase the learning rate to the initial learning rate. The learning rate is then decreased to $1/10$ of itself every 10 epochs. Dropout with rate 0.2 is applied in the proposed model to mitigate the over-fitting problem. To improve the generalization ability of the model and avoid over-confident prediction, label smoothing [20] with $\epsilon = 0.1$ is used in all our experiments. During the inference stage, beam search with a beam size up to 5 is used to improve the decoding performance.

For cross-entropy training, the model is directly trained on the Clotho dataset for 30 epochs or firstly pre-trained on the AudioCaps dataset for 30 epochs then fine-tuned on Clotho dataset for 30 epochs with an initial learning rate of 1×10^{-3}. The best model in terms of the SPIDE$_r$ score is selected to optimize CIDER$_r$ score using reinforcement learning for 60 epochs with a constant learning rate of 5×10^{-5} (in the DCASE challenge, we ran reinforcement learning for 25 epochs with a constant learning rate of 1×10^{-4} [21]).

3.4. Evaluation metrics

In the DCASE 2021 Task 6, audio captioning systems are evaluated by machine translation metrics (BLEU$_n$, ROUGE$_l$ and METEOR) and captioning metrics (CIDE$_r$, SPICE and SPIDE$_r$). BLEU$_n$ [22] is calculated as a weighted geometric mean of modified precision of n-grams. ROUGE$_l$ [23] calculates F-measures based on the longest common subsequence. METEOR [24] measures a harmonic mean of precision and recall based on word level matches between the candidate sentence and references. CIDE$_r$ [25] applies term frequency inverse document frequency (TF-IDF) weights to n-grams and calculates the cosine similarity between them. SPICE [26] transforms captions into scene graphs and calculates F-score based on tuples in them. SPIDEr [12] is a linear combination of SPICE and CIDE$_r$, the SPICE score ensures captions are semantically faithful to the audio clip, while CIDE$_r$ score ensures captions are syntactically fluent.

Model	$BLEU_1$	$BLEU_2$	$BLEU_3$	$BLEU_4$	$ROUGE_L$	METERO	CIDEr	SPICE	SPIDEr
Baseline	0.525	0.344	0.237	0.163	0.359	0.154	0.352	0.100	0.226
B+PANNs	0.564	0.375	0.255	0.171	0.383	0.172	0.421	0.120	0.270
B+PANNs+AC	0.561	0.374	0.257	0.174	0.379	0.171	0.426	0.124	0.275
B+PANNs+RL	0.639	0.415	0.276	0.174	0.401	0.186	0.452	0.131	0.292
B+PANNs+AC+RL	0.634	**0.423**	**0.288**	**0.185**	**0.410**	**0.187**	**0.476**	**0.134**	**0.305**
SJTU [18]	0.643	-	-	0.163	0.404	0.178	0.449	0.123	0.286
SJTU_ensemble [18]	**0.657**	-	-	0.174	0.408	0.182	0.468	0.123	0.295

Table 1: Scores of our models on the Clotho v2 evaluation set. Baseline (B): the proposed model trained from scratch. RL: the model fine-tuned using reinforcement learning. PANNs: use PANNs as the audio encoder. AC: the whole model pre-trained on the AudioCaps dataset. Higher score indicates better system performance.

Examples	B+PANNs (w/o RL)	B+PANNS+RL (w/ RL)
example 1	a crowd of people are talking and cheering	a crowd of people are talking and in the background
example 2	a car is driving down the road with the windows open	a car is driving by on and then the engine of a vehicle
example 3	someone is playing a guitar with a stick	a guitar is being played on a guitar in the background
example 4	a machine is running at a constant rate	a machine is running and a in the background
example 5	a police car with a siren blaring in the background	a siren is blaring while sirens are blaring in the background

Table 2: Examples of selected captions generated by the model "B+PANNs" and "B+PANNs+RL".

Model	# audio clips
B+PANNs (w/o RL)	155
B+PANNS+RL (w/ RL)	765
Ground-truth	302

Table 3: The number of audio clips containing "in the background" in generated and ground-truth captions in the evaluation set.

4. RESULTS

Table 1 presents the performances of the proposed system on the Clotho v2 evaluation set. The proposed system is compared with SJTU's system [18] which won second place in DCASE 2021 Task 6 and shows state-of-the-art performance in audio captioning [11]. SJTU's system is based on a "CNN+RNN" architecture, transfer learning and reinforcement learning are also used. In addition, an ensemble strategy is adopted in their system to enhance the model performance. As can be seen in Table 1, our best model outperforms SJTU's ensemble model in most evaluation metrics (except $BLEU_1$), which shows the effectiveness of the proposed model.

Ablation studies are carried out to investigate the effects of each proposed component. From the experimental results, both transfer learning and reinforcement learning can improve system performance with respect to all the evaluation metrics. For transfer learning, the pre-trained audio encoder (PANNs) significantly improve all the metrics as compared to the system trained from scratch, which indicates that a powerful audio encoder is rather important in this cross-modal translation task. Pre-training on the AudioCaps dataset slightly improves most metrics, which confirms that transfer learning in both audio and language modalities performs better than that in a single modality only.

Reinforcement learning also improves all the evaluation metrics, although it is only used to optimize $CIDE_r$ score. However, it is somewhat surprising that reinforcement learning may impact adversely on the quality of the generated captions. First, reinforcement learning may lead to captions syntactically incorrect, introduces some repetitive words and generates incomplete captions. As shown in Table 2, we present five example captions generated by model "B+PANNs" and "B+PANNs+RL" to demonstrate this observation. Second, after the optimization with reinforcement learning, most captions are appended a phrase "in the background" which was not in their ground truth captions. Table 3 shows the statistics of the number of audio clips for which the generated captions contain "in the background" before and after using reinforcement learning. There are 302 audio clips whose ground-truth captions contain "in the background". After using reinforcement learning, 765 predicted captions contain "in the background", five times more than those without the use of reinforcement learning. These findings suggest that the existing evaluation metrics may not be able to fully reflect the effectiveness of an audio captioning system, or neither are they consistent with human judgement.

5. CONCLUSION

We have presented a "CNN+Transformer" audio captioning system with transfer and reinforcement learning and carried out ablation studies on the proposed methods. The results suggest that transfer and reinforcement learning can both improve the performance in terms of the evaluation metrics, while reinforcement learning may impact adversely on the quality of the generated captions. This finding indicates that the existing evaluation metrics used in the captioning system may not fully reflect the quality of captions. Further research should be carried out to find evaluation metrics that match well with human judgment.

6. ACKNOWLEDGMENT

This work is partly supported by grant EP/T019751/1 from the Engineering and Physical Sciences Research Council (EPSRC), a Newton Institutional Links Award from the British Council, titled "Automated Captioning of Image and Audio for Visually and Hearing Impaired" (Grant number 623805725) and a Research Scholarship from the China Scholarship Council (CSC) No. 202006470010.

References

[1] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2017, pp. 374–378.

[2] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 119–132.

[3] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 736–740.

[4] K. Nguyen, K. Drossos, and T. Virtanen, "Temporal subsampling of audio feature sequences for automated audio captioning," *arXiv preprint arXiv:2007.02676*, 2020.

[5] K. Chen, Y. Wu, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on Transformer and pre-trained CNN," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop*, 2020, pp. 21–25.

[6] A. Tran, K. Drossos, and T. Virtanen, "WaveTransformer: A novel architecture for audio captioning based on learning temporal and time-frequency information," *arXiv preprint arXiv:2010.11098*, 2020.

[7] X. Xu, H. Dinkel, M. Wu, and K. Yu, "A CRNN-GRU based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop*, 2020, pp. 225–229.

[8] H. Wang, B. Yang, Y. Zou, and D. Chong, "Automated audio captioning with temporal attention," DCASE2020 Challenge, Tech. Rep., 2020.

[9] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A Transformer-based audio captioning model with keyword estimation," *arXiv preprint arXiv:2007.00222*, 2020.

[10] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre-and post-processings for audio captioning," *arXiv preprint arXiv:2009.11436*, 2020.

[11] X. Xu, H. Dinkel, M. Wu, Z. Xie, and K. Yu, "Investigating local and global information for automated audio captioning with transfer learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 905–909.

[12] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of SPIDEr," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 873–881.

[13] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[15] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, L. Zhang, W. Han, M. Huang *et al.*, "Pre-trained models: past, present and future," *AI Open*, 2021.

[16] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 7008–7024.

[17] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[18] X. Xu, Z. Xie, M. Wu, and K. Yu, "The SJTU system for DCASE2021 challenge task 6: audio captioning based on encoder pre-training and reinforcement learning," DCASE2021 Challenge, Tech. Rep., July 2021.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[21] X. Mei, Q. Huang, X. Liu, G. Chen, J. Wu, Y. Wu, J. Zhao, S. Li, T. Ko, H. L. Tang, X. Shao, M. D. Plumbley, and W. Wang, "An encoder-decoder based audio captioning system with transfer and reinforcement learning for DCASE challenge 2021 task 6," DCASE2021 Challenge, Tech. Rep., July 2021.

[22] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[23] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004, pp. 74–81.

[24] A. Lavie and A. Agarwal, "METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments," in *Proceedings of the Second Workshop on Statistical Machine Translation*, 2007, pp. 228–231.

[25] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "CIDEr: Consensus-based image description evaluation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4566–4575.

[26] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "SPICE: Semantic propositional image caption evaluation," in *European Conference on Computer Vision*. Springer, 2016, pp. 382–398.

AUDIO CAPTIONING TRANSFORMER

Xinhao Mei[1], *Xubo Liu*[1], *Qiushi Huang*[2], *Mark D. Plumbley*[1], *Wenwu Wang*[1]

[1] Centre for Vision, Speech and Signal Processing (CVSSP),
[2] Department of Computer Science,
University of Surrey, UK

ABSTRACT

Audio captioning aims to automatically generate a natural language description of an audio clip. Most captioning models follow an encoder-decoder architecture, where the decoder predicts words based on the audio features extracted by the encoder. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are often used as the audio encoder. However, CNNs can be limited in modelling temporal relationships among the time frames in an audio signal, while RNNs can be limited in modelling the long-range dependencies among the time frames. In this paper, we propose an Audio Captioning Transformer (ACT), which is a full Transformer network based on an encoder-decoder architecture and is totally convolution-free. The proposed method has a better ability to model the global information within an audio signal as well as capture temporal relationships between audio events. We evaluate our model on AudioCaps, which is the largest audio captioning dataset publicly available. Our model shows competitive performance compared to other state-of-the-art approaches.

Index Terms— Audio captioning, Transformer, sequence-to-sequence model, cross-modal task

1. INTRODUCTION

Automated audio captioning (AAC) is concerned with describing an audio clip using natural language and is a cross-modal translation task at the intersection of audio processing and natural language processing. Generating a meaningful description for an audio clip not only needs to determine what audio events are presented, but also needs to capture and express their spatial-temporal relationships. Audio captioning is practically useful in applications such as assisting the hearing-impaired to understand environmental sounds, retrieving multimedia content, and analyzing sounds for security surveillance.

Unlike image and video captioning, which have been studied in computer vision (CV) for a longer time, audio captioning is a task investigated only recently [1]. With the announcement of the AAC task in DCASE 2020 and 2021, this topic has attracted increasing attention, and several methods have been proposed [2, 3, 4]. The AAC task is usually treated as a sequence-to-sequence problem, and existing methods are typically based on an encoder-decoder architecture, where the decoder generates words according to the audio features extracted by the encoder. Early works often adopted an "RNN-RNN" architecture with an attention mechanism [1, 3]. However, RNNs can be limited in modeling long-term temporal dependencies in an audio signal. Recently, CNNs have become a dominant approach in audio-related tasks (audio tagging and sound event detection) [5], with many researchers using pre-trained CNNs as the audio encoder, which significantly improved the performance in these systems [6]. More recently, inspired by the great success of the Transformer model in natural language processing [7], the RNN decoder has been replaced by a Transformer decoder in captioning models, and the "CNN+Transformer" architecture has been shown to achieve state-of-the-art performance in this area [8, 9].

Description of an audio signal needs to capture temporal-spatial relationships between audio objects that may be far apart in time. However, convolution is a local operator and has limitations in modelling temporal information, especially with a long audio signal. This can be alleviated by enlarging receptive fields with deeper convolutional layers. However, such deep CNNs can be hard to train and can lead to over-fitting. To address this problem, we propose an Audio Captioning Transformer (ACT), a convolution-free Transformer network based on the self-attention mechanism. We use log mel-spectrograms as input and split the mel-spectrograms into smaller non-overlapping patches along the time axis. By adopting the self-attention mechanism, each patch can attend to all the other patches at each layer of the encoder, which can model global long-range dependencies among the small mel-spectrogram patches from the beginning. Without the need for down-sampling, the features extracted by Transformer are fine-grained, which can contain detailed local audio topics.

The Transformer usually requires more training data than CNNs [10]. However, the amount of data currently available for audio captioning is relatively small. To address this issue, the ACT encoder is firstly pre-trained on AudioSet dataset [11] as an audio tagging task in order to improve its generalization ability. A class token designed to model the global information of an audio clip is appended at the beginning of each patch sequence and is used to output audio tagging results. As a result, when generating words, the decoder can attend to local and global information of an audio clip simultaneously. The proposed ACT model is evaluated on the AudioCaps dataset [3] and shows competitive performance as compared to other state-of-the-art methods.

The remaining sections of this paper are organised as follows. In Section 2, we introduce the related work. The proposed model is described in detail in Section 3. Experimental settings are shown in Section 4. Results are discussed in Section 5. Finally, we conclude our work in Section 6.

2. RELATED WORK

Previous work proposed in audio captioning has been based on deep learning methods with an encoder-decoder architecture. Drossos et al. [1] proposed the first approach to AAC using an RNN-based encoder-decoder architecture with an alignment model in between. To control the information contained in the output text, Ikawa and Kashino [4] introduced a conditional parameter called "specificity" to guide the caption generation. With the release of two freely avail-

able datasets AudioCaps [3] and Clotho [12], AAC has attracted increasing attention and more approaches have been proposed. Kim et al. [3] proposed a model with a top-down multi-scale encoder and aligned semantic attention, which enabled the joint use of multi-level features and semantic attributes. As CNNs have achieved state-of-the-art performance in audio tagging and sound event detection tasks [5], some researchers replaced the RNN encoder with CNNs, which brings significant performance gains [8, 6]. Recently, Transformer has been introduced as the language decoder with a powerful ability in natural language generation tasks [8, 13, 14]. Takeuchi et al. [15] formulated audio captioning as a multi-task learning problem, where they proposed keywords estimation and sentence length estimation to avoid the indeterminacy of word selection. Koizumi et al. [16] utilized a pre-trained large-scale language model GPT-2 [17] with audio-based similar caption retrieval to guide the caption generation. Liu et al. [18] introduced a contrastive loss to get better alignment between audio and texts in the latent space. Reinforcement learning was used to optimize the audio captioning models with non-differentiable evaluation metrics [19].

The Transformer was originally proposed for machine translation and has now become the dominant approach in natural language processing tasks [7]. Recently, many researchers adopted the Transformer for computer vision tasks which was shown to approach or outperform the state-of-the-art CNNs-based systems in image recognition. Dosovitskiy et al. [10] proposed a Vision Transformer (ViT) which was based purely on the attention mechanism, i.e. without using convolution kernels, and applied directly to sequences of image patches for the image classification task. However, a large amount of data are required for pre-training the Transformer models, which limits their adoption. To address this problem, Touvron et al. [20] introduced Data-efficient image Transformers (DeiT) using a data efficiency training and distillation strategy. Based on ViT and DeiT, Liu et al. [21] proposed a CaPtion TransformeR (CPTR) for image captioning. As the Transformer is designed to deal with sequential data, we argue that the Transformer can be adapted for audio signals, and the self-attention mechanism makes it more suitable to capture temporal relationships between audio features and to model the global information. Inspired by these ViT-related works, we propose the Audio Captioning Transformer (ACT) for audio captioning, which, to our knowledge, has not been done in the literature.

3. PROPOSED METHOD

Fig. 1 shows the proposed Audio Captioning Transformer model, which is based on the traditional sequence-to-sequence architecture and is convolution-free. The model takes the log mel-spectrogram of an audio clip as input and outputs the posterior probabilities of the predicted words.

