European Commission | Horizon 2020 European Union funding for Research & Innovation

Big Data technologies and extreme-scale analytics

MARVEL

Multimodal Extreme Scale Data Analytics for Smart Cities Environments

D3.1: Multimodal and privacy-aware audio-visual intelligence – initial version [†]

Abstract: This document describes the initial version of the methodologies proposed by MARVEL partners towards the realisation of the Audio, Visual and Multimodal AI Subsystem of the MARVEL architecture. These include methods for Sound Event Detection, Sound Event Localisation and Detection, Automated Audio Captioning, Visual Anomaly Detection, Visual Crowd Counting, Audio-Visual Crowd Counting, as well as methodologies for improving the training and efficiency of AI models under supervised, unsupervised, and cross-modal contrastive learning settings. The effectiveness of these methods is compared against recent baselines, towards achieving the AI methodology-related objectives of the MARVEL project.

Contractual Date of Delivery	*30/06/2022*
Actual Date of Delivery	*30/06/2022*
Deliverable Security Class	*Public*
Editor	*Alexandros Iosifidis (AU)*
Contributors	*AU, TAU, FBK, UNS*
Quality Assurance	*Toni Heittola (TAU)* *Nikola Simić (UNS)*

[†] *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337.*

The *MARVEL* Consortium

Part. No.	Participant organisation name	Participant Short Name	Role	Country
1	FOUNDATION FOR RESEARCH AND TECHNOLOGY HELLAS	FORTH	Coordinator	EL
2	INFINEON TECHNOLOGIES AG	IFAG	Principal Contractor	DE
3	AARHUS UNIVERSITET	AU	Principal Contractor	DK
4	ATOS SPAIN SA	ATOS	Principal Contractor	ES
5	CONSIGLIO NAZIONALE DELLE RICERCHE	CNR	Principal Contractor	IT
6	INTRASOFT INTERNATIONAL S.A.	INTRA	Principal Contractor	LU
7	FONDAZIONE BRUNO KESSLER	FBK	Principal Contractor	IT
8	AUDEERING GMBH	AUD	Principal Contractor	DE
9	TAMPERE UNIVERSITY	TAU	Principal Contractor	FI
10	PRIVANOVA SAS	PN	Principal Contractor	FR
11	SPHYNX TECHNOLOGY SOLUTIONS AG	STS	Principal Contractor	CH
12	COMUNE DI TRENTO	MT	Principal Contractor	IT
13	UNIVERZITET U NOVOM SADU FAKULTET TEHNICKIH NAUKA	UNS	Principal Contractor	RS
14	INFORMATION TECHNOLOGY FOR MARKET LEADERSHIP	ITML	Principal Contractor	EL
15	GREENROADS LIMITED	GRN	Principal Contractor	MT
16	ZELUS IKE	ZELUS	Principal Contractor	EL
17	INSTYTUT CHEMII BIOORGANICZNEJ POLSKIEJ AKADEMII NAUK	PSNC	Principal Contractor	PL

Document Revisions & Quality Assurance

Internal Reviewers

1. Toni Heittola, TAU
2. Nikola Simić, UNS

Revisions

Version	Date	By	Overview
2.0	28/06/2022	*Alexandros Iosifidis*	Final version including comments from the Project Coordinator
1.3	06/06/2022	*Alexandros Iosifidis*	2nd Reviewer (UNS) comments addressed
1.2	03/06/2022	*Alexandros Iosifidis*	1st Reviewer (TAU) comments addressed
1.1	24/05/2022	*Alexandros Iosifidis*	2nd round contributions compiled First version for review
1.0	19/05/2022	*Alexandros Iosifidis*	All partner contributions compiled Ask for additional input
0.3	20/04/2022	*Alexandros Iosifidis*	Release of ToC to partners
0.2	20/04/2022	*Dragana Bajovic*	Comments on the ToC
0.1	18/04/2022	*Alexandros Iosifidis*	Table of Contents (ToC)

Disclaimer

Contents

List of Tables

List of Figures

List of Abbreviations

1DConv	One-dimensional convolution
AAC	Automated Audio Captioning
AAVC	Automated Audio-Visual Captioning
ACCDOA	Activity-Coupled Cartesian Direction of Arrival
AE	Autoencoder
AI	Artificial Intelligence
AUC	Area Under Curve
AudioAnony	Audio Anonymisation component
AUROC	Area Under the Receiver Operating Characteristic
AVCC	Audio-Visual Anomaly Detection
BNN	Binary Neural Network
CCG	Cross-Corpus Generalisation
CPU	Central Processing Unit
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
DCASE	Detection and Classification of Acoustic Scenes and Events
DL	Deep Learning
DOA	Direction of Arrival
E2F2C	Edge to Fog to Cloud
EC	European Commission
ER	Error Rate
EU	European Union
FLOPs	Floating Point Operators
FLOPS	Floating Point Operators per Second
FP	False Positive
FN	False Negative
GA	Grant Agreement
GPU	Graphics Processing Unit
HPC	High-Performance Computing
KD	Knowledge Distillation
KPIs	Key Performance Indicators
KWS	Keyword spotting
MAC	Multiply-Accumulate
MCU	Microcontroller Unit
MEMS	Microelectromechanical
MFCC	Mel-Frequency Cepstral Coefficients
ML	Machine Learning
MVP	Minimum Viable Product
RNN	Recurrent Neural Network
SA	Self-Attention
SDK	Software Development Kit
SER	Speech Emotion Recognition
SED	Sound Event Detection
SELD	Sound Event Localisation and Detection
SL-ViT	Single-Layer Vision Transformer
SSL	Sound Source Localisation
SGD	Stochastic Gradient Descent

SVM	Support Vector Machine
TF-IDF	Term-Frequency Inverse-Document-Frequency
TP	True Positive
TN	True Negative
VAD	Voice Activity Detection
VCC	Visual Crowd Counting
ViAD	Visual Anomaly Detection
VideoAnony	Video Anonymisation component
WP	Work Package

Executive summary

This document provides a description of methodologies for multimodal and privacy-aware audio-visual intelligence in the MARVEL project. This is the initial version of the document reporting work conducted in T3.3 *Multimodal audio-visual intelligence*, and it includes audio-visual data analysis methodologies proposed by MARVEL partners within the first 18 months of the project towards achieving the project objectives. The final version of the multimodal and privacy-aware audio-visual intelligence in MARVEL will be documented closer to the end of the project (M30) in D3.5, and it will contain the enriched set of methodologies further improving the Artificial Intelligence (AI) capabilities in MARVEL. These methodologies belong to the audio, visual, and multimodal AI subsystem of the MARVEL architecture (as reported in D1.3 [5]), and they target the third objective of WP3, i.e., to train new or updated ML algorithms for audio-visual classification and analytics. Work reported in this deliverable contributes to Objective 2 of the project, i.e., to deliver AI-based multimodal perception and intelligence for audio-visual scene recognition, event detection, and situational awareness in a smart city environment. Some of the methods described in this document were integrated in the Minimum Viable Product (MVP), as reported in D5.1 [6]. The document starts with a general introduction, providing the purpose and scope of this document, the contributions of the work conducted so far in T3.3 to WP3 and to the project objectives, and its relation to other WPs and deliverables, followed by a description of how the new methodologies proposed so far meet the Key Performance Indicators (KPIs) related to the AI functionalities in MARVEL. Then, the methodologies proposed by MARVEL partners are described in more detail.

Specifically, methodologies targeting audio data analysis tasks are described in Section 2. These include methodologies for Sound Event Detection at the Edge, Sound Event Detection at the Cloud, Sound Event Localisation and Detection, and Automated Audio Captioning. Moreover, the description includes an audio data processing method for learning from unlabelled data and a new concept on how Citizen Science projects can be enhanced through contextual and structural game elements realised through augmented audio interactive mechanisms. Methodologies targeting visual data analysis tasks are described in Section 3, including methodologies for unsupervised Visual Anomaly Detection and rule-based Visual Anomaly Detection. Moreover, a methodology for vision-based social distance estimation and methods for improving performance and/or efficiency of visual data analysis methods based on dynamic inference and dynamic split computing are described, along with methods targeting better training of deep learning models applied to visual data. Methodologies targeting multi-modal data analysis tasks are described in Section 4, including a methodology for Audio-Visual Crowd Counting, and a methodology for learning enhanced audio representations by combining multiple types of information.

The document concludes by providing plans for future work for enriching and further improving the AI methodologies in the remaining period of the project (up to M30) in Section 5 and by providing a summary of the work conducted so far in Section 6. The work included in this deliverable has been reported in 4 journal articles, 6 conference papers, 4 workshop papers, and 1 preprint. The corresponding articles, papers and preprints are included in Appendices 7.1 - 7.15.

1 Introduction

1.1 Purpose and scope of the document

Deliverable 3.1, entitled "Multimodal and privacy-aware audio-visual intelligence – initial version", reports on the activities carried out within Task 3.3 *Multimodal audio-visual intelligence* in the period of M08 (August 2021) to M18 (June 2022). The contributions in this Task are in the form of methodologies that target addressing limitations of existing AI solutions in the context of Smart Cities. These methodologies aim at improving performance and/or efficiency in tasks involving the analysis of audio, visual, and multi-modal data, towards achieving the objectives of the project, and specifically Objective 2. This objective targets to deliver AI-based multimodal perception and intelligence for audio-visual scene recognition, event detection and situational awareness in a smart city environment. Addressing Objective 2 will be implemented by achieving the third objective of WP3, i.e., train new or updated ML algorithms for audio-visual classification and analytics. The report provides a detailed description of the methodologies proposed by the project partners, along with performance evaluations comparing these methodologies with baselines and state-of-the-art methodologies considering the restrictions set by the application scenarios in MARVEL, such as the need of applying the inference on computationally restricted processing units, and time constraints for providing an inference result. The methodologies described in this deliverable will be enriched by work conducted in the remaining period of the project (up to M30), which will be reported in D3.5.

1.2 Contribution to WP3 and project objectives

The work conducted so far in T3.3 and reported in this deliverable contributes to the third objective of WP3 *AI-based distributed algorithms for multimodal perception and situational awareness*, i.e., to train new or updated Machine Learning (ML) algorithms for audio-visual classification and analytics. Methodologies in this task target addressing limitations of existing solutions, and providing state-of-the-art capabilities to the Audio, Visual and Multimodal AI Subsystem of the MARVEL architecture. This subsystem includes functionalities that are implemented in the following components:

- Visual Crowd Counting (VCC), a component estimating the number of people being present in the field of view of a camera;
- Visual Anomaly Detection (ViAD), a component detecting novelties in the contents of the scene based on visual information captured by a camera;
- Sound Event Detection (SED), a component recognising event types based on an audio signal;
- SED@Edge, a component performing SED at edge devices;
- Sound Event Localisation and Detection (SELD), a component detecting the location of events and recognising their types based on audio signals;
- AudioTagging, a component outputting activity of multiple simultaneous sound classes without temporal activity;

- Automated Audio Captioning (AAC), a component generating a caption which describes the information appearing in an audio signal;
- Automated Audio-Visual Captioning (AAVC), a component generating a caption describing the content in a scene captured by a camera (visual) and a microphone (audio);
- Audio-Visual Crowd Counting (AVCC), a component estimating the number of people being present in a scene based on the enriched information provided by a camera (visual) and a microphone (audio);
- Audio-Visual Anomaly Detection (AVAD), a component detecting novelties in the contents of a scene based on audio-visual information captured by a camera and a microphone;
- devAIce, a C++ SDK used to wrap all of AUD's intelligent audio analysis modules, including voice activity detection and acoustic scene classification, which are relevant for MARVEL.
- CATFlow, a system developed by GRN to analyse video footage of road traffic (possibly in real-time) to determine how road users use the given infrastructure.

Fig. 1 illustrates the MARVEL architecture, in which the Audio, Visual and Multimodal AI Subsystem is highlighted. This subsystem, using the above-described components provides the decision-making capabilities based on audio-visual data analysis in the overall MARVEL architecture. As such, the new methodologies developed within T3.3 primarily contribute to Objective 2 of MARVEL, i.e., to deliver AI-based multimodal perception and intelligence for audio-visual scene recognition, event detection, and situational awareness in a smart city environment.

Work performed during the first 18 months of the project proposed methodologies for improving performance and/or speed of AI models targeting eight of the functionalities included in the Audio, Visual and Multimodal AI Subsystem, and specifically, Sound Event Detection (SED), SED@Edge, Sound Event Localisation and Detection (SELD), Audio Tagging (AudioTagging), Automated Audio Captioning (AAC), Visual Crowd Counting (VCC), Visual Anomaly Detection (ViAD), and Audio-Visual Crowd Counting (AVCC). Moreover, a number of methodologies for improving the training of deep learning models for visual data analysis were proposed. Work targeting the Automated Audio-Visual Captioning (AAVC) and Audio-Visual Anomaly Detection (AVAD) functionalities has been planned to be conducted during the remaining part of the project, mainly due to the lack of well-established benchmark datasets which makes methodology development and comparisons with other approaches difficult.

1.3 Relation to other work packages and deliverables

Within WP3, work conducted in T3.3 is connected to work in T3.1 *AI-based methods for audio-visual data privacy*, as the result of data anonymisation (VideoAnony and AudioAnony components) will be used by the AI components of T3.3 when necessary. AI models in T3.3 will be used within the personalised federated learning framework in T3.2 *MARVEL's personalised federated learning realisation for extreme-scale analytics*, and distributed Deep Learning (DL) architectures will be used in T3.4 *Adaptive E2F2C*

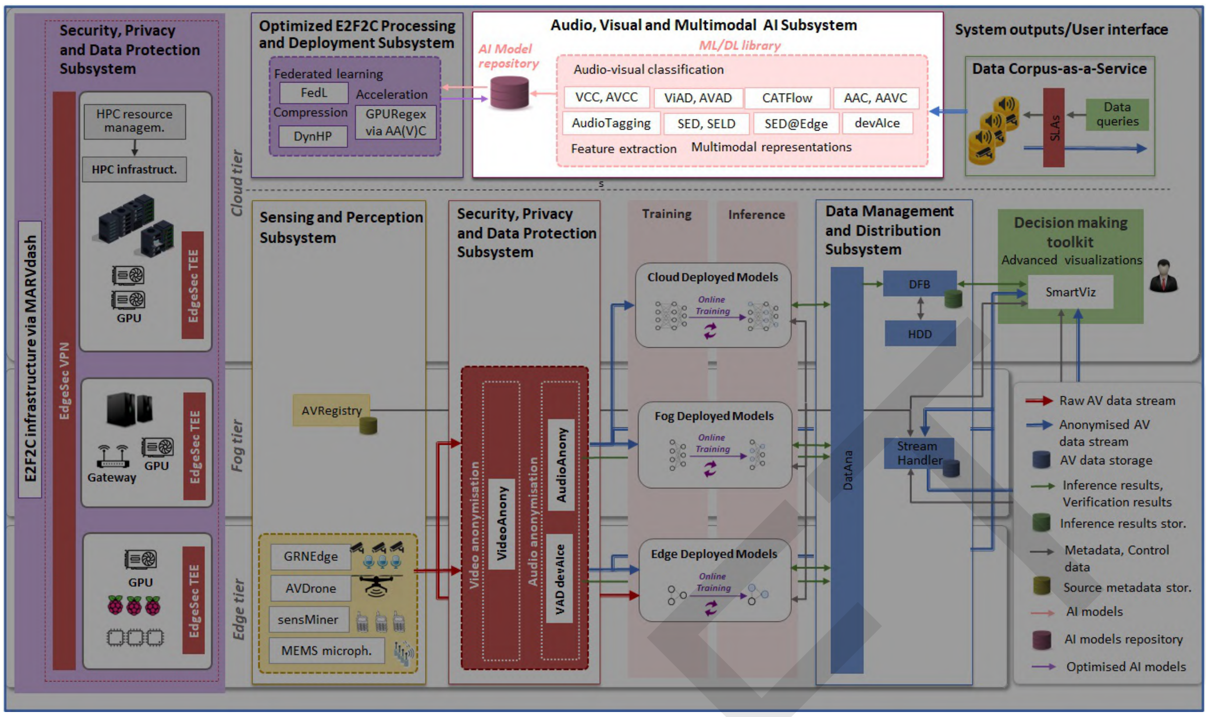


Figure 1: The Audio, Visual and Multimodal AI Subsystem (highlighted) in the MARVEL architecture.

distribution and optimisation of AI tasks for adaptive E2F2C distribution and optimised deployment of AI tasks. ML models in T3.3 will also be optimised in T3.5 *Edge-optimal ML/DL deployment for multimodal processing*, when needed for being deployed at the edge.