3.1. Encoder

Let $X \in \mathbb{R}^{T \times F}$ denote the log mel-spectrogram of an audio clip, where T is the number of time frames and F is the number of mel bins. The log mel-spectrogram is first split into N non-overlapping small patches $X_N = \{x_1, ..., x_n\}$ along the time axis with size of $t \times F$ where $N = T/t$ and t is the number of time frames of each patch. Then each mel-spectrogam patch is flattened to a 1D embedding and projected to a latent space through a learnable matrix $W_e \in \mathbb{R}^{(t \times F) \times d}$, where d is the dimension of the latent embedding. In line with ViT and DeiT, a global learnable class token $X_{\text{cls}} \in \mathbb{R}^{1 \times d}$ is appended to the beginning of the patch sequences,

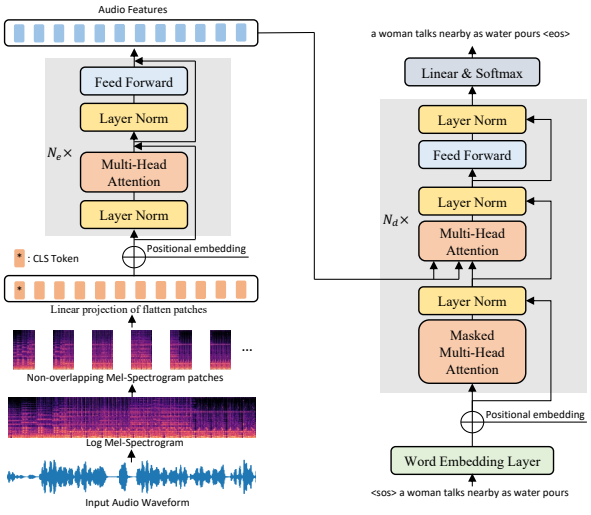


Figure 1: System overview of Audio Captioning Transformer, the encoder is on the left side while the decoder on the right side.

which contains the global information for the audio clip. As the self-attention mechanism cannot capture position information [7], a trainable positional embedding $X_{\text{pos}} \in \mathbb{R}^{(T+1) \times d}$ is added to each patch embedding. Mathematically, the final input representation is given by

$$X_e = [X_{\text{cls}} + W_e X] + X_{\text{pos}} \tag{1}$$

The ACT encoder consists of N_e stacked identical layers. Each layer contains two sub-layers, a multi-head self-attention layer and a position-wise fully-connected feed-forward layer. In the self-attention sub-layer, the input is first transformed into query Q, key K and value V through matrix multiplication with three learnable matrices $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$, where d_k is the dimension of each attention head. Then the scaled dot-product attention is computed as

$$\text{Attn}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

Each self-attention layer contains h attention heads which extends the model's ability to attend to different positions and creates multiple representation subspaces [7]. The outputs of heads are then aggregated through a linear transformation matrix $W_o \in \mathbb{R}^{(h \times d_k) \times d_k}$, which can be formulated as

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_o \tag{3}$$

The feed-forward network contains two linear layers with GLEU activation function and dropout applied between them. Layer normalization is applied before each sub-layer and a residual connection is employed around each of them, such that the output of each sub-layer is given by

$$X_{\text{out}} = X_{\text{in}} + \text{Sub\_layer}(\text{LayerNorm}(X_{\text{in}})) \tag{4}$$

In order to make use of pre-trained models, the encoder architecture is the same as ViT and DeiT containing 12 encoder blocks and 12 heads with an embedding dimension of 768.

Model	embedding dim	# layers (N_d)	# heads
ACT_s	512	2	4
ACT_m	512	4	8
ACT_l	512	6	8

Table 1: Variants of the proposed ACT decoder.

3.2. Decoder

The ACT decoder contains three parts: a word embedding layer, a Transformer decoder block, and a linear layer. Each input word is embedded through the word embedding layer into a fixed dimension word vector and then fed into the Transformer decoder block. The word vectors are pre-trained by a Word2Vec model on all caption corpus [22].

The Transformer decoder consists of N_d identical stacked layers. There are two main differences compared to the ACT encoder block. First, the first self-attention sub-layer in the decoder is a masked self-attention because the caption generating process is causal and auto-regressive. Second, there is a new cross multi-head attention sub-layer between self-attention sub-layer and feed-forward sub-layer, which allows every position in the decoder to attend over all positions in the audio features extracted by the encoder [7]. The output of the decoder module is fed through a final linear layer with a softmax activation function to output a probability distribution over the vocabulary.

The training objective of the model is to minimize the cross-entropy (CE) loss

$$\mathcal{L}_\mathrm{CE}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\log p(y_t|y_{1:t-1},\theta) \tag{5}$$

where y_t is the ground-truth word at time step t and θ are the model parameters. The "Teacher forcing" strategy is used during training, i.e. each word to be predicted is conditioned on previous ground-truth words. We experiment with three models, which share the same encoder architecture described in Section 3.2 but have different number of layers and heads in the decoder. Table 1 summarizes the parameters in the decoder of these models.

4. EXPERIMENTS

4.1. Dataset

4.1.1. AudioSet

AudioSet is a large-scale audio dataset with an ontology of 527 sound classes [11]. AudioSet contains more than 2 million 10-second audio clips extracted from YouTube videos. As some audio clips are no longer downloadable, there are 1 934 187 and 18 887 audio clips in our training and evaluation set, respectively. Each audio clip can have one or more labels for their presented audio events.

4.1.2. AudioCaps

AudioCaps is the largest audio captioning dataset currently available with around 50k audio clips sourced from AudioSet [3]. AudioCaps is divided into three splits. Each audio clip in the training set contains one human-annotated caption, while each contains five captions in the validation and test set.

4.2. Data pre-processing

All audio clips in these two datasets are converted to 32k Hz and padded to 10-second long. Log mel-spectrograms extracted using a 1024-points Hanning window with 50% overlap and 64 mel bins are used as the input features. Each log mel-spectrogram is split into 125 non-overlap small patches with the size of 64×4 along the time axis. SpecAugment [23] is applied to augment the input features during training.

Captions are tokenized and transformed to lower case with punctuation removed. To indicate the start and end of each caption, two special tokens "<sos>" and "<eos>" are padded. The vocabulary of AudioCaps contains 5277 distinct words.

4.3. Audio tagging pre-training

As proved in previous works, Transformer requires more training data to achieve competitive performance with CNNs [10]. However, the amount of training data in audio processing area is much less than that in computer vision. Cross-modal transfer learning from ImageNet pre-trained models to audio-related tasks proves to be effective [24]. Thus we make use of pre-trained DeiT models for image classification to initialize the parameters in ACT encoder [10, 20]. As images have three channels and spectrograms just have one channel, we take the average of the weights from the patch embedding layer in DeiT in order to adapt it for spectrogram.

As pre-trained audio neural networks (PANNs) proved to perform well in audio captioning [9], we pre-train ACT encoder on AudioSet as an audio tagging task in order to solve the data scarcity problem and learn more generalized audio patterns. Audio tagging is a multi-classification task of predicting the presence or absence of sound classes within an audio clip [25]. The class token output from the encoder is fed through a linear layer with sigmoid activation function to output the audio events probabilities. The model is trained to minimize the binary cross-entropy loss between the output of the model and the true label

$$\mathcal{L}_\mathrm{BCE}(\theta) = -\sum_{n=1}^{N}(y_n \cdot \ln f(x_n) + (1-y_n)\cdot \ln(1-f(x_n))) \tag{6}$$

where x_n is the n-th audio clip in AudioSet and N is the number of training samples. $f(x_n) \in [0,1]^K$ is the output of the model and $y_n \in \{0,1\}^K$ is the true label where K is the number of sound classes. The ACT encoder is pre-trained for 20 epochs with batch size of 128 and learning rate of 1×10^{-4}, which achieves a mean average precision (mAP) of 0.43 on the evaluation set of AudioSet dataset.

4.4. Experimental setups

We train the proposed model for 30 epochs using Adam optimizer [26] and a batch size of 32. The learning rate is linearly increased to 1×10^{-4} in the first five epochs using warm-up, which is then multiplied by 0.1 every 10 epochs. To mitigate over-fitting problem, dropout with rate of 0.2 is applied in the whole model. Label smoothing [27] with a smoothing factor of 0.1 is used to avoid over-confident prediction. We use beam search with a beam size up to 5 to improve the decoding performance during inference stage.

4.5. Evaluation metrics

In line with previous works, we evaluate our methods using machine translation and captioning metrics [13]. BLEU$_n$, ROUGE$_l$

Model	$BLEU_1$	$BLEU_2$	$BLEU_3$	$BLEU_4$	$ROUGE_L$	METERO	CIDEr	SPICE	SPIDEr
ACT_s_DeiT_AudioSet	0.643	0.483	0.352	0.249	0.469	0.218	0.669	0.160	0.415
ACT_m_DeiT_AudioSet	0.653	**0.495**	**0.363**	**0.259**	**0.471**	0.222	0.663	0.163	0.413
ACT_l_DeiT_AudioSet	0.647	0.488	0.356	0.252	0.468	0.222	0.679	0.160	0.420
ACT_m_scratch	0.567	0.411	0.285	0.191	0.417	0.187	0.501	0.127	0.314
ACT_m_DeiT	0.606	0.445	0.319	0.224	0.445	0.207	0.586	0.147	0.367
RNN+RNN [3]	0.614	0.446	0.317	0.219	0.450	0.203	0.593	0.144	0.369
CNN+RNN [6]	**0.655**	0.476	0.335	0.231	0.467	**0.229**	0.660	**0.168**	0.414
CNN+Transformer [9]	0.641	0.479	0.344	0.236	0.469	0.221	**0.693**	0.159	**0.426**
CNN+Transformer_scratch [9]	0.610	0.461	0.334	0.234	0.455	0.206	0.629	0.144	0.386

Table 2: Scores of the ACT model on the AudioCaps test set. DeiT: the ACT encoder is initialized with the parameters in DeiT, AudioSet: the ACT encoder is pre-trained on AudioSet.

and METEOR are machine translation metrics. $BLEU_n$ is a modified precision metric with a sentence-brevity penalty, calculated as a weighted geometric mean over different length n-grams. $ROUGE_l$ calculates F-measures by counting the longest common subsequence. METEOR evaluates a caption by computing a harmonic mean of precision and recall based on explicit word-to-word matches between the caption and given references. Captioning metrics contain $CIDE_r$, SPICE and $SPIDE_r$. $CIDE_r$ calculates the cosine similarity between term frequency inverse document frequency (TF-IDF) weighted n-grams. SPICE creates scene graphs for captions and calculates F-score based on tuples in the scene graphs. SPIDEr is the average of SPICE and $CIDE_r$ and is selected as the official ranking metric in DCASE challenge, the SPICE score ensures captions are semantically faithful to the audio content, while $CIDE_r$ score ensures captions are syntactically fluent.

5. RESULTS

5.1. Performance comparison

Table 2 presents the results on AudioCaps test set. We compare the proposed ACT model with three representative audio captioning models, "RNN+RNN" [3], "CNN+RNN" [6] and "CNN+Transformer" [9]. In these models, CNNs are all pre-trained on upstream audio-related tasks. As can be seen in Table 2 that the ACT model outperforms "RNN+RNN" model substantially in all evaluation metrics and achieves slightly higher scores than "CNN+RNN" model in most metrics. Compared with the state-of-the-art "CNN+Transformer" approach, ACT model outperforms it in machine translation metrics but gives slightly lower scores in $CIDE_r$. As machine translation metrics are mostly based on n-grams, these results show that the ACT model has better ability in generating words accurately. In addition, training an ACT model is faster than "CNN+Transformer" architecture, where the former takes less than five minutes for one epoch and "CNN+Transformer" needs seven minutes in our experiments. In summary, the ACT model shows competitive performance as compared to other state-of-the-art approaches, and it is simple as it is based only on the self-attention mechanism.

5.2. Ablation studies

The ablation studies are carried out to investigate the effectiveness of the pre-trained encoder and the influence of the hyper-parameters in the decoder. From the experimental results, we can see that pre-training the ACT encoder can boost the performance significantly. Even only using the pre-trained DeiT model, which is originally trained for image classification task, can bring significant performance gains in all the evaluation metrics. Pre-training on AudioSet as an audio tagging task further improves the system to approach the state-of-the-art performance. We also compare the ACT model with the "CNN+Transformer" model both trained from scratch, the results show that the ACT model performs worse than "CNN+Transformer" without encoder pre-training. These results suggest that pre-training the ACT encoder with a large dataset is important, and prove that Transformer network needs more training data than CNNs to achieve competitive performance.

We perform experiments on the three models with different numbers of layers and heads in the decoder. From the observations, the ACT model is slightly sensitive to the choice of hyper-parameters in the decoder. These three models achieve similar performance, among which ACT_m with four decoder layers performs better in machine translation metrics, while ACT_l achieves higher $CIDE_r$ and $SPIDE_r$ scores. The ACT model only needs shallow Transformer decoder layers compared to machine translation models in natural language tasks which typically contain 12 Transformer decoder layers [7]. There might be two reasons. First, the amount of training data in audio captioning is far less than data in natural language processing tasks. Second, the length of the audio captions are usually shorter than sentences in the natural language tasks.

6. CONCLUSION

We have presented a novel audio captioning model, Audio Captioning Transformer (ACT), which is a full Transformer model based on the self-attention mechanism. The encoder of the proposed ACT model can model the global and fine-grained information within an audio signal simultaneously, and has better ability to capture temporal relationships between audio events than CNNs. Experimental results show that the ACT model can outperform other state-of-the-art audio captioning systems in most metrics. Further research should be carried out to adapt the ACT model for audio clips of varied lengths.

7. ACKNOWLEDGMENT

This work is partly supported by grant EP/T019751/1 from the Engineering and Physical Sciences Research Council (EPSRC), a Newton Institutional Links Award from the British Council, titled "Automated Captioning of Image and Audio for Visually and Hearing Impaired" (Grant number 623805725) and a Research Scholarship from the China Scholarship Council (CSC) No. 202006470010.

References

[1] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 374–378.

[2] M. Wu, H. Dinkel, and K. Yu, "Audio caption: Listen and tell," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 830–834.

[3] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 119–132.

[4] S. Ikawa and K. Kashino, "Neural audio captioning based on conditional sequence-to-sequence model," *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop*, 2019.

[5] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[6] X. Xu, H. Dinkel, M. Wu, Z. Xie, and K. Yu, "Investigating local and global information for automated audio captioning with transfer learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 905–909.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[8] K. Chen, Y. Wu, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on Transformer and pretrained CNN," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop*, 2020, pp. 21–25.

[9] X. Mei, Q. Huang, X. Liu, G. Chen, J. Wu, Y. Wu, J. Zhao, S. Li, T. Ko, H. L. Tang *et al.*, "An encoder-decoder based audio captioning system with transfer and reinforcement learning," *arXiv preprint arXiv:2108.02752*, 2021.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[11] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017.

[12] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 736–740.

[13] A. Tran, K. Drossos, and T. Virtanen, "WaveTransformer: A novel architecture for audio captioning based on learning temporal and time-frequency information," *arXiv preprint arXiv:2010.11098*, 2020.

[14] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A Transformer-based audio captioning model with keyword estimation," *arXiv preprint arXiv:2007.00222*, 2020.

[15] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre-and post-processings for audio captioning," *arXiv preprint arXiv:2009.11436*, 2020.

[16] Y. Koizumi, Y. Ohishi, D. Niizumi, D. Takeuchi, and M. Yasuda, "Audio captioning using pre-trained large-scale language model guided by audio-based similar caption retrieval," *arXiv preprint arXiv:2012.07331*, 2020.

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[18] X. Liu, Q. Huang, X. Mei, T. Ko, H. L. Tang, M. D. Plumbley, and W. Wang, "CL4AC: A contrastive loss for audio captioning," *arXiv preprint arXiv:2107.09990*, 2021.

[19] X. Xu, H. Dinkel, M. Wu, and K. Yu, "A CRNN-GRU based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop*, 2020, pp. 225–229.

[20] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image Transformers & distillation through attention," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 347–10 357.

[21] W. Liu, S. Chen, L. Guo, X. Zhu, and J. Liu, "CPTR: Full Transformer network for image captioning," *arXiv preprint arXiv:2101.10804*, 2021.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[23] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[24] Y. Gong, Y.-A. Chung, and J. Glass, "PSLA: Improving audio event classification with pretraining, sampling, labeling, and aggregation," *arXiv preprint arXiv:2102.01243*, 2021.

[25] Q. Kong, C. Yu, Y. Xu, T. Iqbal, W. Wang, and M. D. Plumbley, "Weakly labelled audioset tagging with attention neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1791–1802, 2019.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

WAVEFORMS AND SPECTROGRAMS: ENHANCING ACOUSTIC SCENE CLASSIFICATION USING MULTIMODAL FEATURE FUSION

Dennis Fedorishin[1], *Nishant Sankaran*[1], *Deen Dayal Mohan*[1], *Justas Birgiolas*[2,3]
Philip Schneider[2], *Srirangaraj Setlur*[1], *Venu Govindaraju*[1]

[1] University at Buffalo, Center for Unified Biometrics and Sensors, USA,
{dcfedori, n6, dmohan, setlur, govind}@buffalo.edu
[2]ACV Auctions, USA, {jbirgiolas, pschneider}@acvauctions.com
[3]Ronin Institute, USA.