Work in T3.3 is also related to multiple tasks belonging to other work packages. Specifically, it is related to tasks T1.1 *The critical role of multimodal analytics in addressing societal challenges*, and T1.2 *Extreme-scale multimodal analytics: progress beyond the state-of-the-art*, which as reported in D1.1 set the scene and describe the advancements in multimodal data analytics required in the context of smart cities. Its results will be contributing to the experimental protocol of the MARVEL project which is defined in T1.3 *Experimental protocol – real life societal trial cases in smart cities environments*, as reported in D1.2. The audio-visual intelligence functionalities in T3.3 and the respective AI components are described in T1.4 *Technology convergence: specifications and E2F2C distributed architecture*, which refines the specification of the conceptual architecture of the MARVEL E2F2C ubiquitous computing framework as reported in D1.3.

The AI models in T3.3 will be trained on the MARVEL data corpus, which will be collected, analysed, managed, and distributed according to WP2 *MARVEL's multimodal data Corpus-as-a-Service for smart cities*, as well as in T4.1 *Optimised audio capturing through MEMS devices*. Moreover, AI functionalities in T3.3 will add to those in T4.2 *openSMILE platform for audio-visual analysis and voice anonymisation*, and will be used in *T4.4 MARVEL's decision-making toolkit* implementing the interactive visualisation and audio-visual analytics tools.

The AI functionalities developed in T3.3 through the components mentioned above will be integrated into the MARVEL architecture, WP5 tasks T5.1 *HPC infrastructure*, T5.2 *Resource management and optimised automatic usage*, T5.3 *Continuous integration* *towards MARVEL's framework realisation*. The effectiveness of the provided AI solutions will be evaluated in T5.4 *Quantifiable progress against societal, academic, and industrial validated benchmarks* and in T5.5 *From the prototype to the final solution*. Three components (SED, VCC, and AVCC) were included in the MARVEL Minimum Viable Product (MVP) as reported in D5.1 [6], and the initial technical evaluation and progress against benchmarks were reported in D5.2 [7]. Finally, the work in T3.3 is connected to WP6 *Real-life societal experiments in smart cities environment*, where the above-mentioned AI components will be used for decision-making in real smart city environments.

1.4 Connection to Key Performance Indicators

Work within T3.3 is connected to a number of methodology-related Key Performance Indicators (KPIs) set to achieve Objective 2 of the MARVEL project and impact-related KPIs (iKPIs). Work conducted so far by project partners contributes to addressing them as summarised in the following. All numbers refer to experimental results conducted on specific datasets, as described in Sections 2, 3 and 4, and the corresponding research papers and preprints copied in the Appendices. The work towards achieving all set KPIs will continue in the second half of the project, and will be reported in D3.5 *Multimodal and privacy-aware audio-visual intelligence – final version* which will be delivered in M30.

KPI-O2-E2-1: *Average accuracy enhancement for audio-visual representations and models at least 20%*

AU developed a method for Audio-Visual Crowd Counting (AVCC) which provides 3.06% improvement (in mean absolute error) against the state-of-the-art (Section 4.1). AU also developed an efficient method for Visual Crowd Counting (VCC) employing Early Exit Branches to reduce the parameter count of the resulting model (Section 3.1). The resulting models achieved improvement up to 11.11% (in mean absolute error) compared to the competing baseline. Moreover, AU developed methodologies for improving the training of deep neural networks through better initialisation of their parameters (Section 3.6) and by increasing the diversity of representations learned by different neurons in each layer (Section 3.7), which improved performance compared to the baselines on visual data classification.

KPI-O2-E3-1: *Increase the average accuracy for audio-visual event detection by at least 10%.*

TAU developed a Sound Event Detection (SED) method based on a light-weight Convolutional Recurrent Neural Network (CRNN) obtained by replacing Convolutional Neural Network blocks with depth-wise separable convolutions, and replacing recurrent neural network blocks with dilated convolutions, and by utilising data augmentation techniques to diversify data (Section 2.2). The resulting model led to 56% increase (in macro-averaged F1-score) compared to the competing method. TAU developed SED method based on audio tagging approach, where tagging is applied inside consecutive segments to get sound event detection output. The system was based on Convolutional Neural Network (CNN) architecture (Section 2.2), and the resulting model led to 76% increase (in macro-averaged F1-score) compared to the competing method.

KPI-O2-E3-2: *Increase the average accuracy for unsupervised audio-visual event detection at the edge by at least 10%.*

AU developed an Unsupervised Visual Anomaly Detection (ViAD) method based on structured parameter pruning on Memory-augmented Deep Autoencoder (Section 3.2) which improves performance by 1.2% while reducing the model size and its number of floating point operations (FLOPs) by 71.78% and 21%, respectively.

KPI-O2-E3-3: *Decrease the time needed to identify an event by at least 30% of current time*

FBK developed edge solutions for SED based on compact neural models with low parameter and parameter count (Section 2.1). These solutions are suitable for low-resource pervasive devices. Being able to detect sound events very close to the microphones they eliminate the latency due to data transfer to the cloud.

TAU developed a Sound Event Localisation and Detection (SELD) method based on self-attention on learned audio features (Section 2.3). The resulting models have almost double the number of parameters compared to the competing method. However, they were benchmarked to be 2.5 times faster than the competition during the inference due to parallelisation achieved with the self-attention blocks, i.e., they lead to a decrease in the inference time of 40%.

AU developed an efficient method for Visual Anomaly Detection (ViAD) which employs structured pruning for reducing the parameters of a Memory-Augmented Deep Autoencoder network (Section 3.2). The resulting model led to a reduction of the parameter count equal to 71.76% and a decrease of the FLOPs equal to 21%, while slightly improving accuracy (AUC).

AU and UNS developed a Dynamic Split Computing method (Section 4.3) which determines the optimal neural network split for achieving the fastest execution time considering the structure of the neural network, the data batch size, and the transmission channel data rate.

iKPI- 3.2: *At least 20% reduction in code complexity (lines of code, amount of scripts, data handling) required due to the adoption of programmer-friendly public frameworks providing implementations of new deep learning models.*

All developed software implements easy-to-use implementations and interfaces based on widely-adopted ML and DL libraries to reduce the number of lines needed for model training and for deploying the resulting models.

iKPI-3.3: *At least three (3) approaches tested for ML training algorithms.*

FBK investigated the use of Knowledge Distillation for training edge neural models (Section 2.1).

TAU employed data augmentation techniques specAugment and mixup to improve the training of the Sound Event Detection (SED) models (Section 2.2). TAU also proposed a continual learning methodology for training Automated Audio Captioning (AAC) models (Section 2.4), an active learning method for unsupervised training of deep learning models (Section 2.5), and a cross-modal contrastive learning method for improving performance in audio data analysis using other data modalities (Section 4.2).

AU employed curriculum learning (Section 3.5) and Copycat finetuning (Section 3.5) for training deep learning models with early exit brances, and unsupervised learning of Convolutional Autoencoders (Section 3.2). AU also developed methods for improving the parameter initialisation of deep learning models based on Discriminant Learning (Section 3.6), as well as training the deep learning model parameters by increasing their diversity (Section 3.7).

1.5 Structure of the document

The structure of this document is as follows: Section 2 provides a description of the audio data analysis methodologies, Section 3 a description of the visual data analysis methodologies, and Section 4 a description of the multimodal data analysis methods proposed by MARVEL partners. Section 5 discusses future plans for further enriching the AI methodologies in the remaining period of the project, and Section 6 provides a summary of the work conducted so far. The work included in this deliverable has been reported in 4 journal articles, 6 conference papers, 4 workshop papers, and 1 preprint. The articles, papers and preprints reporting the detailed description and evaluation of the new methodologies included in this deliverable are appended at the end of the document.

2 Methodologies for audio data analysis

In this Section, methodologies developed by MARVEL partners targeting audio data analysis are described. These include methodologies for Sound Event Detection at the Edge (Section 2.1), Sound Event Detection at the Cloud (Section 2.2), Sound Event Localisation and Detection (Section 2.3) and Automated Audio Captioning (Section 2.4). Moreover, the description includes an audio data processing method for learning from unlabelled data (Section 2.5), and a new concept on how Citizen Science projects can be enhanced through contextual and structural game elements realised through augmented audio interactive mechanisms (Section 2.6).

2.1 Desing and Optimisation of Scalable Neural Network for end-to-end environmental sound classification on low-end devices (SED@Edge)

2.1.1 Introduction and objectives

The goal of Sound Event Detection (SED) is to identify and classify relevant events in audio streams with application in the smart city domain (e.g., crowd counting, alarm triggering), thus is an asset for municipalities and law enforcement agencies. Given the large size of the areas to be monitored and the amount of data generated by the IoT sensors, large models running on centralised servers are not suitable for real-time applications. Conversely, performing SED directly on pervasive embedded devices is very attractive in terms of energy consumption, bandwidth requirements, and privacy preservation. In this section, we describe the use of scalable neural network backbones from the PhiNets architectures' family for the design of real-time sound event detection on low-power low-resource devices (e.g. microcontrollers). The corresponding papers are listed below, and can be found in Appendices 7.1 and 7.2:

- [8] F. Paissan, A. Ancilotto, A. Brutti, E. Farella, "Scalable neural architectures for end-to-end environmental sound classification", IEEE International Conference on Accoustics, Speech, and Signal Processing (ICASSP), 2021.
- [9] A. Brutti, F. Paissan, A. Ancilotto, E. Farella, "Optimizing PhiNet architectures for the detection of urban sounds on low-end devices", European Signal Processing Conference (EUSIPCO), 2022.

As described better in the performance evaluation of the method provided in subsection 2.1.4, SED@Edge performs segment-wise classification and is therefore evaluated here in terms of classification accuracy. This is mostly related to the nature of the datasets used in the evaluation of the component, like UrbanSound8K, and simplifies the benchmarking against state-of-the-art solutions for urban sound classification and detection and the KPIs evaluation. It is worth clarifying that, in the current configuration, the component performs classification of isolated events. In order to achieve actual sound event detection, the component will have to be applied on consecutive (overlapping) segments.

2.1.2 Summary of the state-of-the-art

In literature, many architectures for SED have been proposed, driven by the recent DCASE series [10, 11, 12]. This section will briefly review the state-of-the-art techniques with a specific focus on solutions for embedded platforms.

The most common approaches use convolutional neural networks (CNNs) to process spectrograms, obtained as stacked Fourier transforms [13]. The best performing approaches so far are VGGish [14], PiczakCNN [15], and SB-CNN [16]. These three neural network architectures have different structures but share a high parameter count, with the smallest one being SB-CNN which counts 241K parameters. SB-CNN was presented in [16] alongside data augmentation techniques, which proved to be the most effective technique to train networks on the UrbanSound8K benchmark, given the small size of the dataset. Overall, these architectures are constituted by a massive number of parameters that could not fit on off-the-shelf Microcontroller Units (MCUs).

An emerging trend in audio classification is exploiting one-dimensional convolutions (1DConvs) directly on the audio waveforms. In this case, neural networks learn the filters to be applied to the input audio signal directly. Many approaches that inject previous knowledge into the filter shape are proposed to ease the training. In SincNet [17] the filters of the first convolutional layer are forced to be band-pass filters. In [18] the proposed architecture exploits the Gammatone filter initialisation. In ENVNET-V2 [19], 1DConvs are employed with bi-dimensional convolutions on the feature map. Despite the good performance, this comes with a high cost in computational requirements (more than 1M parameters). AudioCLIP [20] learns a bi-dimensional representation of the waveform with a custom neural network backbone [21] and is currently the best performing model on the UrbanSound8k benchmark [22]. However, this high accuracy is achieved by employing extremely large architectures, counting up to 30M parameters only for the feature extraction. In Wavelet Networks [23], instead, the architecture resembles the wavelet transform to maximise the sound event detection performance by reducing the impact of phase shifts in the signal. Overall, models working on the waveform are less accurate in classifying acoustic events, mainly due to the higher variability of the signals in the time domain. On the positive side, the 1DConv-based models have a higher parameter efficiency given that they need to run in only one dimension.

Audio processing at the edge (i.e. on embedded platforms) is relevant for both research and industrial applications. Many approaches targeting embedded platforms are already available in the literature for a variety of audio tasks, namely keyword spotting (KWS) [24, 25] and SED [26, 27]. In [26], a student-teacher approach is presented for model compression via knowledge distillation based on the joint alignment of the latent representations and cost function optimisation for classification. This approach shows promising results in compressing architectures. However, there is an implicit upper bound to the network's performance since it is empirically shown that the performance of the student network will not surpass that of the teacher. [27] proposes a novel architecture where a dilated convolution replaces the recurrent unit. Moreover, the implementation exploits depth-wise separable convolutions [28], which are well-known for their parameter efficiency. Despite this, the parameter count of the architecture is still higher than what could fit on an MCU. Network quantisation offers another popular approach [29]. The most computationally efficient systems use Binary Neural Networks (BNNs) [30] but compromises classification performance.

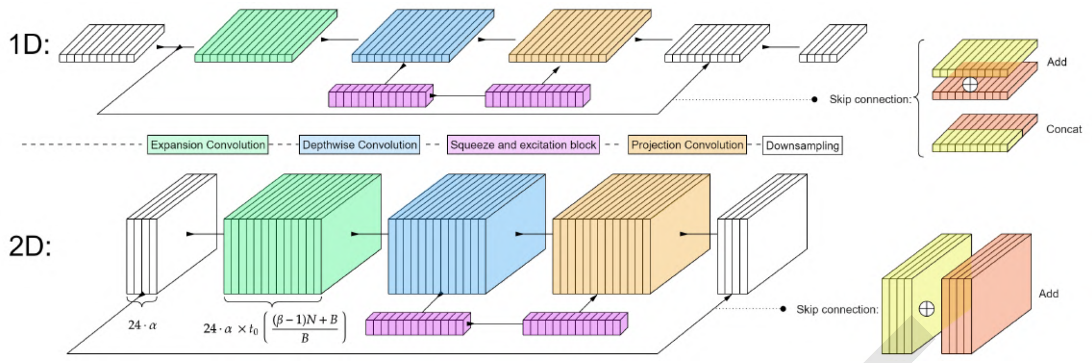


Figure 2: Illustration of the 1D and 2D convolutional blocks. The input map is fed into the expansion convolution, which affects only the number of channels in the feature map. The feature map is fed into a depth-wise convolution followed by a squeeze-and-excite block. The output of the squeeze-and-excite is projected in a lower dimensionality space via a bottleneck layer. At the end of the 1D block, there is an optional downsampling - implemented using average pooling - and a skip connection - either `ADD` or `CONCAT` depending if the layer realises a downsampling operation or not. For the 2D block, the skip connection always uses an `ADD` operation. In the illustration, B is the number of blocks and N is the ID of the current block.

2.1.3 Description of work performed so far

When bringing neural architectures on MCUs, one of the most efficient approaches is *hardware-aware* scaling [1]. Using this paradigm, it is possible to optimise the neural network architectures to fit on embedded platforms with a negligible drop in performance. However, in order to exploit this scaling principle, we need to avoid an exponential decay of the performance with respect to computational requirements, as often occurs [31]. For these reasons, we present two different architectures (*PhiNets 1D* and *PhiNets 2D*) that are in line with the *hardware-aware* scaling paradigm and work on two different multiply–accumulate (MAC) and memory ranges. We exploit the scalability principles of *PhiNets* [1], a scalable neural network backbone based on a sequence of inverted residual blocks (depicted in Fig. 2), where the shape of each block depends on three hyper-parameters α, β, t_0 that control disjointly the MAC count and memory requirements (FLASH and RAM), respectively, as described in [1].

***PhiNets* on spectrograms:** We introduced some modifications to the original *PhiNets* architecture to tailor it to the SED task and improve the classification performance. In particular, we propose down-sampling the feature map using max-pooling instead of strided convolutions. We also replace the original input block with a standard 2D convolution. Max-pooling improves the network's overall performance by close to 5% on the UrbanSound8K dataset, and changing the input block showed a similar trend. We observed that networks with lower than 2K parameters and 5M MAC perform better using a strided convolution for downsampling and a depth-wise separable input block. Despite this small change, the computational load of the *PhiNets* architectures does not change and is analytically described in [1].

***PhiNets* on waveform:** To further reduce the computational cost of the *PhiNets*, we propose a variation of the architecture that works on waveforms, thus exploiting one-dimensional convolutions (1DConvs). By doing this, the relationship between the com-

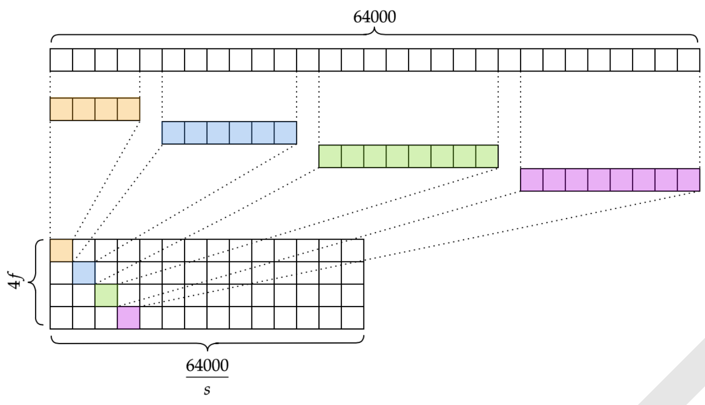


Figure 3: Illustration of the input 1D convolutional block. The stacked convolutions work on the features, which are time-stretched with different phases. In the illustration, f represents the number of filters, while s represents the stride of the convolution.

putational requirements and the number of filters used in each convolutional block is linear instead of quadratic. Moreover, the overall parameter count is lower.

The network architecture is split into three main blocks. The first block is a convolutional block that aims at reducing the shape of the input tensor allowing for a trade-off between accuracy and MAC count. This block consists of four vertically stacked 1DConvs with different kernel sizes (namely 32, 64, 128, and 256 points), which work on time-stretched versions of the waveform to avoid losing information in the striding process, as described in Fig. 3. This first convolutional block is followed by a sequence of convolutional blocks composed of a point-wise convolution to up-sample the features, a depth-wise convolution, a squeeze-and-excite block, and another point-wise convolution to restore the same number of features as the input. At the end of the network, a fully-connected layer for classification compresses the extracted features and outputs the logits for each class. To decrease the computational complexity and to help the convergence of the network [32], we exploit skip connections in the convolutional blocks. In particular, we either (i) concatenate the input and output tensors to double the number of features used in the following layers (instead of increasing channels by means of e.g. a point-wise convolution) or (ii) sum the input and output tensors. Fig. 2 shows the convolutional block of the *PhiNets* and how it is performed both in one and two dimensions.

To scale the computational requirements of *PhiNets 1D*, we can change the number of convolutional blocks, the depth-multiplier, and the number of filters in the first convolution or the stride of the input convolution. In particular, changing all of the above parameters has a linear impact on both computational costs (i.e., MAC count and parameter count) except for the number of filters in the first convolutional block. The latter has a quadratic impact on both parameter and MAC count. Such scalability features allow for extreme model compression and optimisation, while decoupling parameter count and computational cost in alignment with the *harware-aware* scaling paradigm.

2.1.4 Performance Evaluation

We benchmarked the two proposed architectures, for waveforms and spectrograms, on the UrbanSound8K dataset [22]. The dataset consists of a collection of 8,732 samples of 4-second long typical urban sound events, equally distributed among ten different classes (air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music). The sampling rate of the original audio sample varies, so we re-sampled each event at 16kHz resulting in 64,000 timepoints per sample. We used the standard 10-fold benchmarking procedure for this dataset by averaging the test score after training on eight folds and using one for validation. We augmented the dataset with pitch shift, time-stretching, and Gaussian noise. The model input consists of 40 mel-spectrograms computed on the 4s sample using 2,048 sample windows with a hop-length of 512, resulting in 120 frames for each sound event. For the waveform model instead, we used the re-sampled signal, thus leading to a 64,000 entries input vector.

We trained the models on spectrograms for 100 epochs, with a 1×10^{-3} learning rate, 1×10^{-2} weight decay and 0.05 dropout rate in the convolutional blocks. Moreover, we also added label smoothing to help the network avoid over-fitting. For the waveform model, we decreased the learning rate starting from 6×10^{-4} every time the validation accuracy was not improving for 15 consecutive epochs. Moreover, we used L2 regularisation as for the other approach.

Tables 1 and 2 report the performance of the proposed method compared against state-of-the-art architectures using spectrograms and waveforms as input features. The central column reports the parameter count. The 10-fold accuracy is taken from the original papers. When the standard deviation is not available in the paper, it is not reported in the Tables. The notation for the models is taken from the original papers. In particular, M_{20k} in [33] refers to the order of magnitude of the parameter count.

Table 1: Comparison between *PhiNets* and other state-of-the-art architectures, using spectrograms as input features.

Input	Model	Params (K)	10-fold acc
Spectrogram	PICZAKCNN [15]	26 000	73.7
	SB-CNN [16]	241	73.11
	VGG [14]	77 000	70.74
	Cerutti M_{20k} [33]	30	69
	Cerutti M_{200k} [33]	200	72
	Cerutti M_{2M} [33]	2 000	75
	Cerutti M_{20M} [33]	70 000	76
	PhiNets M40	27.1	**76.3** $\pm$ 5.6
	PhiNets M15	32.2	76.1$\pm$ 5.0
	PhiNets M5	3.80	68.8$\pm$ 3.1
	PhiNets M3	2.18	65.3$\pm$ 1.6
	PhiNets M1.5	2.00	62.3$\pm$ 3.9

To demonstrate that scaling *PhiNets* has a marginal impact on the classification accuracy with respect to the compression factor, we benchmarked models in the 0.1-20 MMAC range and with 0.7-30 thousand parameters, which is a typical range for real-time operation with off-the-shelf MCUs. For reproducibility, the generated models are enumerated in Table 3.

For a better understanding of the impact of the scaling parameters on model size and performance, we carried out an empirical study to highlight the effects of α and t_0. β will be kept at the default value ($\beta = 1$), as all networks tested require so few

Table 2: Comparison between *PhiNets* and other state-of-the-art architectures using waveforms as input features.

Input	Model	Params (K)	10-fold acc
Waveform	AudioCLIP [20]	>30 000	90.01
	ENVNET-V2 [19]	101 000	78
	W11-NET-WL [23]	1 806	68.47± 4.914
	W18-NET-WL [23]	3 759	65.01 ± 5.431
	W34-NET-WL [23]	4 021	66.77± 4.771
	1DCNN [18]	453	62± 6.791
	W-1DCNN-WL [23]	458	62.64± 4.979
	PhiNets 1D M1	**11.5**	59.3± 3.7
	PhiNets 1D M0.5	5.91	56.4± 6.4
	PhiNets 1D M0.2	2.11	48.4± 2.5
	PhiNets 1D M0.1	1.15	46.3± 4.2
	PhiNets 1D M0.07	0.766	43.3± 2.6

Table 3: Parameters for generating the neural network architectures. For the spectrogram classification model, the notation is the same as in [1]. For the waveform model, d refers to the depth multiplier while n refers to the number of blocks.

	Model name	Conv Type	α	B	t_0	**MMAC**	**Param. (k)**
Spectrogram	PhiNets M_{40}	Conv2D	0.5	3	4.0	43.00	27.1
	PhiNets M_{15}	SeparableConv2D	0.5	2	5.0	14.43	32.3
	PhiNets M_5	Conv2D	0.2	2	2.0	4.72	3.80
	PhiNets M_3	Conv2D	0.1	2	4.0	2.71	2.18
	PhiNets $M_{1.5}$	SeparableConv2D	0.1	2	2.0	1.59	2.00
	Model name	Conv Stride	Conv Filt	n	d	**MMAC**	**Param. (k)**
Waveform	PhiNets 1D M_1	300	3	4	4.5	1.34	11.50
	PhiNets 1D $M_{0.5}$	500	2	4	4.5	0.40	5.91
	PhiNets 1D $M_{0.2}$	1600	2	3	2.5	0.06	2.11
	PhiNets 1D $M_{0.1}$	300	1	4	1.5	0.15	1.15
	PhiNets 1D $M_{0.07}$	500	1	3	1.5	0.07	0.766

parameters that even the smallest MCUs can store them with a considerable margin. We vary α considering [0.20, 0.35, 0.50] possible values and t_0 in [2, 4, 6]. Note that in this way, we cover different architectures with a very similar number of parameters. Given the small sizes of the resulting PhiNet models, besides training them from scratch, we also investigate the use of Knowledge Distillation (KD) from a larger plain-conv2d model, using both soft and one-hot labels. Table 4 reports the sound event detection accuracy obtained by training the models from scratch, as well as using knowledge distillation, considering different configurations. The table also reports the parameter count for each configuration.

While the performance of the models trained from scratch decays rather linearly with the number of parameters, the specific configuration of the hyper-parameters α and t_0 does not seem to have a direct and evident impact on the performance. Overall, this was expected as the PhiNet architectures are designed to scale efficiently in the MCU range without significantly compromising the network's performance. However, it is worth noting that, in some cases, using larger values of t_0 is preferable with

Table 4: Accuracy on UrbanSound8K varying the α and $t0$ scaling parameters, with and wihtout KD. The table reports also the model parameter count.

α	$t0$	Acc	Acc-KD	# Parameters
0.20	2	64.87	49.68	4,779
0.35	2	64.64	64.82	12,479
0.50	2	63.90	67.61	24,507
0.20	4	65.85	59.58	8,893
0.35	4	71.80	67.14	23,797
0.50	4	72.25	70.15	47,349
0.20	6	66.05	70.95	13,007
0.35	6	68.02	71.05	35,115
0.50	6	70.39	71.20	70,191

respect to α given a target hyper-parameter count (compare for example the two models (0.5;2) and (0.35;34)). This could be related to the fact that larger convolutional blocks can better represent the information, easing the learning task. Finally, note that the best accuracy (72.25%) is achieved using a medium-size architecture (47K parameters obtained with $\alpha = 0.5$ and $t_0 = 4$). PhiNets are actually designed to be efficient in the MCU range. Therefore, they tend to overfit easily when the model size increases. This issue is further accentuated by the relatively small size of the dataset used in our experiments.

2.1.5 Future Work

As future work, we plan to expand the 1DConv model with different input convolutional layers shapes (e.g., Sinc, Wavelet) to boost the models' performance. Finally, in terms of component deployment, we will focus on adapting the component in order to achieve actual sound event detection by processing consecutive overlapping fixed-length audio segments.

2.2 Sound event detection methods for smart city applications

2.2.1 Introduction and objectives

The aim of *sound event detection* (SED) system is to provide a textual label, a start and end time to each sound event instance it recognises in an acoustic scene [34]. Overview of sound event detection is shown in Figure 4. A *sound event* is defined to be a textual label that humans would use to describe a sound-producing event, and these labels allow people to understand the concepts behind events and associate these events with other known events. Therefore, the detection of sound events can be used to gain an understanding of the content of audio recordings in the smart city domain.

Sound event detection systems can be roughly categorised based on their ability to handle simultaneous sound events. A monophonic sound event detection system is able to output only a single sound event at a time, while a polyphonic sound event detection system is capable of outputting multiple simultaneous sound events. Currently, the state-of-the-art SED systems are not able to output multiple sound event instances of the same class at the same time so the polyphony is here defined in terms of distinct event classes used in the system. The audio content analysis systems which are outputting

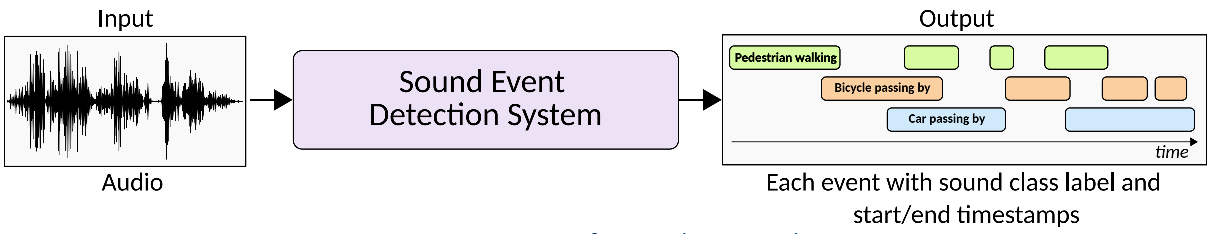

Input
Audio
Sound Event Detection System
Output
Pedestrian walking
Bicycle passing by
Car passing by
time
Each event with sound class label and start/end timestamps

Figure 4: Overview of sound event detection.

activity of multiple simultaneous sound classes without temporal activity are called audio tagging systems. These systems are closely related to SED systems and they can be easily extended to output temporal activity by applying audio tagging in consecutive short time segments. Often the distinction between SED and audio tagging depends on the application [35].

Sound event detection methods discussed in this section directly contribute to the SED component in the MARVEL project. The methods are focusing on approaches suitable for smart cities and to be deployed in an HPC environment. The focus of the study was on approaches for the detection of different vehicle types and overall systems which could be deployed for real-time sound event detection with CPU-only configurations in HPC. The previously discussed sound event detection approach SED@Edge (see Section 2.1) focuses on approaches suitable for edge devices with limited computational resources.

2.2.2 Summary of the state-of-the-art

The state-of-the-art on sound event detection is based on deep learning, and neural network architectures such as CNN [36] and Convolutional Recurrent Neural Network (CRNN) [37] are often used. Generally, these works use handcrafted acoustic features, such as mel-band energies, or directly raw audio signal as input [38]. Deep learning methods require large datasets for robust learning, however, detailed annotations with exact timestamps for start and end of sound events will become soon impractical to produce when dataset sizes are getting larger. Because of this, most open datasets available for SED development are relatively small. Some works utilise data augmentation techniques to increase data diversity while training neural networks. Regularly utilised data augmentation techniques include audio signal manipulation (e.g. time stretching, pitch shifting, and dynamic range compression) [39], simulation of various microphones and acoustic environments (convolution with impulse responses) [40], simulation of various noise conditions (adding background noise) [39], specAugment (time and frequency masking spectrograms) [41], block mixing (mixing additively blocks of audio and their annotations) [42], and mixup (combining blocks of audio as a weighted sum) [43, 44]. In addition to data augmentation techniques, some works use transfer learning techniques to cope limited amount of learning examples from target sound event classes. In these techniques, the acoustic model is first trained to solve a secondary task using a large amount of data and the outputs of this pre-trained model are then used to produce new features, *embeddings*, that are used in the actual target task as input features [45, 36, 46]. Most recent sound event datasets contain only segment or recording level annotation for the sound event activity (weak annotation) as they are easier to produce in larger volumes. To use such datasets for the training of sound event detection systems requires weakly-supervised learning approaches [47, 48].

The target application for the method development was vehicle type detection. This was selected to be aligned with the requirements of use case GRN4 *Junction Traffic Trajectory Collection*. There has been limited prior work in the literature on this application domain [49, 50]. Zinemanas et al. [49] presented the MAVD-traffic dataset with 2.5 hours of annotated audio together with two solutions for vehicle recognition where analysis is done in one-second segments. The dataset is later used in the performance evaluation for our SED system. The first proposed solution in [49] is based on mel-frequency cepstral coefficients (MFCCs) acoustic features extracted on segments and random forest classifiers and the second solution using mel spectrograms as acoustic features and classification is utilising neural networks with CNN architecture [51]. The CNN model is pre-trained with material from the URBAN-SED dataset [51], and the acoustic model is fine-tuned with the MAVD-traffic dataset for the vehicle recognition task. These systems were evaluated for vehicle recognition task with three classes (car, bus, and motorcycle). In [50], a benchmark dataset IDMT-Traffic for acoustic traffic monitoring research was proposed. The dataset contains 4,718 2-second long audio segments (in total 2.5 hours of audio) captured next to the road. The data is annotated by labelling segments with vehicle type (car, bus, motorcycle, truck), traffic direction, speed limit (30, 50, or 70 km/h), and weather conditions (wet or dry road). The neural network-based vehicle type classifier is proposed and experiments show a good performance was achieved with VGGnet and ResNet network architectures. The most prominent misclassification was observed between truck and car classes. In this stage, the IDMT-Traffic dataset was not used in our SED component development as it does not contain an exact timestamp for the event starts and ends (weak annotations), and we did not want to limit the selection of our sound event detection approaches based on the style of the annotation. The MAVD-traffic contains exact annotations for the event starts and ends (strong annotations), and it can be used to test direct sound event detection approaches as well as sound event detection implemented through audio tagging in fixed-length audio segments.

2.2.3 Description of work performed so far

Vehicle type detection was selected as the target application for the development, and the method development focused on approaches relying on supervised learning. For training these methods, audio material with exact start and end timestamps for the active sound events is required. As the open datasets suitable for vehicle type detection research are relatively limited, in the initial stage of the development we studied the possibility of using transfer learning-based approaches. Most of the pre-trained audio embeddings are designed for near-field sounds and for full bandwidth signals (32kHz-44.1kHz). Furthermore, current state-of-the-art pre-trained audio embeddings are also computationally heavy and not well-suited to be used in CPU-only configurations in HPC. As our target material in the use case GRN4 has mostly sounds that are in far-field, the audio signal has a low sampling rate (16kHz), and the computational resources are part of the consideration, the development focused on methods that are learned fully on target data so that we have full control of neural architecture during the development.