ABSTRACT

Acoustic scene classification (ASC) has seen tremendous progress from the combined use of convolutional neural networks (CNNs) and signal processing strategies. In this paper, we investigate the use of two common feature representations within the audio understanding domain, the raw waveform and Mel-spectrogram, and measure their degree of complementarity when using both representations for feature fusion. We introduce a new model paradigm for acoustic scene classification by fusing features learned from Mel-spectrograms and the raw waveform from separate feature extraction branches. Our experimental results show that our proposed fusion model significantly outperforms the baseline audio-only subnetwork on the DCASE 2021 Challenge Task 1B (increase of 5.7% in accuracy and a 12.7% reduction in loss). We further show that the learned features of raw waveforms and Mel-spectrograms are indeed complementary to each other and that there is a consistent improvement in classification performance over models trained on Mel-spectrograms or waveforms alone.

Index Terms— Audio classification, Acoustic scene classification, Feature fusion, Multi-modal features.

1. INTRODUCTION

Mel-spectrograms are the de-facto audio feature representation and have been widely used throughout the history of audio understanding [1]. Mel-spectrograms are created by calculating the short-time fourier transform (STFT) of an audio signal, then passing the STFT frequency responses through band-pass filters spaced on the Mel(logarithmic)-scale and often further passed through a logarithmic compression to replicate the human's non-linear perception of signal pitch and loudness, respectively.

With the advent of deep neural networks, many methods have been introduced that perform audio understanding tasks such as ASC, audio tagging, and sound event detection by using Mel-spectrogram representations of audio as the input to a convolutional neural network [2, 3]. Researchers have also explored other feature representations such as the gammatone and Constant-Q (CQT) spectrogram, and Mel Frequency Cepstrum Coefficients (MFCC) [4, 5]. [6] and found that fusing these representations allows for a network to learn complementary features, creating a stronger model for ASC.

In parallel, other works have utilized the raw waveform directly as input into neural networks, bypassing the need for hand crafted features [7, 8]. Waveform-based networks are trained end-to-end, while networks that utilize spectrograms need to create these hand crafted features that may often be sub-optimal for the given task. Regardless, many state of the art methods in ASC, speaker recognition, sound event detection, and other tasks still utilize spectrogram representations [9, 10]. Further, [11] introduced a fully learnable variation of spectrogram representations, where they are trained end-to-end to automatically find an optimized representation.

As a result, there is still no clear consensus as to the best feature representation that can perform strongly across various audio understanding tasks. Researchers are now looking at hybrid methods that use both waveform and spectrogram representations in a fusion setting. [8, 12] perform early feature map fusion of waveform and spectrogram features that are passed through convolutional layers for audio tagging and environmental sound classification. [13, 14] propose a decision-level ensembling of multiple models that utilize raw waveforms and Mel-spectrograms for environmental sound classification and ASC. Although these works have shown classification performance improvements using waveforms and spectrograms, they do not deeply explore the degree of complementarity and effects of fusing these features together.

In this paper, we investigate waveform and Mel-spectrogram feature fusion and propose a new ASC model that learns complementary features from both modalities using a more effective fusion method. We evaluate our proposed model using the DCASE 2021 Challenge Task 1B dataset to prove the effectiveness and complementarity of waveform and Mel-spectrogram feature fusion. Our work is reproducible and the code is publicly available.[1]

2. PROPOSED METHOD

To investigate and understand the complementarity between learning features from Mel-spectrograms and raw waveforms, we designed a fusion model based on two CNN feature extractors, and a unified classification layer. Figure 1 illustrates the design of our model. The spectrogram branch, F_s, is comprised of repeating 2D CNN blocks followed by a max pooling operation. The CNN blocks contain a convolution layer using a kernel size of 3×3, followed by a batch normalization and a Leaky ReLU nonlinear activation.

The waveform branch, F_w, is of a similar structure, however the two-dimensional max pooling and convolutional layers are replaced with one-dimensional kernels of size 8 and 7, respectively. In addition, the first convolutional layer in the waveform branch are parameterized to Sinc functions, as described in [15].

[1] https://github.com/denfed/wave-spec-fusion

Table 1: Detailed overview of proposed model design.

Spectrogram Branch F_s	
Input shape	[128,188]
Filter responses	32, 64, 128, 256
Global average pooling output l_s	$\langle 1024 \rangle$
Waveform Branch F_w	
Input shape	[48000]
Filter responses	32, 64, 128, 256
Global average pooling output l_w	$\langle 1024 \rangle$
Classification Layers F_c	
Input shape $(l_w + l_s)$	$\langle 1024 \rangle$
Dense layer outputs	$512, 256, 10$
Output classes	10

As both branches are working with different-sized input data, the feature responses may vary in size. We utilize global average pooling layers to condense both waveform and spectrogram features into a vector of 1024 units, denoted by l_w and l_s, respectively.

Feature fusion of both feature extraction branches is accomplished at the latent representation level, where features for both the waveform and spectrogram branch are extracted independently and then fused together into a unified representation. We hypothesize that this style of late fusion allows each feature extractor to extract more complementary features compared to related methods that fuse feature maps in early layers. In addition, late fusion may exploit more feature inter-dependencies of each modality compared to score fusion and ensembling methods that average independent model outputs.

Fusion is accomplished using element-wise summation such that the final latent representation $(l_w + l_s)$ has the same shape as its constituents. The classification layers, F_c, take $(l_w + l_s)$ as input and perform the final classification using repeating dense blocks, as shown in Figure 1. We use dropout layers with $p = 0.3$, followed by linear layers, a Leaky ReLU activation, and batch normalization. The classification $\hat{c}$ of the set of classes $c \in \mathbb{C}$ of an audio sample with its waveform x_w and Mel-spectrogram x_s can be described as:

$$\hat{c}_{(x_w, x_s)} = \underset{c \in \mathbb{C}}{\operatorname{argmax}} F_c(F_w(x_w) + F_s(x_s)) \quad (1)$$

Table 1 describes in detail the configuration of both feature extraction branches and the final classification layers. For our experiments, we utilize three variations of the described model to investigate modality complementarity:

Spectrogram sub-network: The spectrogram branch in Figure 1 is used independently with the classification layers, without the waveform branch. In this model, training is conducted only using Mel-spectrograms, omitting $F_w(x_w)$ from (1).

Waveform sub-network: The waveform branch in Figure 1 is used independently with the classification layers, without the spectrogram branch. In this model, training is conducted only using raw waveforms, omitting $F_s(x_s)$ from (1).

Fusion model: Both the spectrogram and waveform branch are trained end-to-end with their respective inputs. The latent representations of each branch are fused together for classification.

3. TRAINING CONFIGURATION

All models are trained using the SGD optimizer paired with the one-cycle learning rate scheduler described in [16]. The one-cycle learning rate scheduler anneals an initial learning rate to a maximum value, then anneals it back to the initial learning rate, over the entire training procedure. [17] showed that this procedure leads to faster training times. In addition, using large learning rates for a portion of the training procedure acts as a form of regularization. We experimented with various learning rates and found that a max learning rate of 0.008 for the one-cycle scheduler works best for the proposed models. Training batch size is set to 128 and the models are trained for 50 epochs. The models were trained on an RTX 6000 GPU with the most complex model taking 1.5 hours to fully train. We train our models and conduct our experiments using the provided training and validation split of the Task 1B challenge dataset.

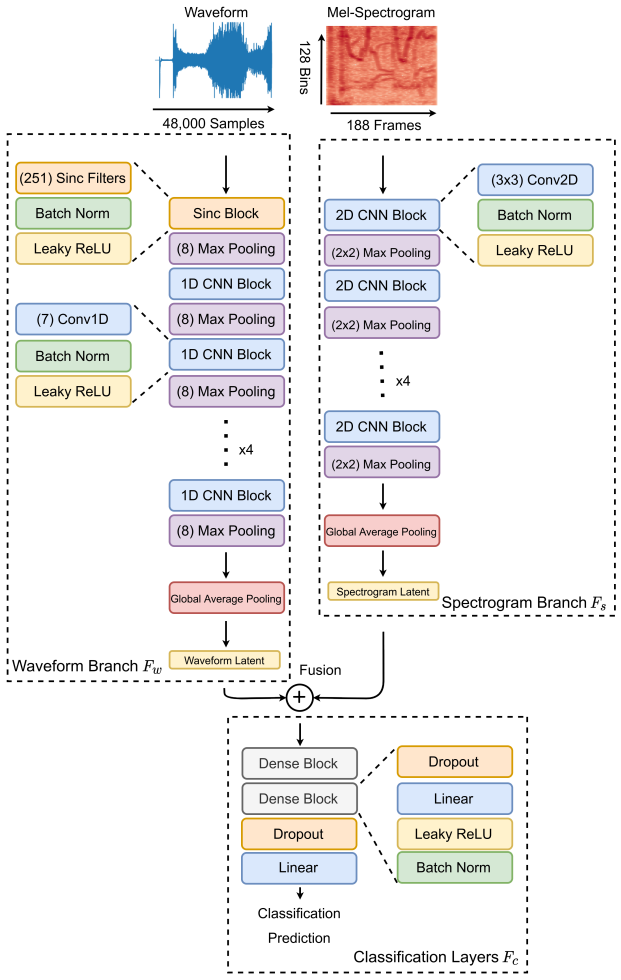


Figure 1: Illustration of the proposed fusion model.

3.1. Dataset: DCASE 2021 Challenge Task 1B, Audio-Visual Scene Classification

Task 1B is based on the TAU Audio-Visual Urban Scenes 2021 dataset, a dataset containing synchronized audio and video recordings from 12 European cities in 10 different scenes. Audio is recorded using a Soundman OKM II Klassik/studio A3 microphone paired with a Zoom F8 audio recorder, sampled at $48kHz$ at a 24-bit resolution. Video is recorded using a GoPro Hero5 Session. The

Table 2: Kernel parameterization performance.

Parameterization	Accuracy %	Log Loss
Unparameterized (normal)	61.87	1.047
Sinc parameterization [15]	**64.79**	**1.045**

Table 3: Model performance compared to challenge baseline.

Model	Accuracy %	Log Loss	# Params
Audio baseline [20]	65.1	1.048	-
Waveform sub-network	64.79	1.045	1.0M
Spectrogram sub-network	66.46	1.072	1.1M
Fusion Model	**70.78**	**0.915**	1.4M

dataset contains 12,292 10-second samples of each modality spread across the 10 scenes. The provided train/validation split consists of 8,646 samples in the training set and 3,645 samples in the validation set [18]. In this paper, we focus on the audio portion of the dataset only, excluding using videos for scene classification.

3.2. Data Preprocessing

We input the raw waveform and its generated Mel-spectrogram into their respective feature extractors. According to the Task 1B rules, we split the development dataset samples into 1 second audio files to perform classification at the 1 second level. This brings the training dataset to 86,460 samples and the validation dataset to 36,450 samples. Audio files are sampled at $48kHz$ and therefore have a sample length of $[48000]$. In addition, the audio waveforms are scaled to the range $[0, 1]$. Mel-spectrograms are generated using 128 frequency bins, a hop length of 256 samples, and a Hann window size of 2048 samples, creating a final size of $[128 \times 188]$. The Mel-spectrograms are also passed through a logarithmic compression and then normalized at an instance level using Z-Score normalization such that each sample has a mean of 0 and unit standard deviation.

3.3. Data Augmentation

For data augmentations, we utilize Mixup [19] and time shifting for all experiments. During training, Mixup has a 50% probability of being used for each batch and time shifting is applied to every batch. For Mixup, we select $\alpha = 0.2$ and apply it to both the sub-networks and fusion model. For the fusion model, Mixup is performed evenly across the waveform and spectrogram such that two audio samples' waveform and spectrogram are mixed together at the same mixing ratio. Time shifting shifts the waveform and spectrogram along the time axis, where overrun samples are shifted to the opposite end of the input. We time shift randomly from 0% to 50% of the time axis size for both the waveform and spectrogram. For the fusion model, time shifting is applied independently to the modalities, such that both the waveform and spectrogram may be shifted by varying degrees. We experimented with various configurations and found that this configuration achieves the highest classification performance, however further research should be conducted on the effects of consistent and inconsistent data augmentations between each modality.

4. EXPERIMENTAL RESULTS

4.1. Waveform Kernel Parameterizations

SincNet [15] introduced the use of parameterized Sinc filters for speech recognition, where the kernels of the first convolutional layer of a model utilizing waveforms are replaced with kernels parameterized to the Sinc function. Works such as [14] have applied these filters for ASC and found Sinc filters to improve ASC performance.

Table 4: Fusion method comparisons.

Model	Accuracy %	Log Loss	# Params
Wavegram-Logmel-CNN	68.35	1.063	80.2M
Decision fusion	68.65	0.955	2.0M
Decision ensemble	70.47	**0.845**	2.0M
Proposed late fusion	**70.78**	0.915	1.4M

Table 2 shows model performance when replacing the first convolutional layer of the waveform branch with parameterized Sinc kernels instead of an unparameterized, fully learnable kernel. As shown, using sinc kernels outperforms unparameterized kernels. We hypothesize that the initialization of the sinc filters to mirror the distribution of the Mel-scale, as described in [15], are a more optimal initialization compared to conventional kernel initializations. In addition, the Sinc kernels are less prone to overfitting as they are constrained to the Sinc function [11]. Using Sinc filters also allows us to reduce model complexity, as two frequency cutoff values are learned per kernel instead of N parameters of a size N kernel.

4.2. Waveform and Spectrogram Feature Fusion

Table 3 shows the classification performance of the provided Task 1B baseline model compared to the three different model variations proposed. The waveform sub-network is not able to outperform the baseline while the spectrogram sub-network performs slightly better than the baseline in accuracy. The fusion model outperforms both the baseline and models trained on single modalities with a 5.7% improvement in accuracy and a reduction of .13 in loss over the baseline. Furthermore, we see that the fusion model outperforms the spectrogram sub-network by 4.3% in accuracy and a .16 reduction in loss. This improvement shows that there are features being learned within the raw waveform that are complementary to features being learned from the Mel-spectrogram, resulting in a more discriminative classification model.

4.3. Feature Fusion Design

We perform a comparison with other fusion paradigms to better understand its significance in fusing waveform and spectrogram features. We compare our method against the Wavegram-Logmel-CNN, a popular acoustic classification model introduced by [12] that performs early feature map fusion on the waveform and spectrogram. We train the Wavegram-Logmel-CNN using the training configuration described in [12]. The same data preprocessing is used as described in section 3.2, however the spectrogram hop length is changed to 320 samples to fit the structure of the model. In addition, we compare the proposed late fusion design against a decision fusion and ensembling method. Instead of latent vector fusion, we fuse the independent sub-network's predictions at the decision level by averaging predictions together. Comparing to (1), the decision fusion classification resembles:

$$\hat{c}_{(x_w, x_s)} = \underset{c \in \mathbb{C}}{\operatorname{argmax}} \frac{1}{2}(F_{c_w}(F_w(x_w)) + F_{c_s}(F_s(x_s))) \quad (2)$$

where F_{c_w} and F_{c_s} depict the waveform and spectrogram subnetwork's classification layers, respectively. We train this decision fusion model using the same configuration as the proposed late fusion model. In addition, we compare the fusion methods against an ensemble of the independently trained sub-networks, using decision averaging described in (2).

As shown in Table 4, the proposed late fusion model outperforms the Wavegram-Logmel-CNN with significantly fewer parameters, in addition outperforming the decision fusion and ensemble

Table 5: Comparison of feature fusion methods.

Fusion Method	Accuracy %	Log Loss	# Params
Element-wise sum	70.78	**0.915**	1.4M
Concatenation	**70.85**	0.924	1.9M
MFB [21]	70.13	0.943	7.6M

Table 6: Feature branch removal ablation study.

Model	Accuracy %	Log Loss
Spectrogram sub-network	66.46	1.072
Fusion spectrogram branch only	51.33	1.720
Waveform sub-network	64.79	1.045
Fusion waveform branch only	31.51	2.500

model. We see that while the ensemble of both sub-networks performs well, the late fusion model is able to extract feature interdependencies of the waveform and spectrogram that still outperform the ensemble model in terms of accuracy. The late fusion model also has fewer parameters and is trained end-to-end.

4.4. Latent Representation Fusion Methods

Most approaches to feature fusion utilize linear methods such as element-wise summation and concatenation of vectors and feature maps. A more advanced operation, Multimodal Factorized Bilinear Pooling [21], has been used within visual question answering and captures more expressive features than linear methods while being less computationally expensive than conventional bilinear pooling.