Two approaches for sound event detection are studied; one processing consecutive non-overlapping 10-second segments and applying sound event detection on them (later called as detection approach), and one processing consecutive non-overlapping one-second segments and applying sound event activity recognition inside the segments (later called as tagging approach). The first approach will produce better time-resolution for event activity start and end times, resolution effectively being the hop length of the feature extraction. At the same time, the overall detection latency of the system in the real-time situation will be longer. The second approach simplifies the problem into audio tagging within one-second segments. The time-resolution of the event activity start and end is relatively rough (one second) in this case, but at the same time, the overall detection latency is lowered to one second. Because of this, the approach is suitable for use cases where the system needs to perform under real-time constraints and lower time-resolution can be tolerated.

In the detection approach, the CRNN network architecture is used. CRNN is a general-purpose network architecture for sound event detection containing convolutional and recurrent layers [37]. In this architecture, the convolutional layers act as feature extractors, recurrent layers have the role of learning the temporal dependencies in the sequence of features extracted by the convolutional layers, and the final fully-connected layer will produce the output with sigmoid activation. Input data for the model is log mel energies (40 mel bands) extracted in 80ms analysis windows with 40ms hop length. The model produces output at the same time resolution as the input is (40ms). The model consumes 256 consecutive feature vectors (10.24-second segment). A typical CRNN neural model has 3.5M parameters, and parameters of the convolutional layers (e.g., number of channels) and weight matrices of recurrent layers contribute directly to this. A computationally lighter CRNN model is produced by replacing convolutional neural network blocks with depthwise separable convolutions, and replacing recurrent neural network blocks with dilated convolutions [52]. The resulting neural network model has only 290K parameters. These two methods are denoted as CRNN and CRNN-DESED.

In the tagging approach, the convolutional neural network (CNN) network architecture is used. In this architecture, the convolutional layers act as feature extractors, global pooling summarises the feature maps into a fixed-length vector, and fully connected is used to produce the output with sigmoid activation. Input data for the model is log mel energies (64 mel bands) extracted in 40ms analysis windows with 20ms hop length. The model produces one output vector for 50 consecutive feature vectors (1-second segment). Two network structures introduced for PANNs audio embeddings [36] were studied in this work: a 6-layer CNN with four convolutional layers (kernel size 5*5) based on AlexNet [53], and a 10-layer CNN with 4 convolutional layers based on VGG-like CNNs [54]. These two methods are denoted as CNN6 and CNN10, and the resulting neural network models have 4.57M and 4.95M parameters.

During the training, data augmentation techniques specAugment [41] and mixup [43] are used to diversify the data.

2.2.4 Performance evaluation

The MAVD-traffic dataset [49], later denoted as MAVD, was used as an application-specific dataset during the development until use case-specific datasets become available. The dataset contains 47 5-minute long recordings (in total 2.5 hours of audio) with 4,718 manually annotated passing vehicles. The dataset is released with a cross-validation setup where 24 audio recordings captured in one location are used for training, 7 audio recordings captured in the same location are used for validation, and 16 audio recordings captured in the other three locations are used for testing. Three event label classes (*bus*, *car*, *motorcycle*) are used from this dataset in the evaluations to align the performance measurements with results reported in [49]. Audio signals have been captured with a 44.1kHz sampling rate. The dataset does not contain recording with the wind in the microphone, as these sections were removed in the data preparation stage.

The dataset collected from the use case GRN4, denoted later as GRN-SED, was used in a later stage of the development. This dataset contains 2 hours of audio data captured from traffic surveillance cameras in Malta in three locations, with manually annotated activity for four vehicle classes (*bus*, *car*, *truck*, *motorcycle*). The audio was captured with microphones inbuild into the surveillance cameras, and the cameras were positioned high to oversee the monitored street. In relation to the microphones, the sound sources can be considered to be relatively far and there are inferring noises such as construction noise and wind in the microphone. Audio signals were captured with a 16kHz sampling rate. An additional dataset, GRN-SED-Mobile, was collected in Malta using smartphones as a recording device to collect material specific to the traffic in Malta. The recording was done from the side of the road, so in this case, sound sources can be considered to be near-field part of the time. Dataset statistics for the used datasets are shown in Table 5.

Table 5: Dataset statistics for MAVD-traffic dataset and GRN4 use case-specific dataset.

	MAVD		GRN-SED		GRN-SED-Mobile	
Event labels	Event instances	Total duration	Event instances	Total duration	Event instances	Total duration
bus	1013	116 min	25	5 min	103	10min
car	1661	304 min	958	281 min	774	61 min
motorcycle	421	43 min	82	10 min	107	8 min
truck	60	6 min	123	41 min	27	3 min
Total	3155		1188		1011	

Segment-based F-score and error rate (ER) metrics are used as evaluate performance [55]. The metrics are calculated in one-second segments, and the sound event activity is compared between reference annotation and the output. Intermediate statistics, true positive (TP), true negative (TN), false positive (FP), and false negative (FN), are calculated within non-overlapping segments. F-score is calculated by first accumulating the intermediate statistics over the evaluated segments for all classes and summing them up to get overall intermediate statistics (instance-based metric, micro-averaging). F-score value is calculated as $P = TP/(TP + FP)$, $R = TP/(TP + FN)$, $F = (2 * P * R)/(P + R)$. The F-score is mainly determined by the number of true positives, which in turn is dominated by the performance in the classes with a large number of tested items. In the case of very unbalanced evaluation datasets, it is preferable to use class-based averaging (macro-averaging) as an overall performance measurement. This F-score is calculated for each class based on the class-wise intermediate statistics, and the overall F-score is get by averaging the class-wise F-scores. F-score is commonly used to evaluate the system performance in sound event detection.

Error rate (ER) measures the number of errors in terms of substitutions (S), insertions (I), and deletions (D) that are calculated per segment from intermediate statistics. ER is calculated by summing the segment-wise counts for S, D, and I over the evaluated segments and normalising with the total number of segments.

$$ER = \frac{\sum_{k=1}^{K} S(k) + \sum_{k=1}^{K} D(k) + \sum_{k=1}^{K} I(k)}{\sum_{k=1}^{K} N(k)}. \quad (1)$$

The error rate metric is commonly used to evaluate the system performance in speech recognition, speaker diarisation, and sound event detection. Interpretation of the metric can be difficult sometimes, as the metric value is a score and can be over 1 if the evaluated system produces more errors than correct estimations. Furthermore, the exact value of 1.0 is a trivial case as it can be achieved with the system outputting no active events. Therefore sound event detection evaluation in this work is done by using segment-based F-score and ER together to get a more comprehensive performance estimate for the system.

Table 6: Sound event detection performance on MAVD dataset (3 sound event classes) with varying approaches and audio sampling rates.

	Micro-avg		Class-wise F1			Macro-avg
Model	F1	ER	Bus	Car	Motorcycle	F1
44.1 kHz signals						
Baseline (S-CNN [49])	55.5	0.51	8	68	0	**25.3**
CRNN	56.5	1.08	40	70	11	**40.3**
CRNN-desed	52.9	1.34	36	68	15	**39.6**
CNN6	59.3	0.99	47	71	21	**46.5**
CNN10	57.4	1.03	44	71	19	**44.6**
16 kHz signals						
CRNN	52.9	1.21	34	70	16	**39.9**
CRNN-desed	50.7	1.56	36	68	15	**39.5**
CNN6	54.6	1.16	38	71	18	**42.6**
CNN10	56.4	1.03	40	73	18	**43.3**

Results for the MAVD dataset are summarised in Table 6. Two signal setups are evaluated to show the effect of a narrow band signal (sampling rate 16 kHz) in comparison to a full-band signal (sampling rate 44.1 kHz). The performance is measured with segment-based F1 and ER. The micro-averaged F1 is largely affected by the unbalanced nature of the data, and because of this class-wise F1 scores are reported together with macro-averaged F1-score. The results show that the proposed methods outperform the baseline system proposed in [49], especially when regarding macro-averaged F1-score. The relatively high error rate for the proposed methods is caused by the high insertion rate, approximately 85% of the errors are caused by insertions. This aspect of the system can be controlled by selecting an application-specific operation point. The systems reported in the table are not tuned for this aspect and the binarisation threshold is set to 0.5. For example, the performance of the system CNN10 (full-band signal) can be increased in micro-averaged ER from 1.03 to 0.60 and F1 from 57.4 to 62.7 by tuning the system to produce a lower amount of active events by setting the binarisation threshold into 0.85. Operation points for the SED systems will be selected based on the

Table 7: Sound event detection performance on GRN-SED dataset (4 sound event classes).

	Micro-avg		Class-wise F1				Macro-avg
Model	F1	ER	Bus	Car	Motorcycle	Truck	F1
System training with GRN-SED dataset							
CRNN	74.7	0.45	0	79	4	47	**32.5**
CRNN-desed	75.7	0.45	11	80	9	57	**39.3**
CNN6	77.0	0.43	7	80	22	62	**42.8**
CNN10	76.5	0.45	14	80	25	62	**45.3**
System training with GRN-SED, and GRN-SED-Mobile datasets							
CRNN	74.8	0.46	0	79	4	50	**33.3**
CRNN-desed	75.4	0.46	0	79	10	58	**36.8**
CNN6	76.9	0.43	11	80	24	63	**44.5**
CNN10	77.3	0.43	16	80	30	66	**48.0**

application specifications. The effect of the narrow-band signal can be regarded as limited based on the results. The results show good overall performance across evaluated systems. *Motorcycle* event class shows low performance across the systems, and this is due to the relatively low amount of learning examples available in the MAVD dataset for such a diverse sound event class. These results complete KPI-O2-E3-1 (increase the average accuracy for audio-visual event detection by at least 10%) by obtaining a 76% increase (in macro-averaged F1-score).

Results for the GRN-SED application-specific dataset are summarised in Table 7. Two training data setups are evaluated to show the effect of training data size and diversity. The results show again good overall performance across evaluated systems. Two sound event classes with a low amount of learning examples, *bus* and *motorcycle*, show limited performance. When adding the GRN-SED-Mobile dataset as learning examples, the performance of the motorcycle sound event class is slightly increased.

2.3 Self-attention on learned features for sound event localisation and detection

2.3.1 Introduction and objectives

The aim of *sound event localisation and detection* (SELD) is a spatiotemporal analysis of acoustic scenes while providing temporal activity information of sound events along with their spatial directions/locations (elevation and azimuth) while they are active [56, 57]. Multichannel audio captured with a microphone array is required as input in order to implement sound localisation. Overview of sound event localisation and detection is shown in Figure 5. Sound event detection provides information about the acoustic environment, and the spatial locations of events bring more detailed information about the sound sources that can be valuable for many applications. SELD can be utilised in many machine listening tasks such as tracking sound sources of interest [58], and audio surveillance [59]. The SELD joins to well-established research problems in acoustical signal processing, i.e., sound event detection (SED) and sound source localisation (SSL). Formulating and addressing these problems as joined SELD problem

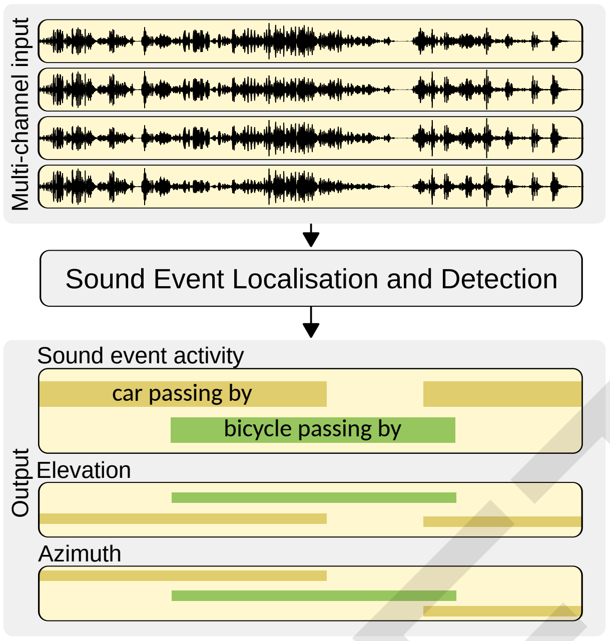


Figure 5: Overview of sound event localisation and detection. Multi-channel input is processed in the sound event localisation and detection system into sound event activity and localisation information (elevation and azimuth).

enables new possibilities in machine listening. Methods discussed in this section directly contribute to the SELD component in the MARVEL project, and the methods are focusing on approaches suitable for smart cities and real-time usage. Here we describe a summary of work on making neural network architectures used in SELD more suitable for faster inference. The corresponding paper is listed below, and can be found in Appendix 7.3:

- [60] P. A. Sudarsanam, A. Politis and K. Drossos, "Assessment of Self-Attention on Learned Features For Sound Event Localization and Detection", Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), 2021

2.3.2 Summary of the state-of-the-art

The state-of-the-art works on sound event localisation and detection are based on deep learning and CRNN-based neural network architectures. SELDnet [56] proposed for the task is based on a CRNN architecture where convolutional layers are used as feature extractors for multichannel spectrograms, and recurrent layers have the role of learning longer temporal dependencies in the sequence of features extracted by the convolutional layers. In comparison to the CRNN structure used for SED [37], an additional localisation inference output branch is introduced to predict the frame-wise direction of arrival (DOA) of each detected sound event class using a regression. Even though many alternative architectures such as ResNet [61], TrellisNet [62], and R3Dnet [63] have been proposed for SELD, the CRNN was the most popular method for the problem in DCASE2019 and DCASE2020 challenges. Many latest works have been concentrating on improved input features [64], a fusion of SED and SSL tasks [64, 65], and improved SELD representations and loss functions [66, 63].

The Transformer [67] architecture uses self-attention (SA) layers to model longer temporal or spatial dependencies similarly to recurrent layers used in CRNN architecture. Self-attention layers can be efficiently parallelised and thus making inference usage of the model significantly faster than one using recurrent layers. Transformers have been proposed for SED [68], however, their usage for SSL and SELD has been limited. In [69], self-attention was integrated into outputs of recurrent layers of CRNN-based architecture in a sound source localisation system for increased performance. In [70] this work was extended by replacing recurrent layers with transformers layers for further performance increase. For SELD, work presented in [66] included SELDnet-like architecture augmented with self-attention layers.

2.3.3 Description of work performed so far

The work investigates the effect of self-attention in a SELD task. The baseline SELD method used in the study is based on learnable feature extraction and learnable temporal pattern identification [56]. The proposed method replaces the temporal pattern identification with a self-attention mechanism. The baseline system has three convolutional neural network (CNN) blocks to learn high-level representations that are used as input to two recurrent neural network blocks (RNN). The output of RNN blocks is fed to a fully connected layer to combine the learned temporal relationships and the regressor layer is used to predict the detection and direction of arrival information at analysis time resolution. The system output is ACCDOA representation, where the sound event detection probability score is represented as the magnitude of the predicted localisation vector. The system is trained in a supervised manner with data having annotations for event activity and sound source location in relation to the capturing microphone array.

The work studied the effect of replacing the RNN block in the baseline system with self-attention blocks while keeping the rest of the architecture the same. The effect of the number of self-attention block, number of attention heads in each self-attention block, positional embeddings for each time step, and the effect of layer normalisation was studied systematically.

2.3.4 Performance evaluation

The systems were trained and evaluated using the development dataset published for the SELD challenge task in DCASE2021 Challenge [71]. The dataset contains 600 one-minute recordings with active sound events from 12 classes. In this work, the multi-channel audio is utilised in a 4-channel first-order ambisonics format with a 24kHz sampling rate. The data was split into 6 folds, each fold having 100 recordings, and four folds are used for training and single folds for validation and evaluation.

The sound event detection performance is measured with F1-score (F_{20}) and error rate (ER_{20}) metrics calculated location-dependent manner using a spatial threshold for true positives as defined in [57]. The true positives are defined to occur when the event is detected correctly and localised within 20° the ground truth. The localisation performance is measured in a class-dependent manner with localisation error (LE_{CD}) and localisation recall (LR_{CD}).

The main results are summarised in Table 8. The evaluated system was trained ten times and average metric values along with standard deviation were reported across these. Eight attention heads produce the best performance, together with positional embeddings for each time step and layer normalisation. Table 8 shows three topologies: system 1 is using two 128 dimensional self-attention, system 2 is using three 128 dimensional self-attention, and system 3 is using a configuration with three self-attentions with dimensions 128-256-128. The proposed systems outperformed the baseline with all metrics. The proposed systems have almost double the number of parameters compared to the baseline. However, the proposed systems were benchmarked to be 2.5 times faster than the baseline system during the inference due to parallelisation achieved with the self-attention blocks. These results complete KPI-O2-E3-3 (decrease the time needed to identify an event by at least 30% of the current time) by decreasing the inference time by 40%.

Table 8: Sound event localisation and detection performance on DCASE2021 SELD task's development dataset.

Model	Params	ER_{20}	F_{20}	LE_{CD}	LR_{CD}
Baseline	0.5M	0.69	33.9	24.1	43.9
Proposed 1	1.1M	0.61±0.01	45.84±1.06	21.51±0.74	54.99±1.87
Proposed 2	1.6M	0.62±0.01	44.63±1.14	21.56±0.46	54.46±0.94
Proposed 3	2.2M	0.62±0.01	45.14±1.03	21.67±0.41	55.29±1.23

2.4 Novel architectures and continual learning for automated audio captioning

2.4.1 Introduction and objectives

Automated audio captioning (AAC) aims at providing textual descriptions for a segment of audio. The AAC system will provide a textual description in form of sentences to describe what is happening in the audio, for example, "People talking while music playing in the background and cars passing by". The length of a given audio segment can be, e.g. 30 seconds or 1 minute. The audio captioning system can be used to collaborate and enhance the predictive behaviour of other systems by providing high-level information given the audio signals. The descriptions can be used as extra and indicative information to assist the decision-making process by the corresponding MARVEL components. The development of audio captioning requires an audio captioning dataset, which consists of audio signals having captions as annotations. In the learning stage, the system learns to map the input audio signals to the corresponding captions. In the prediction stage, the system takes as an input the audio signals and produces the textual descriptions that describe the content. An illustration of an AAC system is shown in Figure 6.

Methods discussed in this section are contributing to the AAC component in the MARVEL project. Here we provide a summary of work on novel architectures for AAC and on continual learning for AAC using the learning without forgetting approach. The corresponding papers are listed below, and can be found in Appendix 7.4 and Appendix 7.5:

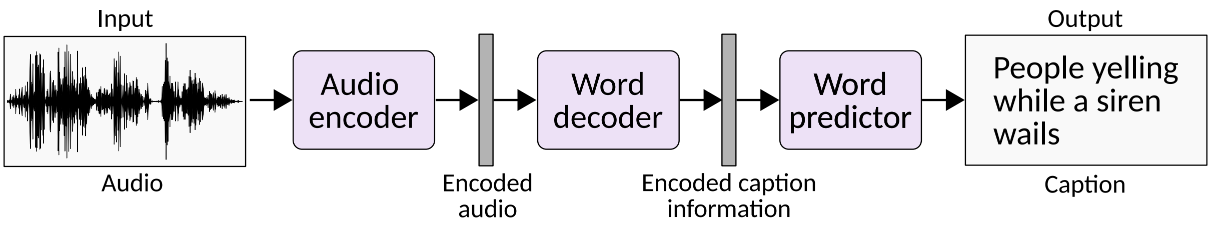


Figure 6: Overview of automated audio captioning. Input audio is processed with an audio encoder, and encoded audio is fed to the word decoder to get encoded caption information. Word predictor is applied to get a textual caption at the output.

- [72] An. Tran, K. Drossos and T. Virtanen, “WaveTransformer: An Architecture for Audio Captioning Based on Learning Temporal and Time-Frequency Information”, European Signal Processing Conference (EUSIPCO), 2021
- [73] J. Berg and K. Drossos, “Continual Learning for Automated Audio Captioning Using the Learning without Forgetting Approach”, Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), 2021

2.4.2 Summary of the state-of-the-art

The state-of-the-art for automated audio captioning systems utilise deep learning approaches with extremely large neural networks (over 100M parameters). Various approaches have been proposed for AAC in the literature. Input audio encoding has been implemented with recurrent neural networks [74, 75, 76], 2D convolutional neural networks [77], or Transformer model [67, 78]. The caption generation has been mostly implemented with Transformer decoder [77, 78] or recurrent neural networks [79, 74, 75]. The alignment of input audio and output captions is implemented with an attention mechanism [80].

2.4.3 WaveTransformer

The method proposed in [72] is using two different audio encoders for representing audio in an audio captioning task. One of the encoders uses 1D-CNNs that model temporal context, whereas the other encoder uses 2D-CNNs that model spectro-temporal context. The output of these two encoders is merged by learnable neural network layers. The proposed methods are applicable in MARVEL tasks that require audio captioning, especially those where it will be beneficial to model both temporal and spectro-temporal contexts.

Performance evaluation: Audio captioning performance is generally evaluated with a set of metrics originally proposed for image captioning (CIDEr, SPICE, and SPIDEr), and for machine translation (BLEU$_1$ to BLEU$_4$ scores, METEOR, and ROUGE$_L$). CIDEr calculates a weighted cosine similarity of n-grams while using term-frequency inverse-document-frequency (TF-IDF) weighting [81]. SPICE evaluates how well the predicted caption recovers objects, attributes, and their relationships [82]. SPIDEr takes advantage of both CIDEr and SPICE by calculating an average of these two metrics [83]. The metrics originally proposed for image captioning correlates better to the human ratings of caption quality, and thus only these metrics are reported below.

Clotho is a state-of-the-art dataset for captioning research [76]. It consists of 5,000 audio examples (15 to 30 seconds long), each example is manually annotated by humans with five captions (8-20 words). The total amount of annotations in the dataset is 25,000.

The main results are summarised in Table 9. Since the proposed method is not using multi-task learning or data augmentation, the results for the baseline systems [77] and [84] are reported without using them. The proposed method outperformed the baselines and showed the benefits of learning time-frequency information in the AAC task.

Table 9: WaveTransformer results for Clotho dataset.

Model	CIDE_r	SPICE	SPICE_r
Baseline 1 [77]	23.3	9.1	16.2
Baseline 2 [84]	23.2	8.5	15.8
WaveTransformer, learning temporal information	19.8	8.7	14.2
WaveTransformer, learning time-frequency information	24.7	9.3	17.0
WaveTransformer, using greedy decoding	24.7	**9.9**	17.3
WaveTransformer, using beam searching	**26.8**	9.5	**18.2**

2.4.4 Continual Learning

The method proposed in [73] investigates novel audio captioning methods where an audio captioning model trained using existing data is further trained using new data that becomes available for training. Such a scenario is typical in many machine learning applications including those studied in MARVEL, where a model is first trained and deployed, and then there is the possibility to further optimise the model once new data becomes available.

The work uses pre-trained AAC model [72] trained with Clotho dataset [76] while introducing data from another audio captioning dataset, AudioCaps [85]. Work proposes the use of continual learning without forgetting approach [86] and knowledge distillation [87] in an iterative process to learn the new information from a stream of new audio data. The aim is to learn new information about degrading the system's performance on the original data.

Performance evaluation: The main results are summarised in Table 10. Results show that the proposed method retains the system's performance on the original AAC dataset while at the same time manages to distill information from the new AAC dataset.

2.5 Learning from unannotated data with active learning and domain adaptation

2.5.1 Introduction and objectives

The purpose of the work was to develop a speech emotion recognition (SER) system to analyse initially unannotated recordings from a hospital environment while focusing on

Table 10: Continual learning results with varying value α to control the contribution of new information at each learning step.

	Testing data	
	Original data	New data
Setup	SPICE$_r$	SPICE$_r$
Training with original data	0.182	0.108
Training with new data	0.318	0.102
Training with original and fine-tuning with new data	0.065	0.247
Continual learning setup 1 ($\alpha = 0.7$)	0.109	0.211
Continual learning setup 2 ($\alpha = 0.8$)	0.186	0.157
Continual learning setup 3 ($\alpha = 0.9$)	0.182	0.153
Continual learning setup 4 ($\alpha = 1.0$)	0.176	0.115

babies' auditory environments. The aim of the SER system is to analyse the emotional content of speech in these environments. Methods discussed in this section are not directly contributing to any component in the MARVEL project, however, the learning from a large set of unlabelled data is very much related to the development of AI components in the project.

The paper providing details about the work on automatic analysis of unannotated audio material from daylong child-centered recordings to produce a speech emotion recognition system is listed below, and can be found in Appendix 7.6:

- [88] E. Vaaras, S. Ahlqvist-Björkroth, K. Drossos and O. Räsänen, "Automatic analysis of the emotional content of speech in daylong child-centered recordings from a neonatal intensive care unit", Annual Conference of the International Speech Communication Association (INTERSPEECH), 2021

2.5.2 Summary of the state of the art

The cross-corpus generalisation (CCG) is the most straightforward method to deploy SER for unlabeled datasets. However, this approach can suffer from domain mismatch. In [89], CCG was shown to be feasible only with certain datasets and emotional classes highlighting the problems with cross-domain SER model generalisation to out-of-domain data. Domain adaptation methods have been proposed for SER to tackle the domain mismatch [90, 91, 92, 93, 94]. In [90], an unsupervised deep denoising autoencoder (AE) was combined with supervised learning objective to create a semi-supervised domain adaptation method for SER. In [91], an unsupervised deep neural network-based adversarial domain adaptation was proposed to learn domain-invariant feature representations between labeled source data and unlabeled target-domain data and to maintain good performance on the SER task. Active learning-based methods have been proposed for SER tasks. In [95], an iterative learning algorithm was proposed to utilise conditional random fields to determine the level of uncertainty of each unlabeled data sample, and the most uncertain ones were selected for manual annotation. There is a limited amount of studies related to SER on large-scale dataset [96, 97]. Work in [97] showed that existing state-of-the-art models are prone to overfit to small-scale datasets limiting their abilities to generalise for real-life data.

2.5.3 Description of work performed so far

The work utilises an active medoid-based learning approach [98] to iteratively query human annotations for data points by starting the largest cluster. The Wasserstein distance-based domain adaptation [99] is used to adapt the source model to a target corpus by using labelled data from the source domain corpus.

2.5.4 Performance evaluation

Work included experiments on active learning, cross-corpus generalisation, and domain adaptation. In the active learning experiment, the aim was to study the performance of the support vector machine (SVM) model learned for SER through an active learning approach while using unlabeled material from the NICU-A dataset. In the cross-corpus generalisation experiment, the aim was to study 1-to-1 and 4-to-1 approaches where one or four source corpora were used for SVM training. In domain adaptation experiments, 1-to-1 and 4-to-1 adaptation conditions were examined. Active learning was observed to be the most consistent performer across the studied conditions. Detailed results can be found in [88] and in Appendix 7.6.

2.6 Contextual and structural game elements in citizen science projects applied on smart cities infrastructure

2.6.1 Introduction and objectives

The purpose of the work was to study how engagement in Citizen Science projects applied to Smart Cities infrastructure can be enhanced via contextual and structural game elements realised through augmented audio interactive mechanisms. An interdisciplinary framework was introduced based on the paradigm of a collaborative bird call recognition game. In this game, users collect and submit audio recordings, and these recordings are then classified and used for augmenting physical space. The work discussed in this section is not directly contributing to any component of the MARVEL project. The work deals with Smart Cities infrastructure and proposes a way to integrate Citizen Science projects into this. This addresses one of the major problems in AI development: how to collect and annotate learning examples. The paper that provides details about the work is listed below and can be found in Appendix 7.7:

- [100] E. Rovithis, N. Moustakas, K. Vogklis, K. Drossos and A. Floros, "Design Recommendations for a Collaborative Game of Bird Call Recognition Based on Internet of Sound Practices", Journal of the Audio Engineering Society, vol. 69, no. 12, pp. 956-966, 2021

2.6.2 Summary of the state of the art

The concept of Citizen Science is used to describe the projects where volunteers contribute to a scientific work by gathering and managing information [101]. In these projects, volunteers can have a variety of roles, including providing information to authorities and participating in making and implementing decisions [102]. Smart Cities can host large-scale Citizen Science projects, where Smart Cities' technological infrastructure is used to host Citizen Science targeting the citizens' well-being. Design principles for the work are playful learning, the internet of audio things, and bird monitoring.

The playful learning concept incorporates game elements into non-game learning environments [103]. Internet of audio things is an emerging field where computer devices embedded in physical objects are used for reception, processing, and transmission of audio information [104]. The bird monitoring projects usually target bird inventory, monitoring, and research, and one example of such project is the collaborative eBird project [105].

2.6.3 Description of work performed so far

The work suggests that motivation and understanding can be enhanced through an interdisciplinary approach that combines structural and contextual game elements with the Internet of Audio Things technologies [106] when implementing Citizen Science projects in Smart Cities environments. The work includes recommendations for developing an appropriate design framework for an augmented reality audio-based game through the paradigm of bird call recognition. The scenario consists of the following stages: a collection of vocalisations of birds and matching them to the corresponding bird species, submitting bird sound entries along with meta information to the server to create a 2D map representing bird presence in the urban environment, augmented version of the map is created through data sonification and sound spatialisation techniques. In the final stage, the bird sounds are presented in a location-specific virtual soundscape for better user engagement and collaboration. This paradigm is enhanced through game elements and citizen scientists become players of a large-scale participatory game where a level advancement, badges and rewards, and collaboration play an important role. The work suggests the following game scenarios: a quiz to recognise birds based on sounds, a treasure-hunting where specific bird species are located with limited time, time travel where players follow past observation locations to study migratory mobility, a bird adoption where players monitor the activity of specific bird for days and information about the bird species are relieved gradually.

A more detailed description of the design architecture of the framework and technical specifications can be found in [100] and in Appendix 7.7.

3 Methodologies for visual data analysis

In this Section, methodologies developed by MARVEL partners targeting visual data analysis are described. These include methodologies for Visual Crowd Counting (Section 3.1), Visual Anomaly Detection (Sections 3.2 and 3.3), and vision-based social distance estimation (Section 3.4). Moreover, the description includes an efficient visual data analysis method based on dynamic inference (Section 3.5), and two methodologies for improving performance of deep learning models for visual data analysis (Sections 3.6 and 3.7).

3.1 Efficient Visual Crowd Counting with Multi-Exit Vision Transformers

3.1.1 Introduction and objectives

Crowd counting is the problem of identifying the total number of people present in a given image. The applications of crowd counting include monitoring the safety of gatherings, managing natural disasters, improving the design of public spaces, gathering and analysing intelligence, creating virtual environments based on data from the real world, forensic search, and many others [107]. The input to a crowd counting model is a color image, and the expected output is a number representing the total number of people present in that image. Optionally, the output of a crowd counting model can be a density map which specifies the density of the crowd for each pixel of the input image, and the total count can be calculated by summing up all the density values for all pixels. Figure 7 provides an example of a crowded scene and its corresponding density map. Methods discussed in this section directly contribute to the VCC component in the MARVEL project, and the methods are focusing on efficient approaches suitable for real-time usage.

Here we describe seven different architectures for early exit branches placed on crowd counting models, that can lead to higher performance. A summary of this work is provided hereafter. The corresponding paper is listed below, and can be found in Appendix 7.8:

- [108] A. Bakhtiarnia, Q. Zhang, and A. Iosifidis, “Multi-exit vision transformer for dynamic inference”, British Machine Vision Conference (BMVC), 2021

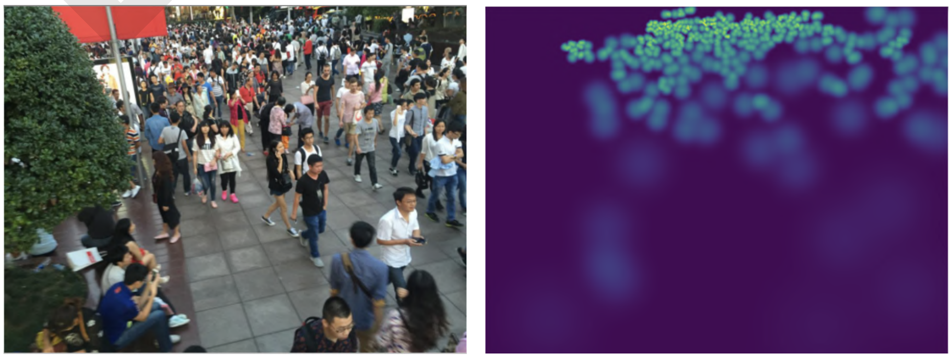


Figure 7: An example image (from the Shanghai Tech crowd counting dataset) and its corresponding ground truth density map.