We experiment using these fusion methods to see whether we can fuse features in a more expressive fashion. Table 1 and Figure 1 depict the design for element-wise summation fusion. For concatentation, latent vectors l_w and l_s are combined to a final size of 2048 units. This new vector is passed into the classification layers, with the dense layers outputting $1024, 512, 10$ units, respectively. For MFB fusion, we set $k = 3$ and $o = 1024$, as described in [21]. The MFB fusion model has the same design as Figure 1, but the element-wise summation operation is replaced with MFB.

Table 5 shows the performance of our fusion model when utilizing element-wise sum, concatenation, and MFB. All methods perform similarly, however element-wise summation produces the lowest validation loss model. Fusion by concatenating latent vectors results in the highest accuracy model. We select element-wise summation fusion as it produced the lowest loss in addition to it being the least computationally expensive operation.

5. ABLATION STUDIES

Although we see a classification performance improvement when fusing waveform and spectrogram features, we must validate that the improvement is from complementary features extracted from both modalities. It may be the case that the feature extraction branches are underparameterized, and when adding more parameters the model performs better solely due to the increase in parameterization and not the second modality. To test this hypothesis, we expand both sub-networks such that their number of parameters exceed the fusion model by doubling each of the CNN block filter responses, classification layer responses, and increasing latent vectors to 2048 units. As shown in Table 7, both sub-networks were unable to surpass the performance of the fusion model, showing that the added performance in the fusion model is from the added modality.

To further understand the differences of each sub-network's performance, we compare each sub-network to their equivalent sub-network trained in the fusion setting. Examining the performance drop when removing feature extraction branches F_w and F_s in the fusion model may give clues into how the branches train alone versus in the fusion setting. The trained waveform and spectrogram sub-networks depicted in Table 3 are compared to the fusion model's respective sub-network. As shown in Table 6, the sub-networks that are trained in the fusion setting have a substantial performance loss when removing the opposite sub-network, far below the performance of the respective sub-network that is trained independently. We infer that when trained end-to-end, each of the sub-networks in the fusion model learn to focus on disparate features that when fused together, improve classification performance.

Table 7: Parameterization ablation study.

Model	Accuracy %	Log Loss	# Params
Fusion model	**70.78**	**0.915**	1.4M
Large spectrogram sub-network	66.48	1.043	4.2M
Large waveform sub-network	63.44	1.041	3.9M

Table 8: Class-Wise losses of the fusion model.

Class-Wise loss	Fusion	Fusion spec. branch only	Fusion wave. branch only
Airport	**0.901**	1.226	3.441
Shopping Mall	**0.944**	0.995	1.612
Metro Station	**1.053**	2.030	1.827
Street Pedestrian	1.104	**1.069**	2.638
Public Square	1.321	1.384	**0.663**
Street Traffic	**0.424**	0.843	3.038
Tram	**0.899**	2.106	4.182
Bus	0.747	4.905	**0.723**
Metro	**1.145**	2.825	4.324
Park	0.535	**0.443**	2.913

We also investigate class-wise loss when removing each branch of the fusion model, as shown in Table 8. Most classes have the lowest loss in the fusion model, however when removing the waveform branch, the spectrogram branch has a lower loss for the Street Pedestrian and Park class. When removing the spectrogram branch, the waveform branch has a lower loss for the Public Square and Bus class. We infer that while the fusion model can generally capture complementary features from each modality, the fusion operation is not able to exploit the full degree of complementarity of each branch's features. A fusion method that can fully exploit the modality complementarity would further improve ASC performance.

6. CONCLUSION

In this paper, we investigate feature fusion of two common audio representations, the raw waveform and Mel-spectrogram, and show that there are complementary features being learned that improve ASC performance. Further, we explore various fusion methods and experimentally validate that the proposed late fusion model is able to outperform other feature fusion designs. Our proposed fusion model utilizes these features to significantly outperform the DCASE 2021 Challenge Task 1B audio baseline and achieve 2$^{\text{nd}}$ place against the audio-only submissions. In future work, we will investigate more fusion methods to better exploit waveform and spectrogram feature complementarity and explore the effects of using independent data augmentations on the separate modalities.

7. ACKNOWLEDGMENT

This work was supported by the Center for Identification Technology Research (CITeR) and the National Science Foundation (NSF) under grant #1822190

8. REFERENCES

[1] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[2] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[3] N. Turpault, R. Serizel, A. Parag Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *2019 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, New York City, United States, October 2019.

[4] L. Pham, H. Phan, T. Nguyen, R. Palaniappan, A. Mertins, and I. McLoughlin, "Robust acoustic scene classification using a multi-spectrogram encoder-decoder framework," *Digital Signal Processing*, vol. 110, p. 102943, 2021.

[5] Y. Su, K. Zhang, J. Wang, D. Zhou, and K. Madani, "Performance analysis of multiple aggregated acoustic features for environment sound classification," *Applied Acoustics*, vol. 158, p. 107050, 2020.

[6] H. Wang, Y. Zou, and D. Chong, "Acoustic scene classification with spectrogram processing strategies," in *2020 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, Tokyo, Japan, November 2020, pp. 210–214.

[7] T. Kim, J. Lee, and J. Nam, "Sample-level cnn architectures for music auto-tagging using raw waveforms," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 366–370.

[8] B. Zhu, C. Wang, F. Liu, J. Lei, Z. Huang, Y. Peng, and F. Li, "Learning environmental sounds with multi-scale convolutional neural network," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[9] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Conformer-based sound event detection with semi-supervised learning and data augmentation," in *2020 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, Tokyo, Japan, November 2020, pp. 100–104.

[10] L. Sarı, K. Singh, J. Zhou, L. Torresani, N. Singhal, and Y. Saraf, "A multi-view approach to audio-visual speaker verification," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6194–6198.

[11] N. Zeghidour, O. Teboul, F. d. C. Quitry, and M. Tagliasacchi, "LEAF: A learnable frontend for audio classification," *arXiv preprint arXiv:2101.08596*, 2021.

[12] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNS: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[13] S. Li, Y. Yao, J. Hu, G. Liu, X. Yao, and J. Hu, "An ensemble stacked convolutional neural network model for environmental event sound recognition," *Applied Sciences*, vol. 8, no. 7, 2018.

[14] J. Huang, H. Lu, P. Lopez Meyer, H. Cordourier, and J. Del Hoyo Ontiveros, "Acoustic scene classification using deep learning-based ensemble averaging," in *2019 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, New York University, NY, USA, October 2019, pp. 94–98.

[15] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 1021–1028.

[16] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 1100612.

[17] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[18] S. Wang, A. Mesaros, T. Heittola, and T. Virtanen, "A curated dataset of urban scenes for audio-visual scene analysis," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[19] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[20] S. Wang, T. Heittola, A. Mesaros, and T. Virtanen, "Audio-visual scene classification: analysis of dcase 2021 challenge submissions," *arXiv preprint arXiv:2105.13675*, 2021.

[21] Z. Yu, J. Yu, J. Fan, and D. Tao, "Multi-modal factorized bilinear pooling with co-attention learning for visual question answering," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2017, pp. 1821–1830.

TRANSFER LEARNING FOLLOWED BY TRANSFORMER FOR AUTOMATED AUDIO CAPTIONING

Hyejin Won[1], *Baekseung Kim*[1], *Il-Youp Kwak*[1*], *Changwon Lim*[1†]

[1] Chung-Ang University, Department of Applied Statistics, Seoul, South Korea,
{whj9492, kbs778, ikwak2, clim}@cau.ac.kr

ABSTRACT

Automated Audio Captioning (AAC) automatically creates captions that can explain the given audio sound data using machine learning techniques. Researchers investigate the solutions for AAC on DCASE 2021 audio captioning challenge. In the challenge, a model is required to generate natural language descriptions of a given audio signal. We use pre-trained models trained using AudioSet data, a large-scale dataset of manually annotated audio events. A large amount of audio events data would help capturing important audio feature representation. To use the learned feature from AudioSet data, we utilize CNN14 or ResNet54 network pre-trained on AudioSet, which achieved state-of-the-art audio pattern recognition performance. Our proposed sequence-to-sequence model consists of a CNN14 or ResNet54 encoder and a Transformer decoder. Experiments show that the proposed model can achieve a SPIDEr score of 0.246 and 0.285 on audio captioning performance. We further experiment the use of three different voice features, log-mel spectrogram, constant Q transform spectrogram, and gammatone filter spectrogram.

Index Terms— Automated audio captioning, Acoustic event detection, Transfer learning, Transformer

1. INTRODUCTION

Automated Audio Captioning (AAC) automatically creates captions that can explain the given audio sound data using machine learning techniques. Many researchers investigated this exciting problem from DCASE challenges and workshops in 2020 and 2021 [1, 2, 3, 4, 5, 6]. One example of a generated caption on a given sound could be "people talking in a small and empty room." The 2021 DCASE AAC uses the Clotho v2.1 dataset [7]. Clotho v2.1 data contains 6,974 (4,981 from version 1 and 1,993 from version 2.1) audio clips in 15-30 seconds each with 5 captions in 8-20 English words.

In the DCASE 2020 competition on AAC, Drossos et al. (2017) [1] introduce an encoder-decoder structured baseline for the AAC task. The encoder-decoder structure is the most widely used architecture for AAC. The baseline model has an encoder-decoder scheme with a multi-layered, bi-directional GRU encoder and multi-layered decoder. Takeuchi et al. (2020) [2] achieve the top performance utilizing data augmentation, multi-task learning, and post-processing with an LSTM decoder. Chen et al. (2020) [3] proposes a pre-training stage for the encoder and combines the transformer decoder to achieve the second-best model. Perez-Castanos et al. (2020) [5] experiment resnet encoder and LSTM decoder with gammatone feature as an input. Pellegrini (2020) [4] propose Listen-Attend-Spell(LAS) architecture with listener encoder and speller decoder. Xu et al. (2020) [6] use a CRNN encoder and a GRU decoder with fine-tuning by reinforcement learning.

This year, the use of external data is allowed. Researchers can train the important sound-related feature representation using a massive amount of data such as AudioSet [8, 9]. Koizumi et al. (2020) [10] utilize a pre-trained VGGish model with a transformer decoder. Xu et al. (2021) [11] propose pre-trained CNN10 and CRNN5 as encoder networks with GRU decoder.

In this paper, we propose transfer learning followed by transformer architecture. With the transfer learning, our proposed model takes two pre-trained networks, 14-layers CNN (CNN14) and 54-layers ResNet (ResNet54), trained on AudioSet as the encoder part [12]. Those pre-trained networks achieve state-of-the-art performance on audio pattern recognition, and we expect our encoder network to compress important audio representation very well. Further, we train a transformer decoder using the Clotho v2.1 dataset for natural language generation. Finally, we experimented with three different audio feature preprocessing methods (log-mel spectrogram, CQT spectrogram, and gammatone filter spectrogram) in the ablation study.

2. PROPOSED METHOD

2.1. System Overview

Figure 1 describes the overview of our proposed system. The pre-trained CNN14 and ResNet54 are taken as an encoder using transfer learning [12]. We used a transformer decoder and trained our model using the Clotho v2.1 dataset. In the training stage, we experiment with three scenarios, 1) training all parameters from scratch without transfer learning, 2) fine-tuning the last block of the pre-trained network, and 3) training without fine-tuning the last block of the pre-trained network. After the training stages, we evaluate the performance of our system using a development set.

2.2. Pre-trained Audio Neural Networks using AudioSet

Kong et al. (2020) [12] propose Pre-trained Audio Neural Networks(PANNs) trained on the large-scale AudioSet dataset and make the 15 pre-trained models available to the public, including CNN14 and ResNet54. The AudioSet dataset includes over 5,000 hours of audio with 527 sound labels [8]. The audio clips from AudioSet data are extracted from YouTube videos. The training

*This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (2020R1C1C1A01013020)

†This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (2021R1F1A1056516)

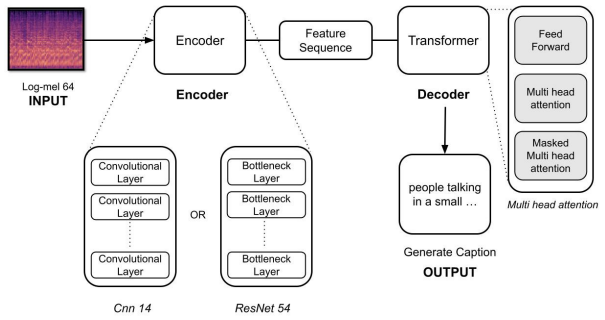


Figure 1: Model Architecture

dataset consists of 2,063,839 audio files, including a "balanced subset" of 22,160 audio files, where there are at least 50 audio files for each sound class. The evaluation dataset consists of 20,371 audio files. Audio files are padded to 10 seconds with silence if they are shorter than 10 seconds. The pre-trained models (CNN14 and REsNet54) are multi-label classification models for the 527 sound classes achieving state-of-the-art performance. These PANNs can be transferred to other audio-related tasks. We take CNN14 and ResNet54 as the encoder part for the AAC model. Table 1 and 2 describe CNN14 and ResNet54 model architecture, respectively.

2.3. Encoder-Decoder

Our proposed model also has the traditional encoder-decoder structure for AAC. Our model uses CNN14 or ResNet54 as an encoder and Transformer Decoder for natural language generation. The CNN14 and ResNet54 models were pre-trained networks (CNN14, ResNet54) learned from PANNs to AudioSet. We freeze the weights taken from the pre-trained networks and train the Transformer decoder with the Clotho data. Furthermore, we attempted to fine-tune the encoder network by unfreezing the last convolution block layers of CNN14 and ResNet54 to find the optimal model.

2.3.1. Encoder

Our model used CNN14 and Resnet54 as an encoder for feature extraction of input log-mel spectrogram [12]. CNN14 and Resnet54 models are pre-trained models from AudioSet, showing the highest mean average precision (mAP)[1] among the available pre-trained models. CNN14 and ResNet54 had the state-of-the-art mAP of 0.431 and 0.429, respectively [12]. Tables 1 and 2 show the structure of the CNN14 and ResNet54 that we used for ACC, respectively. The number after the "@" symbol indicates the number of feature maps. BottleneckB is an abbreviation for bottleneck block.

In Table 1, the 14-layer CNN consists of four convolution blocks, each having two 3 × 3 convolution layers with ReLU activation function and batch normalization, with an 2 × 2 average pooling layer between the blocks. The number of channels in the convolution blocks is 64, 128, 256, 512, 1024,2048, respectively.

[1] Average precision (AP) is defined as the area under the recall-precision curve of a specific class. The mean average precision (mAP) is the average value of AP over all classes.

Table 2 describes the ResNet54 architecture inspired by He et al. (2016) [13] for Audio tagging. Two convolutional layers and a downsampling layer are applied on the log-mel spectrogram to reduce the input log-mel spectrogram size. Additionally, We had three bottleneck blocks with 64 filters, four bottleneck blocks with 128 filters, 6 bottleneck blocks with 256 filters, and 3 bottleneck blocks with 512 filters. Finally, two 3 x 3 convolutions are applied. The BottleneckB in Table 2 is an abbreviation of bottleneck block.

Table 1: CNN14 architecture

CNN14
Log-mel spectrogram 64 mel bins
$(3 \times 3$ @64,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @128,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @256,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @512,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @1024,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @2048,BN,ReLU$)*2$

Table 2: ResNet54 architecture

ResNet54
Log-mel spectrogram 64 mel bins
$(3 \times 3$ @512,BN,ReLU$)\times 2$
Pooling 2×2
(bottleneckB@64)$\times 3$
Pooling 2×2
(bottleneckB@128)$\times 4$
Pooling 2×2
(bottleneckB@256)$\times 6$
Pooling 2×2
(bottleneckB@512)$\times 3$
Pooling 2×2
$(3 \times 3$ @512,BN,ReLU$)\times 2$

2.3.2. Decoder

Figure 2 describes our transformer decoder architecture. It uses a standard transformer decoder consisting of multi-head self-attention as a decoder. The decoder uses a 2-layers transformer with a hidden dimension of 192 and 4 heads.

The input data of our decoder is the word embedding feature pre-trained using the word2vec model. The positional encoding is further applied. Next, it passes to the masked multi-head attention module and returns a query vector for the following multi-head attention module. The key and value vectors for the multi-head attention module are taken from the output of the encoder (CNN14 or ResNet54) network. After following the feed-forward network, we can get the output of the transformer block. The transformer block iterates two times, and then the output is fed into a dense layer and a softmax function to generate output probabilities of the caption words.