3.1.2 Summary of the state-of-the-art

Most crowd counting models consist of many layers of interconnected neurons, which makes them computationally expensive and slow. This issue is exacerbated by the fact that the input images given to crowd counting models have a high resolution, typically HD or higher. This means that existing state-of-the-art crowd counting models cannot be used for inference at the edge out-of-the-box for every single input image. One approach for dealing with this problem is *dynamic inference* where the computation graph of the neural network is modified during execution in order to fit time and resource constraints that vary in an edge computing environment based on the state of the communication channels and the workload of edge servers.

Early exiting (also known as *multi-exit architectures*) is one method for dynamic inference where early exit branches are added after intermediate layers of a deep neural network in order to provide faster but inevitably less accurate results. Figure 8 illustrates a multi-exit architecture.

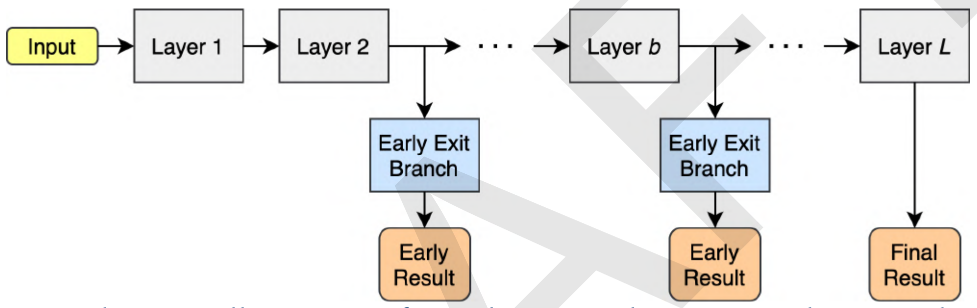


Figure 8: Schematic illustration of a multi-exit architecture with two early exits.

Vision Transformer (ViT) [109] is a recently proposed architecture for image classification which can be used as an alternative for convolutional neural networks (CNN) for other computer vision problems as well, including crowd counting. For instance, TransCrowd [110] is a ViT-based model for crowd counting. The Vision Transformer architecture is depicted in Fig. 9. Vision Transformers are massive models that are computationally expensive, however, due to their novelty with respect to CNNs, there has been limited work on dynamic inference in Vision Transformers, which limits their application in edge computing environments.

3.1.3 Description of work performed so far

Our method titled *Multi-Exit Vision Transformer* [108] includes 7 different architectures for early exit branches placed on Vision Transformer backbones. The proposed architectures include:

- *MLP-EE*: The baseline approach, consists of a handful of dense layers. The architecture of MLP-EE is shown in Fig. 10.

- *CNN-Ignore-EE, CNN-Add-EE and CNN-Project-EE*: These architectures reshape the representations in the ViT backbone based on the locality and neighborhood of the corresponding image patches, and process them using a convolutional filter. The three different variants handle the classification token of the ViT backbone in different ways. The architectures of CNN-based early exits are shown in Fig. 11.

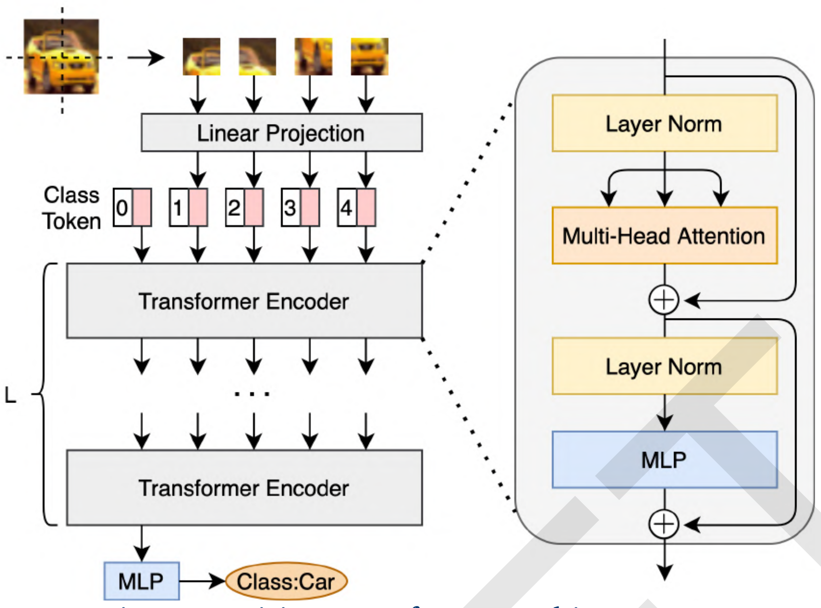


Figure 9: Vision Transformer architecture.

- *ViT-EE*: This architecture uses the building blocks of the ViT backbone (i.e. Transformer encoder) itself. The architecture of ViT-EE is shown in Fig. 12.
- *MLP-Mixer-EE and ResMLP-EE*: Several attention-free MLP-based architectures have been proposed in the literature as an alternative for Vision Transformer that preserve their advantage in terms of the global receptive field, yet are more lightweight since they do not contain the self-attention mechanism present in ViTs. We use the building blocks of two such methods, namely MLP-Mixer [111] and ResMLP [112], in order to obtain lightweight high-performance early exits. The architectures of these attention-free MLP-based early exits are shown in Fig. 13.

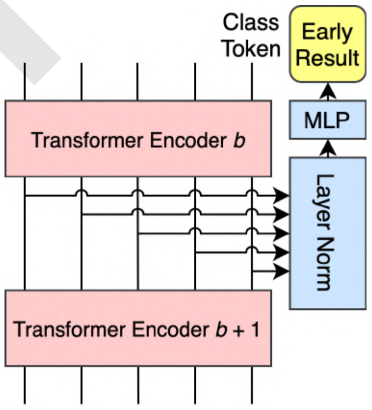


Figure 10: MLP-EE early exit branch architecture.

3.1.4 Performance evaluation

We investigate which architecture is optimal in which scenarios. Our experiments use TransCrowd [110] as the backbone. We use DISCO as the crowd counting dataset, which contains 1935 Full HD images. DISCO is an audiovisual dataset, meaning that

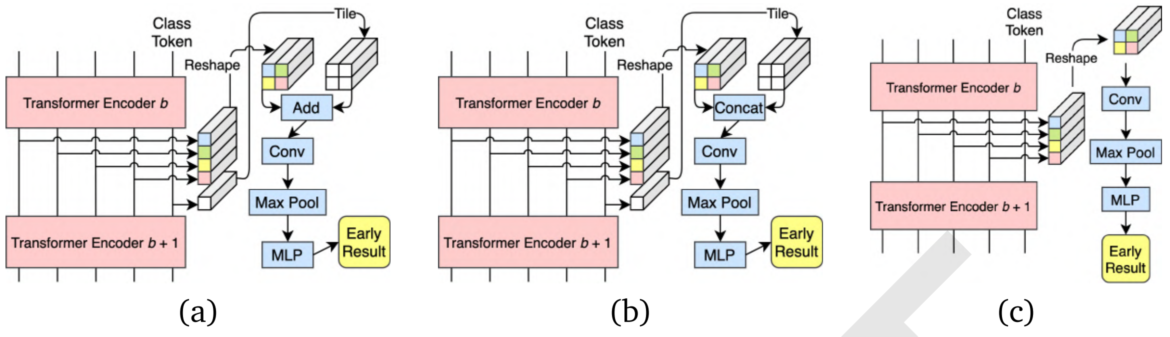


Figure 11: (a) CNN-Add-EE; (b) CNN-Project-EE and (c) CNN-Ignore-EE early exit branch architectures.

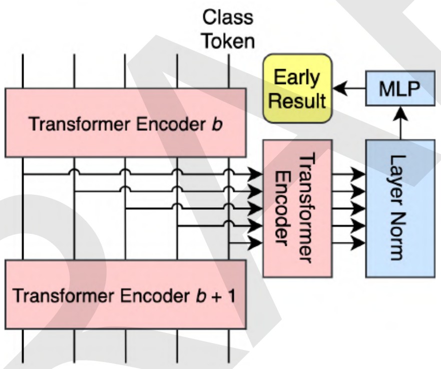


Figure 12: ViT-EE early exit branch architecture.

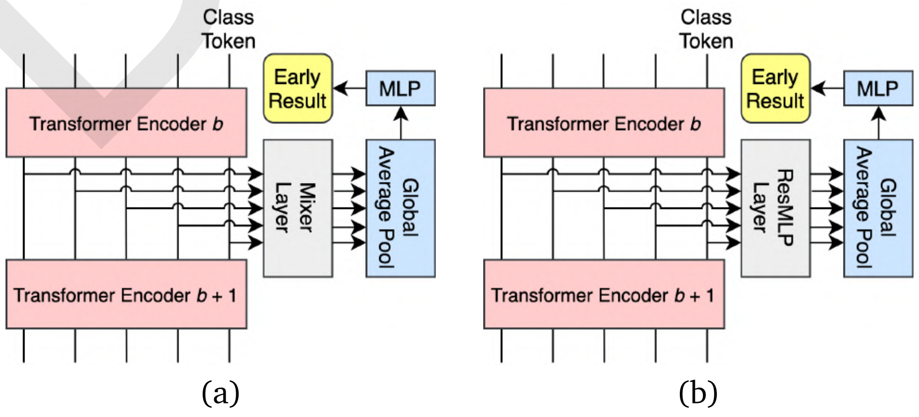


Figure 13: (a) MLP-Mixer-EE and (b) ResMLP-EE early exit branch architectures.

it contains audio clips corresponding to each image. However, we discard the audio clips for these experiments. Our results are summarised in Table 11. The performance of early exit branches is measured using mean absolute error (MAE) where lower is better.

Table 11: Multi-Exit Vision Transformer: Performance on the DISCO dataset, floating point operations (FLOPS), parameter count, and maximum allocated memory at inference.

Model	MAE	Improvement
MLP-EE at 3rd layer (Baseline)	13.77	-
MLP-Mixer-EE at 3rd layer (Ours)	**12.24**	11.11%
MLP-EE at 6th layer (Baseline)	11.54	-
ViT-EE at 6th layer (Ours)	**11.2**	2.95%
MLP-EE at final layer (Baseline)	11.44	-
ResMLP-EE at final layer (Ours)	**11.09**	3.06%

3.2 Unsupervised Visual Anomaly Detection with Pruned Memory-Augmented Deep Autoencoder

3.2.1 Introduction and objectives

Visual Anomaly Detection can be defined as the task of identifying novel situations in a scene based on the visual information captured in an input video or image. Models are trained to learn and replicate a situation considered normal in a certain setting, e.g., pedestrians on a pathway. Situations that do not occur often, for example, a car on the pathway, are not present in the data used to train the model to identify as normal and, thus, they will be detected as anomalies. One of the key challenges posed for the Anomaly Detection use cases in the MARVEL project is their ability to operate efficiently on the edge, fog, and cloud. The lower computing capabilities of the edge and fog devices necessitate the development and adjustment of efficient methods to enable real-time operation on restricted computing resource devices while maintaining performance levels as close as possible to the original and more computationally complex models. Methods discussed in this section directly contribute to ViAD component in the MARVEL project, and the methods are focusing on efficient approaches suitable for real-time usage.

3.2.2 Summary of the state-of-the-art

The problem of Visual Anomaly Detection is well studied in the literature. In unsupervised learning settings, the most common approach, and one that reliably performs the best, is to employ autoencoders followed by the application of thresholding to the attained regularity scores. Multiple notable works exist that follow this direction. One of those that effectively combine accuracy with efficiency, required in MARVEL for real-time operation, is Memory-augmented Deep Autoencoder for Unsupervised Anomaly Detection (MemAE) [113]. The MemAE method receives as input a sequence of video frames. Input video frame sequences can vary in length, depending on the required trade-off related to the method's properties, like processing speed, detection accuracy, and optimal hyperparameters. The output of the method is a list of regularity scores for all frames in the sequence.

The CNN-based Autoencoder (AE) is a model capable of reproducing high-dimensional data, such as images or videos, in an unsupervised setting. The encoder obtains a compressed description of the input, and the decoder learns to reconstruct the input from that compressed form. The encoding thus forms an information bottleneck, which incentives the network to learn the typical patterns in the presented input. However, along with the advances in the computational power of modern GPUs, the CNN networks can learn to reconstruct normal frames even when presented with an anomalous input. This violates the assumption that reconstructing an anomalous frame would result in a higher reconstruction error, which in turn results in less obvious anomalies, or those highly resembling normal situations, slipping past the anomaly detector.

To alleviate that problem, MemAE uses a version of AE augmented with a memory module. The idea behind this choice can be seen in Fig. 14. After training the neural network on normal video frame sequences, prototypical normal patterns are encoded in the feature space defined in the bottleneck layer of the AE architecture. Given an abnormal input, MemAE retrieves a relevant normal pattern in the memory for reconstruction. This leads to an output that is different from the network input. The introduction of memory modules that hold the prototypes of the normal images and then use them to guide the AE to reconstruct a frame that is closer to a prototype decreases the AE's ability to create a normal image from an anomalous input. This means that its sensitivity to anomalous frames is increased, which will lead to higher anomaly detection performance.

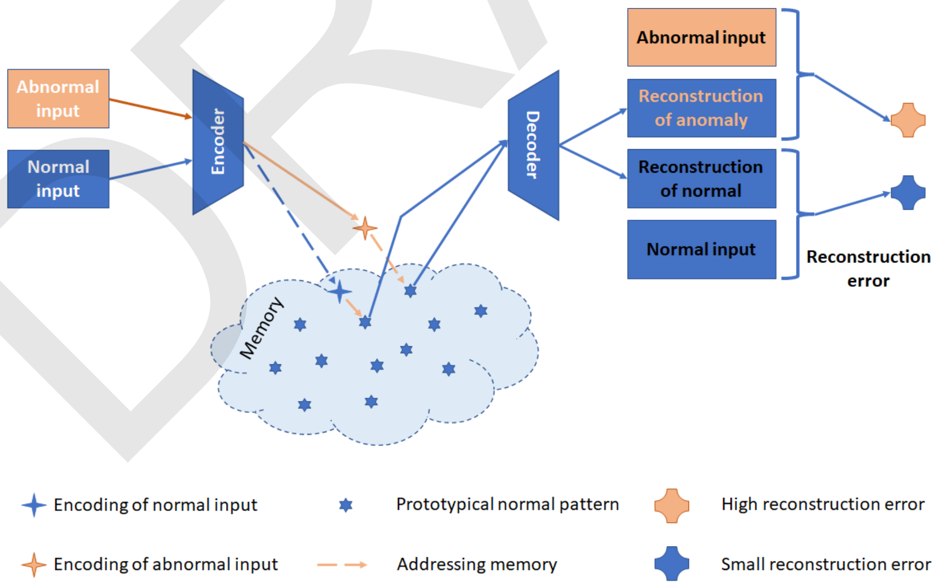


Figure 14: Anomaly detection via MemAE.

For testing purposes, the Area Under the Receiver Operating Characteristic (AUROC) is calculated. For deployed application, this is of course neither feasible nor meaningful. The model will produce the regularity scores for each frame as during the testing. But instead of calculating AUROC, these scores can then be thresholded to achieve binary anomaly labels for each frame. One method that can be used for this purpose is Persistence1D [114].

3.2.3 Description of work performed so far

A key aspect of the Anomaly Detection component for the MARVEL project is its applicability in real-time operation on the edge, fog, and cloud. To assure that capability on less powerful hardware than desktop or server-grade CPUs and GPUs available when conducting experiments, we have undertaken experiments with pruning the MemAE network to a varying degree, and then measuring the model's inference time, size and accuracy. It was shown [86] that in many deep networks a significant portion of parameters can be eliminated without adversely affecting the performance of the network. AE-based anomaly detectors for which the problem, as resolved by MemAE, is not the lack of the network's capacity, but rather its relatively too powerful generalisation capability, can be expected to be able to achieve substantial reduction in size while maintaining similar levels of performance.

The pruning method introduced in [86] prunes filters across multiple convolutional layers, i.e. when filters in a layer are removed, it takes into account the filters in the following layers that will be affected. Fig. 15 provides a graphic overview of that method. The user decides which layer of the network will be pruned, and the information about the removed layer filters is propagated through the network leading to the removal of filters in the remaining layers that are affected by the pruning process. This approach has the advantage of performing the retraining only once while still providing robust pruning capabilities. Then, the whole network can be retrained. This process can be applied for pruning multiple layers in a sequence. This is a suitable process for our case since the MemAE model is rather small and its training is relatively quick.

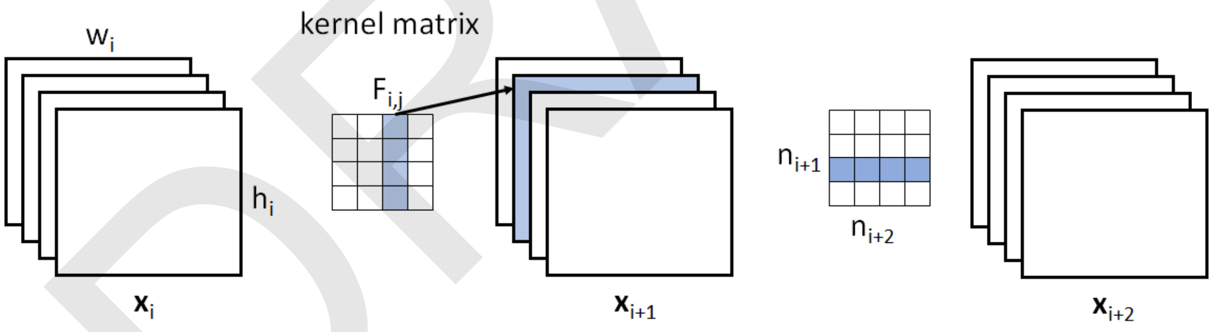


Figure 15: Pruning a filter results in removal of its corresponding feature map and related kernels in the next layer.

When it is applied to the l-th layer of the network, the pruning process applies the following processing steps:

1. For each filter calculate the sum of its absolute kernel weights.
2. Sort the filters based on the calculated weights.
3. Prune $\mathbf{m}$ filters with the smallest sum values and their corresponding feature maps. The kernels in the next convolutional layer corresponding to the pruned feature maps are also removed.
4. A new kernel matrix is created for both the l-th and $(l+1)$-th layers, and the remaining kernel weights are copied to the new model.

The MemAE network consists of four 3-dimensional convolutional layers, forming the encoder, and four 3-dimensional deconvolutional (performing transposed convolution) layers, forming the decoder. The model also contains batch normalisation layers, placed between the convolutional layers. During our experiments, we do not directly prune the first and last layers. For the six remaining layers, starting with the second to last 3-D deconvolutional transpose layer, we apply pruning in 5% increments. The percentage here means the fraction of the total nodes in a layer. After each pass, the network is retrained for 5 epochs. After the layer has been pruned up to 95% of its starting nodes, we proceed to prune the layer before it. But this time we start from the best performing pruned model from the last iteration.

3.2.4 Performance evaluation

Fig. 16 presents the per layer impact of pruning on the performance of the model, along with its impact on the model's FLOPs. For reading convenience, on each plot values visible for 0% of pruned units show the performance of the original, unpruned model. It can be seen that retraining for 5 epochs after each pruning is not enough to provide a resilient and reliable change in performance. However, even with this insufficient amount of retraining, the per layer results show that for most layers removing 40 to 60 % of nodes can still provide a high area under the curve (AUC).

Table 12 presents the experimental results in terms of the impact the pruning process has on the accuracy of the model and the resulting gains in FLOPs, number of parameters, and model size. The best model in terms of performance, reduction in size, and computing complexity, together with the increase in throughput for the optimally pruned model was pruned, layer by layer, by $45\%, 60\%, 10\%, 80\%, 45\%$ and 25%.

Table 12: MemAE: Performance on UCSD pedestrian 2 dataset, floating point operations (FLOPs), parameter count, and maximum allocated memory of method at inference.

Model	Performance (%)	FLOPs (G)	Parameters (M)	Memory (MB)
Original MemAE	97.8%	284.76	$6,492,801$	25.97
Pruned MemAE	99%	255.06	$1,833,368$	7.33

By comparing the original model with the pruned model determined by applying the above-described process, we observe the following:

- Number of parameters: $6,492,801 \rightarrow 1,833,368$, improvement of 71.76%
- Model size: 25.97MB $\rightarrow 7.33$MB, improvement of 71.78%
- Forward pass size: 991.76MB $\rightarrow 574.83$MB, improvement of 42.04%
- GFLOPs: $284.66 \rightarrow 225.06$, improvement of 21%
- AUC: $97.8\% \rightarrow 99\%$, improvement of 1.2%

The increase in the model's AUC can be explained by the relatively small size of the test set, and the experiment methodology. In effect, it amounts to iterative transfer

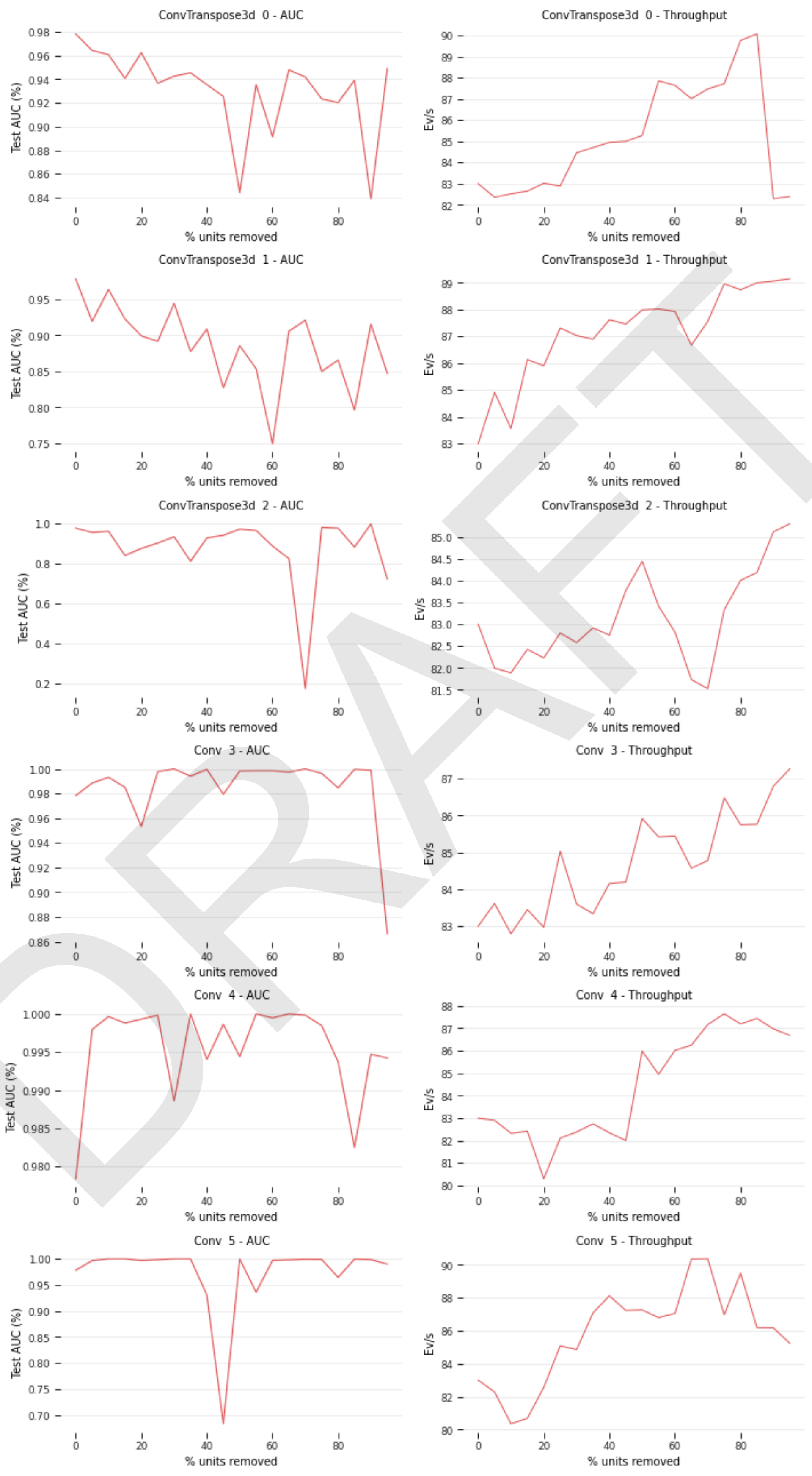


Figure 16: Impact of pruning on model's AUC and evaluations per second measured on RTX 2080 GPU.

learning. However, it shows that despite the reduction of the number of parameters by almost 75% and computing complexity by over 20%, the model is capable of producing retaining performance at levels similar to those of the original model. Furthermore, it is possible to prioritise different aspects of the model's characteristics. This experiment has been performed by prioritising the model's size reduction while maintaining the highest possible accuracy. But it is also possible to prioritise the decrease in computing complexity, or any other measurable metric.

3.2.5 Future work

As future work, we plan to evaluate the performance of the adopted structure parameter pruning methodology to other publicly available visual anomaly detection datasets, as well as data in the MARVEL Data Corpus.

3.3 Rule-based Visual Anomaly Detection based on object detection

3.3.1 Introduction and objectives

As an alternative solution for the Visual Anomaly Detection problem, a rule-based method that employs a pre-trained object detector and a binary scene mask created by users to indicate image locations where the rules are applied was developed, which is called Rule-based Visual Anomaly Detection (RviAD). The founding assumption behind this approach is that for some use cases in the MARVEL project, the problem of anomaly detection can be defined in the form "if object X is in area Y, this corresponds to an anomaly". This problem can be thus split into two subproblems, i.e., a) object detection, classification, and localisation of that object within a frame, and b) a rule check on whether that object is allowed to occupy that location in a frame. Thus, an objective for such a visual anomaly detection method is to create an interface that can allow the introduction of new rules in an easy and intuitive way. The method discussed in this section does not currently directly contribute to the MARVEL project, but it can potentially be used as an alternative to the approach described in Section 3.2.

3.3.2 Description of work performed so far

The first part of the method, i.e., object detection, is an exhaustive research field, with many state-of-the-art pre-trained models achieving reliable performance. This means that open source, pre-rained state-of-the-art object detectors can be used with only limited finetuning for each use case, and good accuracy can be expected. The second part, i.e., assigning rules to regions along with connecting those pieces into a working pipeline, was created by researchers in AU.

The main and significant advantage of that system is its flexibility and ability to dynamically adapt to changing situations, when new rules are added. The same model can be used to detect any combination of anomalous objects in regions. Using the developed binary scene mask creation tool, the object detector and rule-based setup implement many useful functionalities:

- Ability to apply several rules to the same video frame and detected objects. The rules can be prioritised, or have any possible interaction between them.

- Completely changing the nature of the anomaly, e.g., changing allowed objects on the roads from only cars to only motorbikes takes a few minutes and can be deployed instantly.

- Extending or temporarily changing rules can be applied in the same manner. For example, when for a certain period the road on which usually cyclists are not allowed is designated to host a cycling race, the anomaly detector can be adapted to that use case within minutes, or even scheduled to do so at a certain time and return to normal at a certain time.

- When the setup of the road under surveillance changes, only adjustment of the binary scene mask regions is required. For example, if a new lane is added to the road, the standard anomaly detection model will need to be retrained from scratch, while the rule-based anomaly detection model requires only a change of the binary scene mask. The difference in time and effort required to do that is substantial.

A disadvantage of this approach compared to the unsupervised anomaly detection approach described in Section 3.2 is that, if the object detector needs to be finetuned on data from the specific scene for improving its performance, it requires a time-consuming data labeling process as every object in each video frame needs be labeled and annotated with a bounding box. The amount of such labeled data that would be required to finetune the already pre-trained state-of-the-art object detection model to achieve the required performance is an open question.

The RViAD implementation uses Centernet [115] as its object detector, and a web browser-based tool for an-easy-to-use binary scene mask creation process. The tool along with an example created binary scene mask applied on the scene from the GRN3 use case (*Traffic Conditions and Anomalous Events*) are shown in Fig 17. This use case focuses on monitoring the traffic and detecting anomalous events in road junctions for assisting short term decision-making.

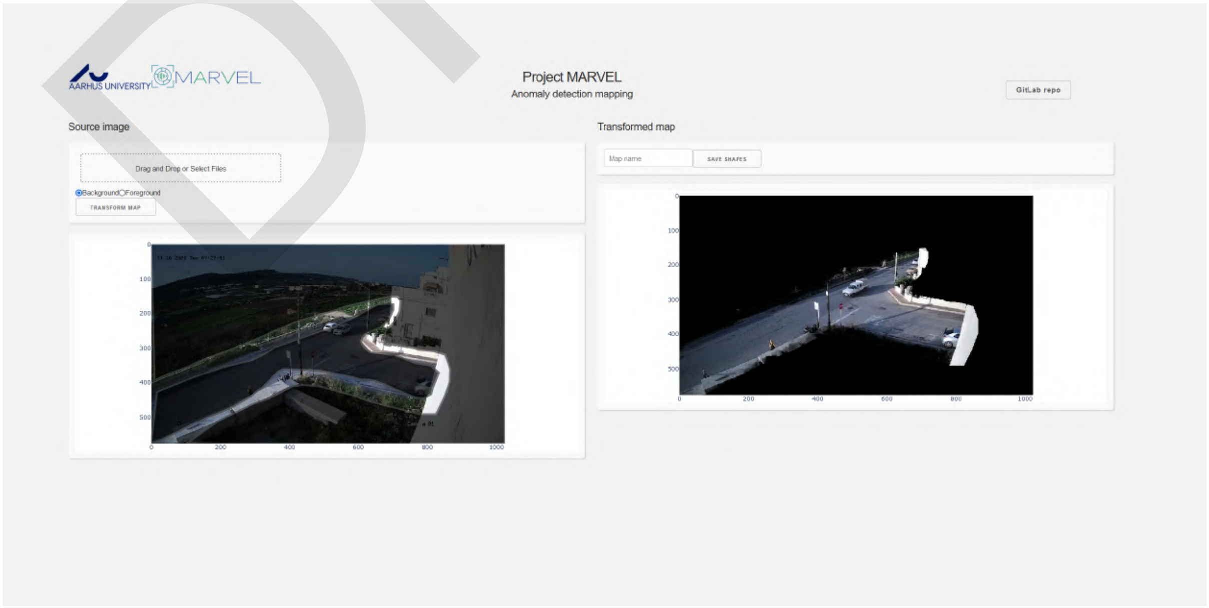


Figure 17: Web browser-based tool for creating visual anomaly detection rules and binary scene masks.

Fig 18 shows how the tool works underneath. From top left to bottom right, the figure presents:

- The original video frame.
- The raw binary scene masks created by the user-drawn shapes using the web tool. White regions represent the foreground regions, i.e., in that area appearance of the chosen object or objects is anomalous will be detected as an anomaly. Black regions represent the map background regions, i.e., in that area the chosen object or objects can appear without indicating an anomaly.
- The binary scene mask obtained after applying Alpha Matting [116, 117] in order to adjust the user-defined binary scene mask to the contents of the scene.
- The binary scene mask over the original frame. This mask can be described as a 'filter' for the object detector.

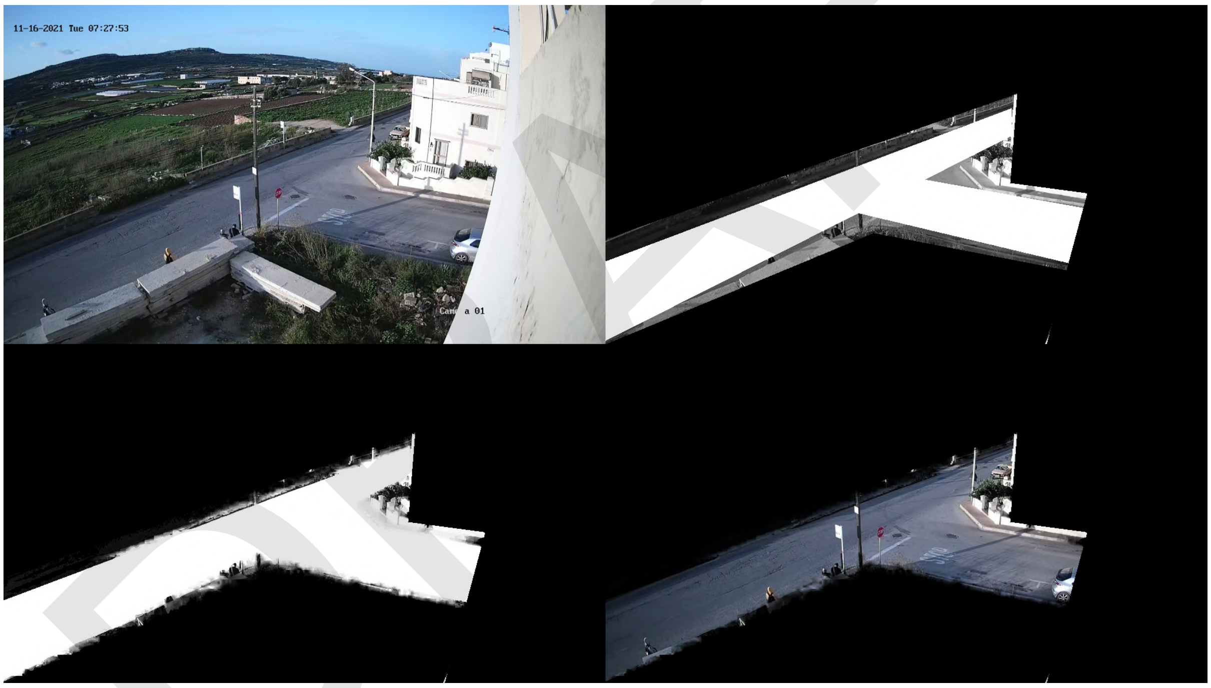


Figure 18: The processing sequence of a binary scene mask.