Table 3: Score for model performance on evaluation data

Model	$BLEU_1$	$BLEU_2$	$BLEU_3$	$BLEU_4$	$ROUGE_L$	METEOR	CIDEr	SPICE	**SPIDEr**
Baseline Model	0.378	0.119	0.050	0.017	0.263	0.078	0.075	0.028	0.051
CNN14 + Transformer (From Scratch)	0.466	0.262	0.156	0.092	0.309	0.137	0.208	0.087	0.148
ResNet54 + Transformer (From Scratch)	0.459	0.253	0.152	0.084	0.312	0.131	0.182	0.085	0.133
CNN14 + Transformer (Transfer-learning with fine-tuning)	0.552	0.364	0.244	0.159	0.378	0.168	0.395	0.118	0.257
ResNet54 + Transformer (Transfer-learning with fine-tuning)	0.546	0.358	0.239	0.156	0.373	0.166	0.379	0.113	0.246
CNN14 + Transformer (Transfer-learning)	0.564	0.376	0.254	0.163	0.388	0.177	0.441	0.128	0.285
ResNet54 + Transformer (Transfer-learning)	0.540	0.345	0.230	0.152	0.361	0.161	0.383	0.109	0.246

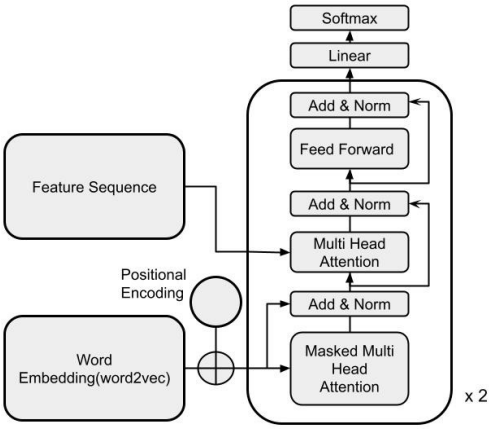


Figure 2: Transformer decoder

3. EXPERIMENTS

3.1. Dataset and Data Pre-processing

Clotho contains audio clips of CD quality (44.1 kHz sampling rate, 16-bit sample width) and five captions for each audio clip. The time duration of the audio clips ranges from 15 to 30 seconds, and the amount of words in each caption ranges from eight to 20 words. Clotho provides three splits for developing audio captioning methods, namely development, evaluation, and testing. The development and evaluation splits are freely available online, while the testing split is withheld for scientific challenges. In this work, we employ the development and evaluation splits of Clotho, having 2893 and 1045 audio clips, yielding 14465 and 5225 captions, respectively. We choose Clotho because it is built to offer audio content diversity, and extra care has been taken for eliminating spelling errors, named entities, and speech transcription in the captions. Additionally, Clotho is already employed at the DCASE 2020 audio captioning task6 [7].

The log-mel spectrogram feature is used for the input feature[14]. Audio data has 44.1kHz sampling frequency, and we apply a Hann window of 1024 size with 50% overlaps. From each window frame, we extract 64 log mel-band energies. We calculate the maximum time window number, T, among sample datasets for the number of time windows. We pad zero to the time dimension to have a fixed size T for the input feature on our model.

The word embedding is pre-trained using the Word2Vec model [15] via python package gensim [16]. Each caption sentence in the training set is used to form a training corpus.

Spec Augment [17] is applied as a data augmentation method for more robust training. With the Spec Augment, frequency masks and time masks are randomly applied onto the log-mel spectrogram before we feed the log-mel spectrogram input to the CNN14 or ResNet54 encoder.

3.2. Hyper-parameters and Training Procedure

3.2.1. Hyper-parameters

In training, a batch size of 8 is used with a learning rate of 10^{-4}. An l2 regularization is applied to all trainable parameters with factor λ = 10^{-6}. We use the Adam Optimizer [18] and apply the Stochastic Weight Averaging (SWA) method [19] to boost performance. Dropout in $P = 0.2$ is applied to ResNet54 encoder and Transformer decoder.

3.2.2. Training procedure

The training procedure consists of three parts. 1) Transfer learning step for encoder network, 2) training transformer decoder network while freezing the pre-trained encoder network, and 3) fine-tuning unfreezing last convolutional block parameters from encoder network.

We transfer two pre-trained networks, CNN14 and ResNet54, as our encoder network in the transfer learning stage. These two models are trained using log-mel spectrogram features using a large amount of AudoSet data. We freeze the encoder network for the following training stage.

In the training transformer decoder stage, we utilize the Clotho data. Each audio is combined with each one of five caption annotations and used as a sample. Each audio is used as one sample in the evaluation, and all five captions are used as a reference for metric computation. We used the 64 mel-band log-mel spectrogram of the audio as our input feature and then converted the amplitude into a decibel scale. A beam search with a beam size of 3 is implemented to achieve better decoding performance in the inference

Table 4: Ablation studies SPIDEr Score

Feature	Scratch	Transfer learning
Log-mel spectrogram	0.148	0.285
CQT spectrogram	0.15	-
gammatone spectrogram	0.2	-

stage. The Word2Vec model is trained with 1000 epochs, and the proposed model is trained using 30 epochs.

Finally, in the fine-tuning stage, we unfreeze the last convolution blocks of pre-trained networks, CNN14 and ResNet54. The fine-tuning is performed for 30 more epochs.

3.3. Evaluation Metrics

For the assessment of the performance of our method, we employ The proposed metrics from the audio captioning task at DCASE 2021 challenge. These metrics can be divided into two categories.

Firstly there are machine translation metrics, which are BLEU_n [20], ROUGE_L [21],and METEOR [22]. BLEU is a precision-based metric. It calculates a weighted geometric mean of a modified precision of n-grams between predicted and ground truth captions. Due to the calculation of the modified precision that favors short predicted captions, BLEU uses a penalty in the calculation of the geometric mean. This penalty penalizes predicted captions that are shorter than the ground truth. Typical lengths for n-grams are one to four, resulting in BLEU_n ($n \in \{1, 2, 3, 4\}$), respectively [20, 23]. ROUGE_L [21] is a Longest Common Subsequence (LCS) based metric. It calculates an F-measure using LCS between the predicted and ground truth caption. The F-measure is oriented towards recall using a value for the $\beta = 1.2$ in the F-measure calculation [21, 23]. METEOR [22] calculates a harmonic mean of precision and recall of segments of the captions between the predicted and ground truth captions. The recall is weighted significantly more than precision, and thus METEOR is considered a recall-based metric [23]. It employs alignment between the words of the predicted and ground truth captions and matches exact words, stems of words, synonyms, and paraphrases. The alignment is computed over segments of the captions between the ground truth and predicted captions while minimizing the number of chunks needed.

Then, the captioning metrics are CIDEr [24], SPICE [25], and a linear combination of these two metrics called SPIDEr [26]. CIDEr calculates a weighted sum of the cosine similarity between the predicted and ground truth captions for n-grams of length n with $\in [1, 4]$. The cosine similarity is calculated using Term Frequency Inverse Document Frequency (TF-IDF) weighting for each n-gram [24, 23]. SPIDEr is the average of CIDEr and SPICE, and it evaluates both fluency and semantic properties of the predicted captions.

3.4. Experimental Results

We experiment with three different scenarios, 1) training all parameters from scratch without transfer learning, 2) fine-tuning the last block of the pre-trained network, and 3) training without fine-tuning the last block of the pre-trained network. Table 3 shows the experimental results from the three different scenarios and the baseline. All scenarios of CNN14 Encoder+Transformer Decoder and ResNet54 Encoder+Transformer Decoder models have higher scores than the baseline model in all evaluation metrics. Also, All transfer learning scenarios have better performance than the training from the scratch scenario. This shows that the transferred encoders trained with sufficiently large amounts of audio data perform well on AAC. Between CNN14 and ResNet54 encoders, CNN14 encoder performed better than ResNet54 encoder in all evaluation metrics. We also experiment with fine-tuning the last convolution block of encoder networks (CNN14 and ResNet54 models). For CNN14 Encoder + Transformer Decoder, the transfer-learning model without fine-tuning works better than with the fine-tuning scenario in all evaluation metrics. On the other hand, for ResNet54 Encoder + Transformer Decoder, the model with fine-tuning has better performance with BLEU, ROUGE_L, and METEOR metrics which are machine translation metrics. However, for the captioning metrics such as SPICE, CIDEr, and SPIDEr scores, the result is inconclusive in that the models with fine-tuning have better SPICE scores but have worse CIDEr scores and have similar SPIDEr scores.

3.5. Ablation Studies

We have tested three feature extraction methods, 1) log-mel spectrogram, 2) constant Q transform (CQT) spectrogram, [27] and 3) Gammatone filter spectrogram. The Constant-Q-Transform (CQT) is a time-frequency representation where the frequency bins are geometrically spaced, and the Q-factors (ratios of the center frequencies to bandwidths) of all bins are equal. Gammatone filter spectrogram is computed by decomposing the input speech signal into the time-frequency (T-F) domain using a bank of Gammatone filters, followed by a down-sampling operation of the filter-bank responses along the time dimension [28].

We compare these three features without transfer learning because there are no pre-trained models with CQT and gammatone features. Table 4 shows the SPIDEr score performance using three different features. The model using CQT spectrogram (0.148) has similar SPIDEr performance with the model using log-mel spectrogram (0.15). However, the gammatone spectrogram (0.2) model performs better than the model using the log-mel spectrogram. Possibly, we may have better performance with the gammatone spectrogram feature if the pre-trained model using the gammatone spectrogram is available.

4. CONCLUSION

The DCASE community is hosting competitions for better AAC models in 2020 and 2021. In 2021, the use of external data is allowed. Thus, we propose transfer learning followed by a transformer approach. We adopt CNN14 and ResNet54 pre-trained on AudioSet data because it achieves state-of-the-art performance on audio pattern recognition. The pre-trained CNN14 or ResNet54 models are taken as encoder networks for informative audio feature extraction. With the transferred encoder and a transformer decoder, our proposed systems outperform the baseline system with all evaluation metrics. Further, we experiment with three training scenarios, 1) from scratch, 2) transfer learning, and 3) transfer learning with fine-tuning. Among them, the transfer learning of CNN14 encoder without fine-tuning works the best, achieving a SPIDEr score of 0.285.

5. REFERENCES

[1] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 374–378.

[2] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre- and post-processings for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, pp. 190–194.

[3] K. Chen, Y. Wu, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on transformer and pre-trained cnn," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, pp. 21–25.

[4] T. Pellegrini, "IRIT-UPS DCASE 2020 audio captioning system," DCASE2020 Challenge, Tech. Rep., June 2020.

[5] S. Perez-Castanos, J. Naranjo-Alcazar, P. Zuccarello, and M. Cobos, "Listen carefully and tell: An audio captioning system based on residual learning and gammatone audio representation," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, pp. 150–154.

[6] X. Xu, H. Dinkel, M. Wu, and K. Yu, "A crnn-gru based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE2020)*, 2020, pp. 225–229.

[7] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 736–740.

[8] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[9] Q. Kong, C. Yu, Y. Xu, T. Iqbal, W. Wang, and M. D. Plumbley, "Weakly labelled audioset tagging with attention neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1791–1802, 2019.

[10] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A transformer-based audio captioning model with keyword estimation," *arXiv preprint arXiv:2007.00222*, 2020.

[11] X. Xu, H. Dinkel, M. Wu, Z. Xie, and K. Yu, "Investigating local and global information for automated audio captioning with transfer learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 905–909.

[12] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] B. Thornton, "Audio recognition using mel spectrograms and convolution neural networks," 2019.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[16] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks*. Citeseer, 2010.

[17] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[20] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[21] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[22] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the second workshop on statistical machine translation*, 2007, pp. 228–231.

[23] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[24] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[25] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *European conference on computer vision*. Springer, 2016, pp. 382–398.

[26] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 873–881.

[27] T. Lidy and A. Schindler, "Cqt-based convolutional neural networks for audio scene classification," in *Proceedings of the detection and classification of acoustic scenes and events 2016 workshop (DCASE2016)*, vol. 90, 201 organization=IEEE Budapest, pp. 1032–1048.

[28] B. Ayoub, K. Jamal, and Z. Arsalane, "Gammatone frequency cepstral coefficients for speaker identification over voip networks," in *2016 International Conference on Information Technology for Organizations Development (IT4OD)*. IEEE, 2016, pp. 1–5.

SELF-TRAINED AUDIO TAGGING AND SOUND EVENT DETECTION IN DOMESTIC ENVIRONMENTS

Janek Ebbers, Reinhold Haeb-Umbach

Paderborn University, Germany
{ebbers, haeb}@nt.upb.de

ABSTRACT

In this paper we present our system for the *Detection and Classification of Acoustic Scenes and Events (DCASE) 2021 Challenge Task 4: Sound Event Detection and Separation in Domestic Environments*, where it scored the fourth rank. Our presented solution is an advancement of our system used in the previous edition of the task.We use a forward-backward convolutional recurrent neural network (FBCRNN) for tagging and pseudo labeling followed by tag-conditioned sound event detection (SED) models which are trained using strong pseudo labels provided by the FBCRNN. Our advancement over our earlier model is threefold. First, we introduce a strong label loss in the objective of the FBCRNN to take advantage of the strongly labeled synthetic data during training. Second, we perform multiple iterations of self-training for both the FBCRNN and tag-conditioned SED models. Third, while we used only tag-conditioned CNNs as our SED model in the previous edition we here explore sophisticated tag-conditioned SED model architectures, namely, bidirectional CRNNs and bidirectional convolutional transformer neural networks (CTNNs), and combine them. With metric and class specific tuning of median filter lengths for post-processing, our final SED model, consisting of 6 submodels (2 of each architecture), achieves on the public evaluation set poly-phonic sound event detection scores (PSDS) of 0.455 for scenario 1 and 0.684 for scenario 2 as well as a collar-based F_1-score of 0.596 outperforming the baselines and our model from the previous edition by far. Source code is publicly available at `https://github.com/fgnt/pb_sed`.

Index Terms— sound event detection, audio tagging, weak labels, self-training

1. INTRODUCTION

Automatic Detection and Classification of Acoustic Scenes and Events (DCASE) has huge potential for various applications such as smart homes, multimedia search and environmental monitoring, to name a few. Due to the high diversity and variability of sounds, however, it is a challenging problem. Driven by the increasing interest from academia and industry and the success of data-driven approaches, the state-of-the-art in DCASE has recently progressed rapidly. The annual DCASE Challenges [1] further push and evaluate the current state-of-the-art in multiple sub-disciplines.

In this contribution we are concerned with the recognition of individual sound events. Here, sound event detection (SED) is the task of recognizing and temporally localizing sound events in an audio clip, whereas audio tagging aims to only recognize their presence within an audio clip without its temporal localization [2].

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 282835863. Computational resources were provided by the Paderborn Center for Parallel Computing.

One particular challenge in SED is that large-scale sound databases, such as Google's Audio Set [3], usually only provide tags a.k.a. weak labels, which only indicate the presence or absence of sound events within an audio clip without the information about the temporal location. Several approaches have been proposed for learning to localize sound events from weakly labeled data [4, 5, 6, 7], most of which use some sort of multiple instance learning (MIL) pooling function [8]. Another topic of interest, not only for sound recognition, is semi-supervised learning, which aims to exploit unlabeled data in addition to labeled data to improve performance. Here, approaches are usually based on representation learning [9], pre-training [10], teacher-student learning [11, 12] or self-training [13]. Self-training initially trains models on the available labeled data followed by iterative pseudo labeling [14] of the unlabeled data and retraining on labeled and pseudo labeled data.

For several years now, the Task 4 of the DCASE Challenge [15, 16, 17] tackles both of above challenges. Recently, it also explores the benefit of strongly labeled synthetic data in addition to weakly labeled and unlabeled real data. For this, the Domestic Environment Sound Event Detection (DESED) data set [16] with 10 different target sound events from a domestic environment has been designed. It is composed of 10-sec audio clips and comprises 1578 weakly labeled and 14412 unlabeled real training clips as well as isolated sound events and backgrounds for synthetic soundscape generation. Further, 1168 and 692 strongly labeled real audio clips are provided for validation and public evaluation, respectively.

In this paper we present our solution for the most recent *DCASE 2021 Challenge Task 4: Sound Event Detection and Separation in Domestic Environments*. Here, we built on our previously proposed forward-backward convolutional recurrent neural network (FBCRNN) and tag-conditioned SED [18], and propose three measures to improve performance. First, we introduce an explicit strong label loss in the FBCRNN training to exploit the strong labels from the synthetic data. Second, we perform more extensive self-training. Third, we explore more sophisticated CRNNs and convolutional transformer neural networks (CTNNs) for tag-conditioned SED in addition to the previously used CNN architecture. We show that all three measures improve performance, allowing us to significantly outperform the baseline and, to the best of our knowledge, set a new state-of-the-art in terms of collar-based F_1-score on the public evaluation set.

The rest of the paper is structured as follows. In Sec. 2 we recap the FBCRNN, introduce the strong label loss and outline the proposed FBCRNN self-training. In Sec. 3 we discuss architectures and self-training for the tag-conditioned SED. Sec. 4 presents implementation details w.r.t. data preparation, training and post processing. Finally, results are presented in Sec. 5 after which we draw conclusions in Sec. 6.

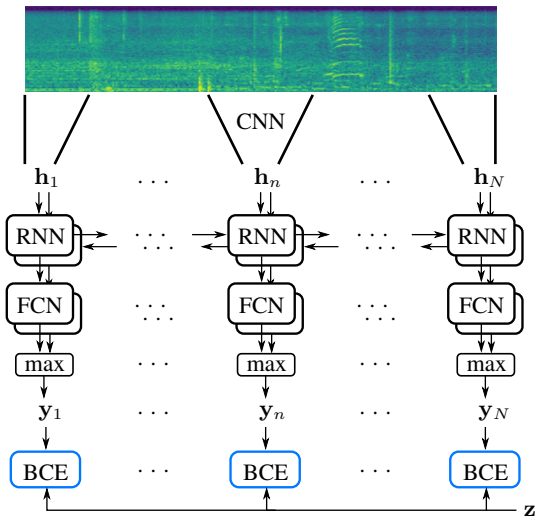


Figure 1: FBCRNN

2. FORWARD-BACKWARD CRNN

The FBCRNN [18] is illustrated in Fig. 1. It consists of a shared CNN front-end and two separate recurrent classifier networks (RNN+fully connected neural network (FCN)) with one processing the audio in forward direction and the other in backward direction. Note that unlike a bidirectional RNN the two classifiers do not exchange hidden representations and, therefore, at each frame one classifier has only seen previous frames and the other only subsequent frames.

To encourage the model to output tag predictions as soon as it has seen the event in the input when training with clip-level (weak) labels, we compute, at each frame, the binary cross entropy (BCE) loss between the point-wise maximum of the predictions of the two classifiers and the weak label. Fig. 2 shows an example, where the weak target and prediction signals are shown purple in the first and fourth subplots, respectively, assuming some decent forward and backward predictions shown in the third subplot. Note, that the FBCRNN training scheme can be seen as MIL with two instances. One instance comprises the current plus all previous frames, which has been processed by the forward classifier, and the other instance comprises the current plus all subsequent frames, which has been processed by the backward classifier. Hence, if an event is labeled positive in the clip at least one of the classifiers has to be able to classify the event as positive, given that the event is either present in previous or in subsequent frames or both.

At test-time a clip-level prediction is obtained by averaging the final forward and the final backward predictions of all models in an ensemble. As the proposed training scheme forces the forward and backward classifiers to output predictions without having processed the whole audio, the FBCRNN generalizes to much shorter segments at test-time. This enables FBCRNN-based SED, where FBCRNNs are applied to small contexts of, say, a couple of 100 ms around each frame to obtain frame-wise SED scores.

We use the same architecture as in [18], where, however, we removed the last pooling layer between the Conv2d and Conv1d blocks.

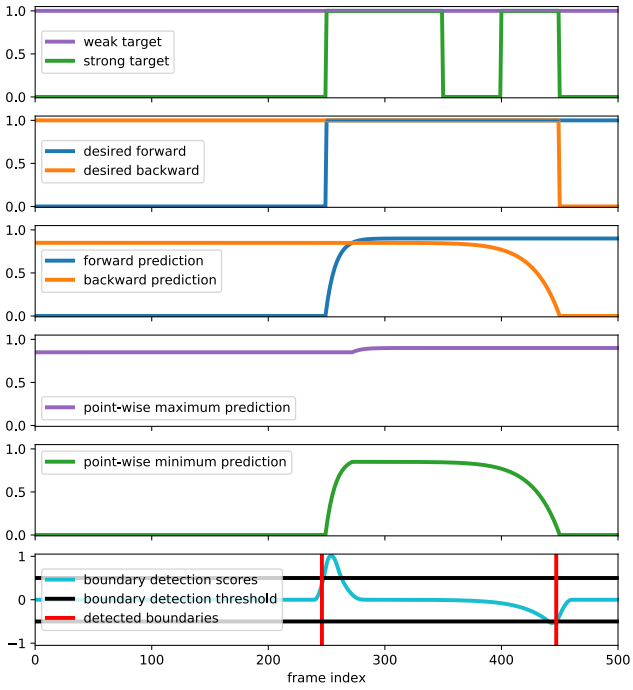


Figure 2: FBCRNN signals

2.1. Strong Label Loss

As the training data of the challenge contains synthetic data which comes with strong labels, it is desirable to make use of the strong labels in the FBCRNN training, which we previously did not do. If strong labels are given, we now, instead of the weak label loss, compute a strong label BCE for both classifiers with respect to the desired outputs, which are illustrated exemplarily in the second subplot of Fig. 2, and average the forward and backward loss terms.

2.2. Self-Training

As a large fraction of the provided data is unlabeled, we now perform more extensive self-training with training 8 initial FBCRNNs on only weakly labeled real and strongly labeled synthetic data followed by three iterations of pseudo labeling and retraining 4 FBCRNN models in each iteration.

In each iteration we generate weak pseudo labels for the complete unlabeled data, where tagging thresholds are tuned on the validation set to maximize the F_1-score. Additionally, we perform a boundary detection for weakly labeled and unlabeled data by filtering the point-wise minimum of the two classifier signals with $\begin{bmatrix}-2/N & \dots & -2/N & 2/N & \dots & 2/N\end{bmatrix}$ where N is the filter size. Exemplary point-wise minimum and subsequent boundary detection are depicted in the two last subplots of Fig. 2. The class-specific filter sizes and thresholds that the output or negative output has to exceed to detect an onset or offset boundary, respectively, are tuned on the validation data such that a minimum collar-based precision of 75 % is achieved, when using collars of 500 ms. For those events where onset and offset can be detected, the strong label loss from Sec. 2.1 is used in the following FBCRNN retraining.

Finally, we use both the FBCRNN ensemble after the second and third iteration to separately perform strong pseudo labeling of the real data (weakly labeled and unlabeled) giving us a set of strong

pseudo labels for each of the ensembles, i.e. two in total. For the FBCRNN-based SED, class specific context lengths, median filter lengths and detection thresholds are tuned on the validation set to maximize the frame-based F_1-score. The obtained strong pseudo labels allow us to train SED systems in a strongly supervised manner as described in the following.

3. TAG-CONDITIONED SED

As in the previous edition, our SED model uses tag-conditioning [18], which means we also input the predicted tags from a FBCRNN (ensemble) in addition to the audio input features. While in the previous edition we only used a tag-conditioned CNN, we now also train a tag-conditioned bidirectional CRNN and tag-conditioned bidirectional CTNN.

Here, we use similar architectures as in the FBCRNN with, however, only one classifier back-end. For the pure CNN the CNN1d and RNN Blocks are removed. In the bidirectional CRNN, a bidirectional RNN is employed instead of unidirectional RNNs as in the FBCRNN. For the CTNN a Transformer Encoder [19] is used instead of an RNN, where we use 3 Transformer blocks each with 10 heads and 32-dimensional embeddings in each head. Also a positional encoding is added at the Transformer input.

Tag-conditioning is performed by concatenating a 10-dimensional multi-hot encoding of the tags with the inputs of the CNN2d, CNN1d, RNN/Tranformer, and FCN Blocks. For the CNN1d, RNN and FCN the encoding is concatenated along feature dimension at each frame. For the CNN2d the encoding is concatenated along channel dimension at each time-frequency bin.

The models are trained with standard strong label BCE loss. For each set of the 2 strong pseudo label sets we train each of the model architectures giving us 3 models for each of the 2 strong pseudo label sets. For each of the strong pseudo label sets, we perform one iteration of self-training, i.e., generating new strong pseudo labels using the 3 models of that particular set followed by retraining the 3 architectures. Finally, we combine all the models from the two sets of pseudo labels into our final ensemble, i.e., 6 models in total.

4. IMPLEMENTATION DETAILS

4.1. Data Preparation/Augmentation

Initially, waveforms are resampled to $16\,\mathrm{kHz}$ and normalized $x(t) = s(t)/\max(|s(t)|)$ to be within the range of -1 and 1. As our system's input we then extract a $M{=}128$-dimensional log-mel spectrogram using a short-time Fourier transform (STFT) with frame-length of $60\,\mathrm{ms}$ and hop-size of $20\,\mathrm{ms}$. Each mel-bin is globally normalized to zero mean and unit variance.

At training time we perform various on-the-fly data augmentations, which is similar to what we already used previously [20, 18] and is described in the following.

Scaling: We randomly scale the waveform with a scale weight sampled out of a Log Truncated Standard Normal distribution with truncation at $\log(3)$.

Shifted superposition: We randomly superpose two audios as $x'_i(t) = x_i(t) + x_j(t-\tau)$ with a random shift τ sampled uniformly such that the superposed signal is not longer than $15\,\mathrm{s}$, i.e., if we, e.g., superpose 2 signals each having a length of $10\,\mathrm{s}$, the shift is uniformly sampled between -5 s and 5 s. Labels are superposed accordingly and clipped at 1 to retain binary labels. We apply superposition with a probability of $2/3$. Due to the similarity to mixup [21], we previously referred to this augmentation also as mixup. However, as we do not interpolate the signals, calling it superposition is more accurate.

Frequency warping: We randomly warp the center frequencies of the mel filter bank similar to vocal tract length perturbation (VTLP) [22]. The boundary frequency is sampled from a Truncated Exponential distribution with $\sigma = M/2$ and truncation at $5 \cdot M$. The warping factor is sampled from a Log Truncated Normal distribution with $\mu = 0$, $\sigma = 0.8$ and truncation at $\log(1.3) \approx 0.26$. Note that the boundary frequency can fall above M, in which case the whole spectrogram is stretched or squeezed and filled with zeros.

Frequency-/Time-Masking: As in SpecAugment [23], we apply one time- and one frequency mask for each input with random locations and widths. The locations are uniformly sampled along the time- and frequency axes, respectively. Widths are uniformly sampled between 0 and $\min(1.4\,\mathrm{s}, 0.2T)$ for the time mask, where T is the length of the audio, or between 0 and 20 bins for the frequency mask.

Gaussian Noise: We add Gaussian noise to the final feature map with its standard deviation being uniformly sampled between 0 and 0.2.

Note, that in contrast to [18], we here neither perform blurring nor reverberation of events in the synthetic data, since it has proven to be not effective.

4.2. Training

Training is performed for $40\,\mathrm{k}$ update steps with a batch size of 16. To balance the different data sets we repeat certain data sets in one epoch multiple times. Here, one epoch consists of 20 times the weakly labeled data, two times pseudo labeled unlabeled data (if used), one time the provided synthetic data from this edition (synthetic21) and two times the provided synthetic data from previous edition (synthetic20). This sums up to $\approx 31\,\mathrm{k}{+}28\,\mathrm{k}{+}10\,\mathrm{k}{+}5\,\mathrm{k}$ audio clips in one epoch. We further ensure that each batch includes at least 6 clips from the weakly labeled data, 2 clips from synthetic21 and 1 clip from synthetic20 as well as at least 1 example of each event class. We employ Adam [24] for optimization with a learning rate of $5 \cdot 10^{-4}$, with a ramp up during the first $1\,\mathrm{k}$ update steps and a reduction to 10^{-4} after $20\,\mathrm{k}$ update steps. We perform validation every $1\,\mathrm{k}$ update steps and choose the checkpoint with best validation performance in terms of (frame-based) F_1-score as the final model.

4.3. Post-Processing

At test-time we use median filtering and a non-linear score transformation for post-processing.

Median filter sizes are tuned for each event class and for each evaluation metric separately to give best performance on the validation set.

The class-specific non-linear score transformation serves the purpose of getting a smooth poly-phonic sound event detection scores (PSDS)-ROC [25] with linearly spaced detection thresholds. It transforms the prediction scores such that in the validation set prediction scores from positively labeled frames are uniformly distributed between 0 and 1. Note that the non-linear score transformation followed by linearly spaced detection thresholds is equivalent to non-linearly spaced detection thresholds.

Table 1: Single model FBCRNN performance on eval-public in %. Bold values indicate best performance in a column. Underlines indicate significant improvements within a block.

Iteration	PSDS1	PSDS2	$F_1^{(\text{collar})}$	$F_1^{(\text{tag})}$
0	31.6±0.6	67.3±1.7	44.1±1.1	83.8±0.8
w/o sll	29.0±2.1	67.2±3.0	41.2±1.9	83.3±0.6
1	36.4±0.5	68.0±1.0	49.1±1.4	84.6±0.3
w/o psll	33.2±0.7	68.9±1.3	47.4±0.6	85.1±0.7
2	**38.2**±0.9	68.9±1.3	**50.9**±1.0	85.1±0.4
3	37.9±1.4	**70.2**±1.2	50.7±1.2	**85.6**±0.6

Table 2: Single model tag-conditioned SED performance on eval-public in %. Bold values indicate best performance in the iteration.

Iteration	Model	PSDS1	PSDS2	$F_1^{(\text{collar})}$
0	CNN	38.2±2.7	64.4±0.4	54.4±0.1
	CRNN	39.7±0.8	**66.7**±0.8	54.5±0.1
	CTNN	**40.9**±1.5	66.2±0.6	**55.7**±0.5
1	CNN	39.6±1.2	64.3±0.6	54.4±0.3
	CRNN	39.8±0.6	**67.0**±1.0	**56.6**±0.1
	CTNN	**40.8**±1.6	66.3±0.4	56.5±0.6

5. RESULTS

We report results on the public evaluation set (eval-public), also referred to as Youtube evaluation [16, 26], as well as official challenge results[1] (eval-2021). Performance is measured in terms of

- PSDS1 / PSDS2: PSDSs [25] with two different sets of parameters as used in the challenge[2]
- $F_1^{(\text{collar})}$: macro-average collar-based F_1-score [27] with a 200 ms collar on onsets and a 200 ms / 20% of the event length collar on offsets
- $F_1^{(\text{tag})}$: macro-average audio tagging F_1-score

For F_1-scores, which evaluate a single operating point, class-specific detection thresholds are tuned to give best performance on the validation set. Note that PSDS1 and $F_1^{(\text{collar})}$ have a focus on accurate temporal localization of sound events, while PSDS2 and $F_1^{(\text{tag})}$ focus on the recognition of active classes.

For FBCRNN-based SED evaluation, context lengths are tuned along with the post-processing hyper parameters for each event class and for each SED evaluation metric separately to give best performance on the validation set. Table 1 presents the single model FBCRNN performance on eval-public over the iterations of the proposed self-training. For reference, we further report in iteration 0 the performance without the strong label loss (sll), as described in Sec. 2.1, and in iteration 1 the performance when not pseudo labeling boundaries in the real data, as described in Sec. 2.2, i.e., without a pseudo strong label loss (psll) on some real data. In each line we report the means and standard deviations over 4 independently trained models.

It can be observed that all metrics improve with the first two iterations of self-training. In the third iteration only PSDS2 and $F_1^{(\text{tag})}$ improve further, whereas PSDS1 and $F_1^{(\text{collar})}$ decrease insignificantly. Further, the proposed strong label loss (sll) and the pseudo strong label loss (psll), in iterations 0 and 1, respectively, allow to significantly improve PSDS1 and $F_1^{(\text{collar})}$ demonstrating their benefit for the temporal localization of sounds.

Table 3: Ensemble results on eval-public and eval-2021 in %. Bold values indicate best performance.

	eval-public			eval-2021		
Model	PSDS1	PSDS2	$F_1^{(\text{collar})}$	PSDS1	PSDS2	$F_1^{(\text{collar})}$
Baseline [12]	35.9	59.6	40.8	31.5	54.7	37.3
Winner [28]	**51.7**	**77.8**	57.4	**45.2**	**74.6**	52.3
FBCRNN	40.6	70.7	52.4	-	-	-
TCSED	45.5	68.4	**59.6**	41.6	63.7	**56.7**

Next, we evaluate the tag-conditioned SED (TCSED) models. Recap from Sec. 3 that we train each of the tag-conditioned architectures (CNN,CRNN,CTNN) on each of the strong pseudo label sets obtained from the FBCRNNs from iterations 2 and 3, followed by one iteration of self-training within each label set separately. In Table 2 we report the means and standard deviations of the results on eval-public over the two label sets.

When comparing performances between iterations 0 and 1, one can see that only for $F_1^{(\text{collar})}$ a significant improvement can be achieved in iteration 1. When comparing results for the different model architectures, it can be observed that tag-conditioned CRNNs and CTNNs perform more or less similar and outperform the tag-conditioned CNNs.

Finally, we report ensemble results in Table 3. Our final FBCRNN ensemble consists of 8 FBCRNNs from after the second and third iterations of the FBCRNN self-training (Table 1). Our TCSED ensemble, which was submitted to the challenge, consists of the 6 models after the single TCSED self-training iteration (Table 2). Both ensembles significantly outperform the challenge baseline w.r.t. all metrics. While the TCSED ensemble significantly outperforms the FBCRNN in PSDS1 and $F_1^{(\text{collar})}$, which both measure temporal localization of sound events, the FBCRNN achieves better results for PSDS2 which primarily measures recognition performance. Here, the FBCRNN-based SED benefits from the tuning of the context lengths, where large contexts are beneficial for PSDS2 evaluation. Compared to the winning system, our system, is outperformed in terms of PSDS1 and PSDS2. However, our TCSED ensemble achieves the highest $F_1^{(\text{collar})}$ of the challenge[2] and, to the best of our knowledge, the highest so far published $F_1^{(\text{collar})}$ on the eval-public set.

6. CONCLUSIONS

In this paper we presented our system for the *DCASE 2021 Challenge Task 4: Sound Event Detection and Separation in Domestic Environments*, where it scored the fourth rank. Starting from FBCRNNs followed by tag-conditioned SEDs, which we proposed in the previous challenge edition, we here presented three measures which significantly improve SED performance. First, we introduced a strong label loss in the FBCRNN training to leverage strong annotations, which is shown to improve temporal sound localization. Then, we performed extensive self-training in both FBCRNN training and tag-conditioned SED training, which particularly improves FBCRNN-based audio tagging and SED performance. Finally, we explored CRNN and CTNN architectures for tag-conditioned SEDs, in addition to CNNs used previously, which gives another performance gain. The proposed measures allow us to set a new, to the best of our knowledge, state-of-the-art in terms of collar-based F_1-score on the public evaluation set of the DESED data set.

[1] http://dcase.community/challenge2021/task-sound-event-detection-and-separation-in-domestic-environments-results

[2] http://dcase.community/challenge2021/task-sound-event-detection-and-separation-in-domestic-environments

7. REFERENCES

[1] "DCASE Challenges," http://dcase.community/events#challenges, accessed: 2021-07-20.

[2] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. Springer, 2018.

[3] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[4] A. Shah, A. Kumar, A. G. Hauptmann, and B. Raj, "A closer look at weak label learning for audio events," *arXiv preprint arXiv:1804.09288*, 2018.

[5] B. McFee, J. Salamon, and J. P. Bello, "Adaptive pooling operators for weakly labeled sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2180–2193, 2018.

[6] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 121–125.

[7] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Weakly-supervised sound event detection with self-attention," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 66–70.

[8] Y. Wang, J. Li, and F. Metze, "A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 31–35.

[9] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 126–130.

[10] E. Fonseca, D. Ortego, K. McGuinness, N. E. OConnor, and X. Serra, "Unsupervised contrastive learning of sound event representations," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 371–375.

[11] L. JiaKai, "Mean teacher convolution system for dcase 2018 task 4," DCASE2018 Challenge, Tech. Rep., September 2018.

[12] N. Turpault and R. Serizel, "Training sound event detection on a heterogeneous dataset," in *DCASE Workshop*, 2020.

[13] B. Elizalde, A. Shah, S. Dalmia, M. H. Lee, R. Badlani, A. Kumar, B. Raj, and I. Lane, "An approach for self-training audio event detectors using web data," in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 1863–1867.

[14] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2, 2013.

[15] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. P. Shah, "Large-scale weakly labeled semi-supervised sound event detection in domestic environments," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 19–23.

[16] N. Turpault, R. Serizel, J. Salamon, and A. P. Shah, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, October 2019, pp. 253–257.

[17] N. Turpault, R. Serizel, S. Wisdom, H. Erdogan, J. R. Hershey, E. Fonseca, P. Seetharaman, and J. Salamon, "Sound event detection and separation: a benchmark on desed synthetic soundscapes," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 840–844.

[18] J. Ebbers and R. Haeb-Umbach, "Forward-backward convolutional recurrent neural networks and tag-conditioned convolutional neural networks for weakly labeled semi-supervised sound event detection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 41–45.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[20] J. Ebbers and R. Hb-Umbach, "Convolutional recurrent neural network and data augmentation for audio tagging with noisy labels and minimal supervision," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, October 2019, pp. 64–68.

[21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[22] N. Jaitly and G. E. Hinton, "Vocal tract length perturbation (vtlp) improves speech recognition," in *Proceedings of ICML Workshop on Deep Learning for Audio, Speech and Language*, vol. 117, 2013.

[23] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] Ç. Bilen, G. Ferroni, F. Tuveri, J. Azcarreta, and S. Krstulović, "A framework for the robust evaluation of sound event detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 61–65.

[26] N. Turpault, R. Serizel, A. Shah, and J. Salamon, "Desed_public_eval," Dec. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3588172

[27] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[28] X. Zheng, H. Chen, and Y. Song, "Zheng ustc teams submission for dcase2021 task4 semi-supervised sound event detection," DCASE2021 Challenge, Tech. Rep., June 2021.

IMPROVING SOUND EVENT DETECTION WITH AUXILIARY FOREGROUND-BACKGROUND CLASSIFICATION AND DOMAIN ADAPTATION

Michel Olvera[1], *Emmanuel Vincent*[1], *Gilles Gasso*[2]

[1] Université de Lorraine, CNRS, Inria, Loria, F-54000 Nancy, France
[2] LITIS EA 4108, Université & INSA Rouen Normandie, 76800 Saint-Étienne du Rouvray, France
michel.olvera@inria.fr

ABSTRACT

In this paper we provide two methods that improve the detection of sound events in domestic environments. First, motivated by the broad categorization of domestic sounds as foreground or background events according to their spectro-temporal structure, we propose to learn a foreground-background classifier jointly with the sound event classifier in a multi-task fashion to improve the generalization of the latter. Second, while the semi-supervised learning capability adopted for training sound event detection systems with synthetic labeled data and unlabeled or partially labeled real data aims to learn invariant representations for both domains, there is still a gap in performance when testing such systems on real environments. To further reduce this data mismatch, we propose a domain adaptation strategy that aligns the empirical distributions of the feature representations of active and inactive frames of synthetic and real recordings via optimal transport. We show that these two approaches lead to enhanced detection performance in terms of the event-based macro F1-score on the DESED dataset.

Index Terms— Sound event detection, foreground-background classification, semi-supervised learning, domain adaptation

1. INTRODUCTION

Over the past five years, the interest in environmental acoustic scenes has increased considerably among researchers in the field of audio signal processing, largely driven by the Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge and Workshop series [1–3]. As a result, many tasks involving the automatic analysis of ambient sounds have progressed substantially. This includes sound event detection (SED), a core task for home surveillance or assisted living [4–6].

To this end, DCASE Challenge Task 4 encourages the development of methods that contribute to the advancement of SED methods that are trained in a semi-supervised way with a heterogeneous dataset [7] including a set of synthetic soundscapes with annotations indicating the sound events class labels and timestamps (strong labels), as well as a set of real recordings, mostly unlabeled and where only a small subset contains at most information about the active sound classes in the recordings (weak labels). The objective of the task is to find for a given soundscape the class of the active sounds as well as their onset and offset time. Many improvements to the Task 4 baseline system have been proposed: augmentation schemes to improve generalization [8, 9]; changes in the acoustic front-end with alternative time-frequency representations to log-Mel spectrograms [10] or time-frequency resolutions for each sound event class [11]; modifications to the backbone architecture with appended multi-task branches and post-processing techniques to refine the outputs [12–14].

In this work we propose two methods to improve SED in domestic environments. First, we propose the classification of sounds by their spectro-temporal content as foreground or background events as an auxiliary task for SED. This broad categorization of sound events was shown to be useful for deep neural network-based source separation systems to differentiate rapidly varying spectro-temporal features of short duration sound events from slowly varying features of long duration sounds [15, 16]. The proposed foreground-background classifier is jointly trained with the SED branch in a multi-task fashion and the combination of both branches is also investigated. The second improvement is a domain adaptation strategy to reduce the mismatch between synthetic and real recordings. Our proposed strategy aligns the empirical distributions of the feature representations of active and inactive frames of synthetic and real data via optimal transport [17, 18]. Altogether the proposed methods lead to enhanced performance in terms of the event-based macro F1-score on the Domestic Environment Sound Event Detection Dataset (DESED) validation and public evaluation sets [19, 20].

The remainder of this article is organized as follows. In Section 2 we present the proposed improvements to the SED task. Experimental evaluation is discussed in Section 3 and lastly, we conclude the paper in Section 4.

2. PROPOSED METHODS

2.1. Foreground-background classification

Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be the input, output and latent space. For the SED task we denote the soundscape time-frequency representation by $x \in \mathcal{X}$ with corresponding annotations $y \in \mathcal{Y}$. We have access to a synthetic dataset with strong labels $\mathcal{D}^{\mathcal{S}} = \{(x_i^{\mathrm{s}}, y_i^{\mathrm{s}})\}_{i=1}^{n_{\mathrm{s}}}$ and two datasets of real recordings: a weakly labeled dataset $\mathcal{D}^{\mathcal{W}} = \{(x_i^{\mathrm{w}}, y_i^{\mathrm{w}})\}_{i=1}^{n_{\mathrm{w}}}$ and an unlabeled dataset $\mathcal{D}^{\mathcal{U}} = \{x_i^{\mathrm{u}}\}_{i=1}^{n_{\mathrm{u}}}$.

The SED model is a Mean Teacher model [21] in which both the student and the teacher models have the same convolutional-recurrent neural network (CRNN) architecture. We use the CRNN from the student model as a representation mapping $g : \mathcal{X} \to \mathcal{Z}$, where the log-Mel spectrograms are mapped to the latent space. The

This work was made with the support of the French National Research Agency, in the framework of the project LEAUDS "Learning to understand audio scenes" (ANR-18-CE23-0020). Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (https://www.grid5000.fr).

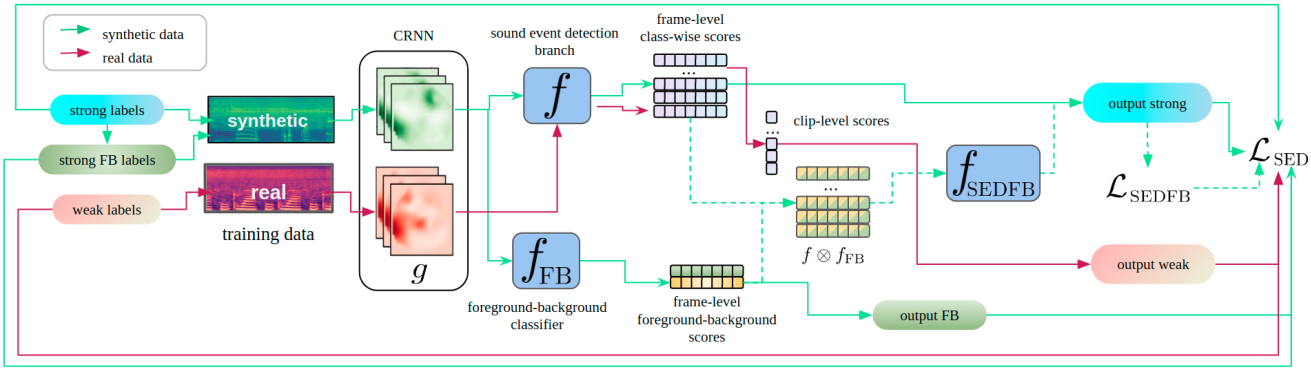


Figure 1: Proposed model with colored data flow for synthetic and real data. For simplicity, the diagram depicts only the student model and the associated classification costs. The dash-lined path represents the training scheme with the fusion of the sound event detection and foreground-background classification branches.

SED model is represented by the function $f : \mathcal{Z} \rightarrow \mathcal{Y}$ that maps the latent representations to the output space.

2.1.1. FB branch

Motivated by the broad categorization of sound events into foreground and background according to their spectro-temporal structure, we propose a foreground-background (FB) auxiliary classifier $f_{\text{FB}} : \mathcal{Z} \rightarrow \mathcal{Y}^{\text{FB}}$ that maps the latent space to foreground-background labels. We learn this classifier jointly with the SED model in a multi-task fashion, hypothesizing that these different yet related classification tasks will help improve the network's generalization capability. Analogously, for the teacher model we denote by g', f' and f'_{FB}, the CRNN embedding function, SED and FB branches, respectively. Figure 1 shows the proposed system depicting the student model.

To train the FB classifier in the multi-task paradigm, we derived foreground-background ground-truth annotations y_i^{fb} from the strong labels y_i^{s} of the synthetic data by combining the sound event labels in two categories: foreground: (*alarm - bell ringing, speech, cat, dog*, *dishes*) and background (*blender, vacuum cleaner, frying, electric shaver - toothbrush*, *running water*). The SED model is optimized by minimizing

$$\begin{aligned}\mathcal{L}_{\text{SED}} = L(y_i^{\text{s}}, f(g(x_i^{\text{s}}))) + \lambda L_{\text{strong}}(f(g(x_i)), f'(g'(x_i)))+ \\ L(y_i^{\text{w}}, f(g(x_i^{\text{w}}))) + \lambda L_{\text{weak}}(f(g(x_i)), f'(g'(x_i)))+ \\ L(y_i^{\text{fb}}, f_{\text{FB}}(g(x_i^{\text{s}}))) + \lambda L_{\text{strong}}(f_{\text{FB}}(g(x_i)), f'_{\text{FB}}(g'(x_i)))\end{aligned} \quad (1)$$

where $L(\cdot,\cdot)$ is a binary cross-entropy classification loss, and $L_{\text{strong}}(\cdot,\cdot)$ and $L_{\text{weak}}(\cdot,\cdot)$ are mean-square error consistency costs which are differentiable on their second parameter over strong (frame-level) and weak (clip-level) scores, respectively. The consistency weight λ is tied to all consistency costs.

2.1.2. SEDFB branch

Going beyond the proposed FB classification branch, we explored its fusion with the SED branch into a detection branch (SEDFB) to refine outputs. This branch is represented by a function $f_{\text{SEDFB}} : \mathcal{Y} \times \mathcal{Y}^{\text{FB}} \rightarrow \mathcal{Y}$ (f'_{SEDFB} for the teacher model). The input for the SEDFB branch is the outer product of the outputs from the SED and FB branches $w_i = f(g(x_i)) \otimes f_{\text{FB}}(g(x_i))$, as this fusion creates a representation containing information from the joint interaction of the SED and FB classifiers. The following classification-consistency cost pair is added to the training objective in (1):

$$\mathcal{L}_{\text{SEDFB}} = L(y_i^{\text{s}}, f_{\text{SEDFB}}(w_i^{\text{s}})) + \lambda L_{\text{strong}}(f_{\text{SEDFB}}(w_i), f'_{\text{SEDFB}}(w_i)). \quad (2)$$

The overall cost involving the SEDFB branch is given by

$$\mathcal{L} = \mathcal{L}_{\text{SED}} + \mathcal{L}_{\text{SEDFB}}. \quad (3)$$

2.1.3. Output smoothing

We used two methods to post-process the SED frame-level scores. The first method corresponds to smoothing the binary multi-label frame-level scores with a median filter of 0.45 s. The second approach consists of Hidden Markov Model (HMM) decoding. Following the same procedure as in [14], we determined the optimal transition probabilities for each sound event class using the validation set. We contrast the contribution of both post-processing schemes in Section 3.3.

2.2. Domain adaptation for sound event detection

From the unsupervised domain adaptation perspective, we regard the synthetic dataset with strong labels as the source domain $\mathcal{S} = \mathcal{D}^{\mathcal{S}}$, and the combination of real recordings from the weakly and unlabeled dataset as the target domain $\mathcal{T} = \mathcal{D}^{\mathcal{W}} \cup \mathcal{D}^{\mathcal{U}}$. We denote as x^{s} and x^{t} the soundscapes from $\mathcal{S}$ and $\mathcal{T}$, respectively.

In contrast to adversarial adaptation approaches that introduce a domain discriminator to reduce the distribution discrepancy between domains [14, 22, 23], our proposed strategy relies on optimal transport for its ability to find correspondences between samples by exploiting the geometry of the underlying space. We adopt the DeepJDOT framework [18], to correct the mismatch between the distributions of learned feature representations in the two domains.

2.2.1. Joint distribution optimal transport

Let $\mu_{\text{s}} = \sum_{i=1}^{n_{\text{s}}} a_i \delta_{g(x_i^{\text{s}}), y_i^{\text{s}}}$ and $\mu_{\text{t}} = \sum_{i=1}^{n_{\text{t}}} b_i \delta_{g(x_i^{\text{t}}), y_i^{\text{t}}}$ be two empirical distributions on the product space $\mathcal{Z} \times \mathcal{Y}$, where $\delta_{g(x_i), y_i}$

is the Dirac function at position $(g(x_i), y_i) \in \mathcal{Z} \times \mathcal{Y}$, and a_i and b_i are uniform probability weights, i.e. $\sum_{i=1}^{n_s} a_i = \sum_{i=1}^{n_t} b_i = 1$. The associated cost for the i-th source and j-th target element can be expressed as a weighted combination of costs in the latent and label spaces

$$d(g(x_i^s), y_i^s; g(x_j^t), y_j^t) = \alpha c(g(x_i^s), g(x_j^t)) + \beta \mathcal{L}(y_i^s, y_j^t) \quad (4)$$

where $c(\cdot, \cdot)$ is the squared ℓ_2 distance, $\mathcal{L}(\cdot, \cdot)$ is a cross-entropy loss that enforces regularity between the source and target domain labels, and α and β are two scalar values. Since no labels y_j^t are available in the target domain they are replaced with pseudo-labels $f(g(x_j^t))$ obtained from the classifier $f : \mathcal{Z} \to \mathcal{Y}$. We seek for a transportation coupling $\gamma \in \mathbb{R}^{n_s \times n_t}$ in the space $\Gamma(\mu_s, \mu_t)$ of joint probability distributions with marginals $\gamma \mathbf{1}_{n_t} = \mu_s$ and $\gamma^T \mathbf{1}_{n_s} = \mu_t$, where $\mathbf{1}_d$ is a d-dimensional vector of ones, and a pair of mapping functions g and f that minimize

$$\min_{\gamma \in \Gamma(\mu_s, \mu_t), g, f} \sum_{i,j} \gamma_{i,j} d(g(x_i^s), y_i^s; g(x_j^t), f(g(x_j^t))). \quad (5)$$

We follow a two-step procedure to solve this optimization problem. In the first step, we compute the optimal coupling matrix γ with fixed model parameters f and g,

$$\min_{\gamma \in \Gamma(\mu_s, \mu_t)} \sum_{i,j} \gamma_{ij} (\alpha ||g(x_i^s) - g(x_j^t)||^2 + \beta \mathcal{L}(y_i^s, f(g(x_j^t)))). \quad (6)$$

In the second step, with fixed γ, we update the models g and f as

$$\min_{g,f} \mathcal{L}_s + \sum_{i,j} \gamma_{ij} (\alpha ||g(x_i^s) - g(x_j^t)||^2 + \beta \mathcal{L}(y_i^s, f(g(x_j^t)))) \quad (7)$$

where $\mathcal{L}_s$ correspond to the classification cost on the source domain to avoid losing performance on synthetic data.

2.2.2. *Sampling strategy*

For each data batch we sample all active and inactive frames from the source and target domains as indicated by the strong labels and the pseudo-labels. For both domains we only keep active frames where no sound event overlap occurs, so that the optimal transport takes place between the empirical distributions of the sound classes of both domains. The number of sampled active frames per class can vary considerably from batch to batch for synthetic and real data, which can lead to the absence of certain classes in one type of data for some batches. To account for this imbalance problem we only keep active frames from common classes to both domains, and then balance all classes by resampling them randomly, taking as many elements as there are in the class with fewer elements. Similarly, the number of inactive frames in the source and target domains varies from batch to batch, so we sample randomly each set by taking as many inactive frames as there are in the set with fewer elements.

2.2.3. *Pseudo-label refinement*

To improve the reliability of the pseudo-labels assigned to real data, we leverage the provided annotations of the weakly labeled set to refine pseudo-labels on this subset. The refinement process consists of fusing the frame-level outputs of the SED branch f on soundscapes from $\mathcal{D}^{\mathcal{W}}$ with their clip-level annotations by an element-wise multiplication. The target domain pseudo-labels are thus updated for

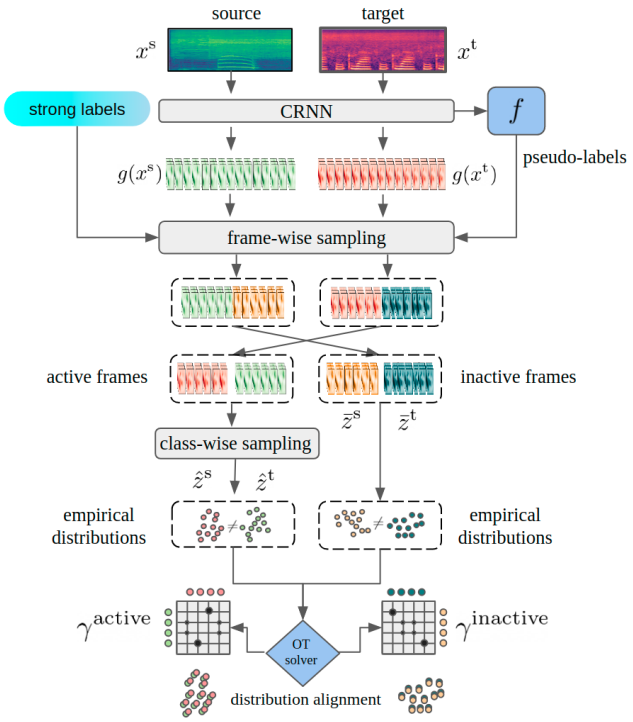


Figure 2: Proposed adaptation method to correct domain mismatch.

weakly labeled data as $\hat{y}_j^t = f(g(x_j^w)) \odot y_j^w$, $j = 1, \ldots, n_w$. This operation constrains the estimated labels to contain at most the same classes present in the weakly labeled soundscapes. Filtering out all extra classes helps reduce false positives and allows more reliable pseudo-labels to be obtained for the proposed sampling strategy and domain adaptation process.

2.2.4. *Training objectives*

We denote as $\hat{z}^s$ and $\hat{z}^t$ the sampled active frames and as $\bar{z}^s$ and $\bar{z}^t$ the sampled inactive frames from the source and target domain latent representations z^s and z^t, respectively. After an initial pre-training stage using (3), we construct the following objective function to account for the mismatch between the empirical distributions of active and inactive learned feature representations

$$\mathcal{L}_s + \mathcal{L}_{active} + \mathcal{L}_{inactive} \quad (8)$$

where

$$\mathcal{L}_s = \frac{1}{n_s} \sum_{i=1}^{n_s} L(y_i^s, f(g(x_i^s))) + \frac{1}{n_s} \sum_{i=1}^{n_s} L(y_i^{fb}, f_{FB}(g(x_i^s))) \quad (9)$$

corresponds to the first and third classification cost terms of the training classification cost in (1). As only the student model undergoes adaptation, no consistency losses are included in the above objective to train the source domain classifier. The cost function $\mathcal{L}_a$ corresponds to the distribution alignment loss of active frames

$$\mathcal{L}_{active} = \frac{1}{|C_{active}|} \sum_{i,j}^{N_{active}} \gamma_{ij}^{active} (\alpha ||\hat{z}_i^s - \hat{z}_i^t||^2 + \beta \mathcal{L}(y_i^s, \hat{y}_j^t)) \quad (10)$$

where $|C_{\text{active}}|$ is the cardinality of the subset of labels $C_{\text{active}} \in C$ representing the total number of active classes in the batch. The second term in (10) enforces consistency between the target domain pseudo-labels and source domain labels. The cost function $\mathcal{L}_{\text{inactive}}$ accounts for the alignment of the marginal distributions of the learned representations of inactive frames in both domains:

$$\mathcal{L}_{\text{inactive}} = \sum_{i=1}^{N_{\text{inactive}}} \gamma_{ij}^{\text{inactive}} (\alpha ||\bar{z}_i^{\text{s}} - \bar{z}_i^{\text{t}}||^2). \quad (11)$$

Figure 2 depicts the proposed frame-level domain adaptation strategy based on optimal transport for the SED task.

3. EXPERIMENTS AND RESULTS

3.1. Model

The selected model architecture is the same as the baseline system of DCASE 2020. The CNN part is composed of 7 layers with 16, 32, 64, 128, 28, 128, 128 filters, respectively. A kernel of size 3x3 was used with max-pooling [2, 2], [2, 2], [1, 2], [1, 2], [1, 2], [1, 2] and [1, 2], respectively. A gated linear unit activation is applied to the convolution operations. The RNN part is composed of 2 layers of 128 bidirectional gated recurrent units. The output of the CRNN is followed by a dense layer with sigmoid activation to produce frame-level (strong) class-wise posteriors. Clip-level (weak) scores are obtained by multiplying the aforementioned linear layer with a dense layer with softmax activation followed by mean temporal aggregation. The FB branch consists of a dense layer with sigmoid activation, which acts upon the outputs of the RNN block. The SEDFB branch is composed of a bidirectional RNN with 128 gated recurrent units and a dense layer with sigmoid activation.

3.2. Dataset and training procedure

We conducted experiments on the Domestic Environment Sound Event Detection Dataset (DESED) dataset [19, 20], composed of a training set of 2,584 synthetic audio clips generated by Scaper [24], 1,578 real soundscapes with clip-level annotations and 14,412 unlabeled real recordings. In the training stage, the model was trained for 200 epochs with the Adam optimizer, a dropout value of 0.5, and a gradually increasing learning rate with a max value of 10^{-3} [25]. The consistency weight λ was set to 1. In the adaptation stage, the student model was adapted for 300 epochs. We used cost weights $\alpha = 0.2$, $\beta = 5.0$, and the contribution of the source domain classifier cost $\mathcal{L}_s$ to the total adaptation cost was multiplied by 100. The learning rate was fixed to 10^{-4}. Experiments with optimal transport were performed using the Python Optimal Transport package [26].

3.3. Results

In Table 1 we compare results obtained by the proposed methods on the validation and public evaluation sets of the DESED dataset in terms of the event-based macro F1 score. The model labeled as Baseline correspond to the baseline system of DCASE 2020 Challenge Task 4. Although the baseline system of the 2021 edition comprises the same architecture as Baseline, it cannot be compared with our methods as it was trained on a different synthetic dataset with data augmentation. For each evaluation set we show performance with median filtering (+MF) or HMM smoothing (+HMM) post-processing.

Adding the FB branch is beneficial to the SED task as it improved results on the validation set compared to Baseline by 8.3%.

Table 1: Performance on the validation and public evaluation sets.

Method	F1 score		F1 score	
	val +MF	val +HMMs	eval +MF	eval +HMMs
Baseline	34.8		38.1	
+ DA	42.41	43.89	44.8	47.12
+ FB	43.12	45.42	46.06	49.38
+ FB + DA	45.68	47.77	50.79	53.10
+ SEDFB	46.15	46.20	48.40	49.79
+ SEDFB + DA	**47.61**	47.75	**52.12**	53.30
DCASE 1	45.13	**48.07**	50.58	**53.35**
DCASE 2	45.15	47.08	50.28	52.23

Further enhancement was achieved by refining outputs with HMM smoothing, as performance increased by 10.6%. Moreover, its fusion with the SED branch into a combined SEDFB branch brought an additional gain of 3% with median filtering. A similar trend is observed for the public evaluation set.

Model adaptation with the proposed strategy increased performance over Baseline by 7.6% as a standalone method, and by 10.8% and 12.8% when combined with the FB and SEDFB branches. These results prove the effectiveness of the system in reducing mismatch between synthetic and real data. Compared to the validation set, performance was about 2% larger on the public set, which might be due to the fact that the empirical distribution of the active and inactive frames of this set resembles more that of the provided real training data on which adaptation was carried out.

HMM smoothing as a post-processing method yielded greater improvement to the scores over median filtering for all proposed models except for the SEDFB-based methods, in which +MF and +HMM provided similar scores, implying that the SEDFB branch plays a similar role as HMM decoding in the modeling of time-varying spectra of sound events.

DCASE 1 and DCASE 2 are model ensembles comprising three and two + FB + DA systems from different training runs, respectively. Ensembling is achieved by simply averaging the model outputs. These models correspond to the submissions made to the DCASE 2021 Challenge Task 4 and are labeled as *Olvera_INRIA_task4_SED_1* and *Olvera_INRIA_task4_SED_2*, respectively. Both systems showed competitive performance in terms of the event-based macro F1-score on the evaluation and public evaluation sets among 65 systems.

4. CONCLUSION

In this paper we proposed two methods that enhance the detection of domestic sound events. Motivated by the categorization of the spectro-temporal characteristics of domestic sounds as foreground or background, we proposed the use of an auxiliary foreground-background classifier that is jointly trained with the sound event classifier to improve generalization. Furthermore, we proposed to incorporate an adaptation stage based on the joint distribution optimal transport of feature embeddings and labels to account for the acoustic mismatch between the available synthetic and real training data. We showed that the multi-task training scheme together with the adaptation stage substantially improved the performance of the baseline system.

5. REFERENCES

[1] M. D. Plumbley, C. Kroos, J. P. Bello, G. Richard, D. P. Ellis, and A. Mesaros, Eds., *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*. Tampere University of Technology. Laboratory of Signal Processing, 2018.

[2] M. Mandel, J. Salamon, and D. P. W. Ellis, Eds., *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. New York University, 2019.

[3] N. Ono, N. Harada, Y. Kawaguchi, A. Mesaros, K. Imoto, Y. Koizumi, and T. Komatsu, Eds., *Proceedings of the Fifth Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2020)*, 2020.

[4] P. van Hengel and J. Anemüller, "Audio event detection for in-home care," in *Int. Conf. on Acoustics (NAG/DAGA)*, 2009.

[5] R. M. Alsina-Pagès, J. Navarro, F. Alías, and M. Hervás, "homesound: Real-time audio event detection based on high performance computing for behaviour and surveillance remote monitoring," *Sensors*, vol. 17, no. 4, p. 854, 2017.

[6] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring activities of daily living in smart homes: Understanding human behavior," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 81–94, 2016.

[7] N. Turpault and R. Serizel, "Training sound event detection on a heterogeneous dataset," in *Proc. DCASE*, 2020, pp. 200–204.

[8] X. Li, "Semi-supervised sound event detection using random augmentation and consistency regularization," *arXiv preprint arXiv:2102.00154*, 2021.

[9] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Conformer-based sound event detection with semi-supervised learning and data augmentation," in *Proc. DCASE*, 2020, pp. 100–104.

[10] A. Copiaco, C. Ritz, N. Abdulaziz, and S. Fasciani, "A study of features and deep neural network architectures and hyperparameters for domestic audio classification," *Applied Sciences*, vol. 11, no. 11, p. 4880, 2021.

[11] D. de Benito-Gorron, D. Ramos, and D. T. Toledano, "A multi-resolution approach to sound event detection in dcase 2020 task4," in *Proc. DCASE*, 2020, pp. 36–40.

[12] Y. Huang, L. Lin, S. Ma, X. Wang, H. Liu, Y. Qian, M. Liu, and K. Ouchi, "Guided multi-branch learning systems for sound event detection with sound separation," in *Proc. DCASE*, 2020, pp. 61–65.

[13] L. Cances, T. Pellegrini, and P. Guyot, "Multi task learning and post processing optimization for sound event detection," IRIT, Université de Toulouse, CNRS, Toulouse, France, Tech. Rep., 2019.

[14] S. Cornell, M. Olvera, M. Pariente, G. Pepe, E. Principi, L. Gabrielli, and S. Squartini, "Domain-adversarial training and trainable parallel front-end for the dcase 2020 task 4 sound event detection challenge," in *Proc. DCASE*, 2020, pp. 26–30.

[15] M. Olvera, E. Vincent, R. Serizel, and G. Gasso, "Foreground-background ambient sound scene separation," in *Proc. EUSIPCO*, 2020, pp. 281–285.

[16] D. D. Varma, R. Padmanabhan, and A. Dileep, "Learning to separate: Soundscape classification using foreground and background," in *Proc. EUSIPCO*, 2020, pp. 21–25.

[17] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *Proc. NIPS*, 2017, pp. 3733–3742.

[18] B. B. Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty, "Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation," in *Proc. ECCV*, vol. 11208, 2018, pp. 467–483.

[19] N. Turpault, R. Serizel, J. Salamon, and A. P. Shah, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Proc. DCASE*, 2019, pp. 253–257.

[20] R. Serizel, N. Turpault, A. Shah, and J. Salamon, "Sound event detection in synthetic domestic environments," in *Proc. ICASSP*, 2020, pp. 86–90.

[21] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *arXiv preprint arXiv:1703.01780*, 2017.

[22] L. Yang, J. Hao, Z. Hou, and W. Peng, "Two-stage domain adaptation for sound event detection," in *Proc. DCASE*, 2020, pp. 230–234.

[23] H. Park, S. Yun, J. Eum, J. Cho, and K. Hwang, "Weakly labeled sound event detection using tri-training and adversarial learning," in *Proc. DCASE*, 2019, pp. 184–188.

[24] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello, "Scaper: A library for soundscape synthesis and augmentation," in *Proc. WASPAA*, 2017, pp. 344–348.

[25] L. Delphin-Poulat and C. Plapous, "Mean teacher with data augmentation for dcase 2019 task 4," Orange Labs Lannion, France, Tech. Rep., 2019.

[26] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer, "Pot: Python optimal transport," *Journal of Machine Learning Research*, vol. 22, no. 78, pp. 1–8, 2021.