Finally, Fig 19 demonstrates a video frame from the GRN3 use case, with the binary scene mask and rules defined in Fig 18 applied to it. The rule assigned to that mask was *"appearance of people and cars in the pre-specified region corresponding to the road is an anomalous event"*. We can see that the single person at the bottom-left side of the video frame standing entirely on the disallowed region is marked as an anomaly, while the remaining people in the video frame are not. This is because their bounding box is also inside the background part of the mask. If that is not the expected behaviour, a simple mask region change in that region could include those people as anomalies. The car present at the centre of the video frame is correctly identified as an anomaly. The truck at the upper center-right is not identified as an anomaly, as the rule mask allows trucks to be present in that region. All detected objects outside the mask foreground have been discarded.

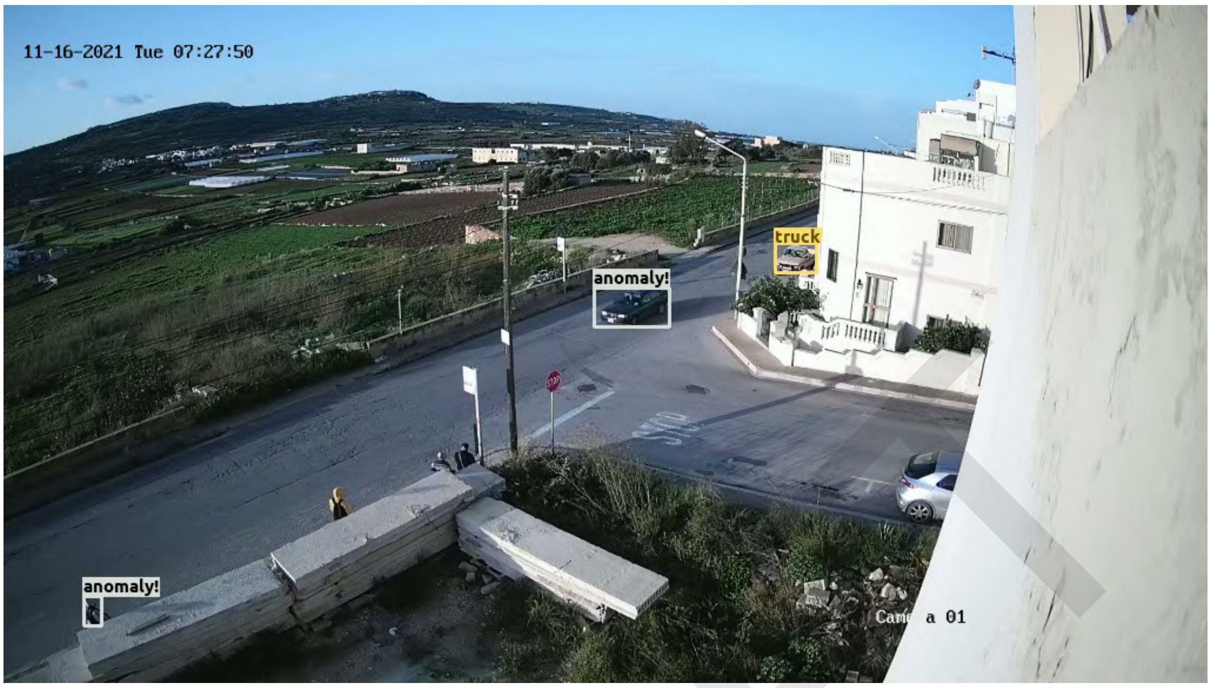


Figure 19: Example frame from the rules based anomaly detector.

3.3.3 Future work

As future work, we will evaluate the effectiveness of the above-described rule-based visual anomaly detection methodology in connection to the MARVEL use cases.

3.4 Social Distance Estimation in Uncalibrated Images based on Pose Estimation and Projective Geometry

3.4.1 Introduction and Objectives

Automatic social distance estimation from images and videos can find many applications in the context of smart cities. One of the recently emerged applications where such an automatic and non-invasive functionality can lead to improved services for citizens is through public recommendations for avoiding crowded places. Social distancing has recently received a lot of attention due to the outbreak of the SARS-CoV-2 virus [118], as it has been one of the measures used for slowing down the spread of the the virus. Moreover, automatic social distance estimation can be used for analysing the behavior of people in various public places. For such cases, it is possible to use fixed camera setup and location, and exploit 3D scene information by, for instance, using multiple camera setups or depth or thermal cameras. However, such setups will require careful calibration of the multi-camera system, or increase the operation cost. In addition, most of the existing surveillance systems are based on low-cost cameras which are used to capture areas having low overlap with the field of view of other cameras. While the use of social distance estimation methods based on uncalibrated single cameras can lead to lower performance compared to calibrated multi-camera setups, they can lead to a reliable solution for a wide range of applications. We proposed a social distance estimation method that can be applied to uncalibrated images taken by a regular camera as long as the focal length and the camera sensor size are known. To do so, we combine object detection and human pose estimation methods with projective geometry using the intrinsic parameters of the camera. The work deals with social interactions in Smart Cities and proposes a way to visually estimate social distances between people. This problem is related to the Visual Crowd Counting functionality in MARVEL.

A summary of this work is provided hereafter. The corresponding preprint is listed below, and can be found in Appendix 7.9:

- [119] M. Seker, A. Mannisto, A. Iosifidis and J. Raitoharju, "Automatic Social Distance Estimation From Images: Performance Evaluation, Test Benchmark, and Algorithm", 10.5281/zenodo.6737051, pp. 1-14, 2022.

3.4.2 Summary of the state-of-the-art

Most existing methods approach the automatic social distance estimation problem for preventing social distance regulation violations (motivated by the COVID-19 social distance estimation scenario). That is, they try to solve a binary classification problem where the goal is to classify the pair-wise distances between people either as safe or unsafe, depending on a given threshold distance value. Such methods employ deep learning based methods that detect and track people [120] or keypoints corresponding to the human body joints obtained by applying a pose estimation algorithm [121] and use some form of scene information to perform binary classification. For example, [121] requires manual input to estimate the homography matrix of the image plane to the ground plane. The homograph matrix is also used in [122] which is obtained through calibration and person detected bounding boxes obtained by applying deep learning-based object detectors [123, 124]. The intrinsic parameters of the camera are also used in [125] along with keypoints obtained by a pose estimation model to train a feedforward network. This is used to predict the 3D locations of people and detect distance violations. 2D keypoints obtained by a pose estimation method are also used in [126], where a regressor block is trained on public datasets containing both 2D and 3D visual information, in order to estimate pair-wise distances. The method can be applied to uncalibrated images. The method in [127] uses a person detection method based on the deep neural network YOLOv3 [128] to monitor social distancing violations from overhead view cameras.

3.4.3 Description of work performed so far

The method starts by performing person detection using the YOLOv4 [123] object detection model. Bounding boxes are cropped and they are introduced to OpenPose [129] human pose estimation model to extract human body skeleton keypoints. The pixel locations of these keypoints are used in our distance estimation algorithm to obtain 3D location estimates for each person in the image. We use both person detection and human body pose estimation to eliminate false positives which may be detected by each of these methods independently. From the extracted keypoints obtained by the human body pose estimation method, we select a subset of them based on the following criteria: a) mutual distance is independent of the person's pose, b) average distance is available in the literature, c) angle towards the lens is as constant as possible, and d) are visible in most of the photos. Based on these criteria, we selected the following three key point pairs: 15-16 (these are the indices of the keypoints based on the OpenPose output) for pupillary distance, 2-5 for shoulder width, and 1-8 for torso length. We assume average adult body proportions for the three keypoint pairs: 389 mm for shoulder width [130], 63 mm for pupillary distance [131], and 444 mm for torso length [132]. The extracted keypoint pairs are then processed by our distance estimation algorithm that estimates 3D positions with respect to the camera for each person.

We use the pinhole camera model [133] and assume that every keypoint pair is parallel to the camera's sensor plane. We make these assumptions because the subjects' poses and camera's exterior orientation parameters [134] are not known. Estimating the exterior orientation parameters [134] of the camera from single images is an ill-posed problem [135], but in most cases, the angle between a person's torso and the camera's sensor plane is negligible for our calculations. We use the intrinsic camera parameter values and the pixel locations of the selected keypoints to estimate the 3D coordinates of the keypoints in the world coordinate system. Then, the middle points of each detected keypoint pair are used to represent the 3D location of the person. Finally, the distances between all the pairs of detected people are calculated.

3.4.4 Performance Evaluation

We conducted experiments on the KORTE Social Distance Estimation dataset [119]. Table 13 shows the person detection rates and pair-wise percentual distance estimation errors for different focal lengths included in the dataset. The most reliable body part to estimate the 3D locations is the torso. When all pairs of keypoints corresponding to the three body parts are used, the estimated distance error is lower.

Table 13: Person detection rates and pair-wise percentual distance errors when using a single body part (shoulder, pupil, and torso) and when combining all of them.

Focal Length	Number of	Shoulder Based Method		Pupil Based Method		Torso Based Method		Combined Method	
		Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error
16	28	0.75	333.42	0.55	39.79	0.82	36.30	**0.89**	**28.80**
24	46	0.81	346.05	0.55	39.52	0.91	33.22	**0.94**	**24.68**
35	48	0.81	450.49	0.58	65.63	0.91	48.52	**0.92**	**34.68**
50	84	0.80	306.56	0.44	72.37	0.91	39.29	**0.94**	**35.03**
105	69	0.72	332.72	0.57	110.50	0.79	73.29	**0.89**	**52.50**
200	7	0.69	105.28	0.73	52.28	0.69	93.53	**0.78**	**53.66**
300	18	0.70	1244.59	0.60	52.88	0.61	148.94	**0.78**	**52.51**
All	300	0.78	385.22	0.54	68.56	0.84	51.01	**0.91**	**38.24**

3.5 Improving performance of Multi-Exit Deep Learning models for Dynamic Image Classification

3.5.1 Introduction and Objectives

As mentioned in Section 3.1, when using a deep neural network for performing image classification, incorporating early exit branches at intermediate neural network layers at different locations in the neural network topology leads to a dynamic inference model.

The early exit branches can be used to provide an early classification result following the just-in-time classification paradigm. This is an essential part of the Edge Intelligence framework, as the edge, fog, or cloud computing component may receive the image to be classified with a varying delay which can be caused by, e.g., delays in the communication channel. Exit branches placed early in the network are inevitably less accurate than the final exit. In Section 3.1, we focused on architectural designs for early exit branches that can improve their accuracy. However, accuracy can also be improved by a better training procedure. Here we describe two different approaches for training early exit branches that can lead to higher performance. The methods discussed in this section are not directly contributing to any component of the MARVEL project. The methods are contributing to the general development of visual representation learning and apply to the project from that point of view.

A summary of this work is provided hereafter. The corresponding papers/preprints are listed below, and can be found in Appendices 7.10 and 7.11:

- [2] A. Bakhtiaria, Q. Zhang and A. Iosifidis, "Improving the Accuracy of Early Exits in Multi-Exit Architectures via Curriculum Learning", International Joint Conference on Neural Networks, Virtual, 2021.
- [136] A. Bakhtiarnia, Q. Zhang and A. Iosifidis, "Single-layer vision transformers for more accurate early exits with less overhead", accepted to Neural Networks journal (10.5281/zenodo.6737408), 2022.

3.5.2 Summary of the state-of-the-art

Curriculum learning is a machine learning strategy that draws inspiration from the way humans learn new subjects throughout their formal education. For studying a new topic, a teacher determines which notions of the subject are simplest and easier for the students to grasp, then more difficult aspects are introduced with an increased level of difficulty during the learning process. Curriculum learning treats the problem of training a machine learning model in a similar manner, by using a dataset as the topic and introducing training samples progressively based on their difficulty. Several strategies exist for measuring the difficulty of the training samples. Automatic sorting functions include those belonging to the category of self-paced learning, transfer teacher, and reinforcement learning teacher [137].

There exist several theoretical analyses showing why curriculum learning can improve the training procedure of a neural network. The connection between curriculum learning and continuation methods was shown in [138]. Continuation methods are optimisation strategies for non-convex problems that start with a smooth objective and gradually introduce less smooth versions, hoping that this will lead to approaching a more suitable location in the parameter space by exploiting a more global perspective of the optimisation process [137]. The study in [139] reached the conclusion that curriculum learning modifies the optimisation landscape to highlight the optimal parameter vector in comparison to all other candidate solutions. While several methods for improving the performance of neural networks based on curriculum learning have been proposed [139, 137], the adoption of this learning paradigm for training multi-exit branches has not been studied. However, other strategies, such as knowledge distillation have recently been shown that can improve the accuracy of early exits.

In the copycat training strategy [140], a “fake” dataset is created by taking images from a different domain and given to a model trained on the target domain. The output of the model is then recorded as ground truth labels for that image. For instance, if images from the ImageNet dataset are given to a model trained on the CIFAR-10 dataset, a camel may be labelled as a dog, since there are no labels for camel in CIFAR-10. Finally, this fake dataset is mixed with the main dataset and used for training. The intuition behind this approach is that the fake data can increase the number of training examples without introducing too much noise and thus improve the accuracy of deep learning models which are data-hungry. Experimental results are presented in Table 14. From these results, it can be observed that in all cases, our approach (i.e. curriculum learning and anti-curriculum) obtains a higher accuracy compared to the baseline (i.e. applying a standard training process, referred to as vanilla).

Table 14: Comparison of the Final Test Accuracy of Early Exit Branches using Different Training Methods.

Backbone	Dataset	BN*	Vanilla	Curriculum	AC†	RC§	Opt.	LR¶	Teacher	Pacing
DenseNet	CIFAR-10	1	71.71% ± 0.47	71.59% ± 0.57	**71.75% ± 0.75**	71.58% ± 0.57	Adam	10^{-4}	EfficientNet	FEP(100)
		2	77.43% ± 0.64	**77.91% ± 0.03**	77.21% ± 0.80	77.25% ± 0.35	SGD	0.12	EfficientNet	FEP(100)
	CIFAR-100	1	38.36% ± 0.31	**39.56% ± 0.57**	35.32% ± 2.06	38.74% ± 1.37	SGD	0.12	EfficientNet	FEP(200)
		2	61.72% ± 1.26	**64.05% ± 1.18**	58.78% ± 1.68	62.95% ± 0.78	Adam	10^{-4}	EfficientNet	FEP(300)
MobileNet	CIFAR-10	1	67.30% ± 0.25	**67.33% ± 0.31**	67.04% ± 0.45	67.02% ± 0.48	Adam	10^{-4}	EfficientNet	SSP(300)
		2	79.06% ± 0.65	**79.47% ± 0.05**	79.04% ± 0.43	78.55% ± 0.41	Adam	10^{-4}	Inception	FEP(100)
	CIFAR-100	1	44.26% ± 0.69	44.83% ± 0.19	**44.89% ± 0.26**	44.84% ± 0.45	Adam	10^{-4}	Inception	FEP(300)
		2	47.48% ± 0.99	**48.39% ± 0.66**	47.54% ± 0.45	48.34% ± 0.74	Adam	10^{-4}	EfficientNet	FEP(200)
ResNet	CIFAR-10	1	67.87% ± 0.76	**68.75% ± 0.16**	67.78% ± 0.26	67.44% ± 0.74	Adam	10^{-4}	EfficientNet	FEP(100)
		2	76.25% ± 0.25	76.24% ± 0.28	**76.32% ± 0.25**	76.29% ± 0.11	Adam	10^{-4}	EfficientNet	FEP(100)
	CIFAR-100	1	35.53% ± 0.74	**36.57% ± 1.07**	36.46% ± 0.82	36.08% ± 0.70	Adam	10^{-4}	EfficientNet	SSP(300)
		2	41.26% ± 0.56	41.30% ± 1.02	**41.45% ± 0.73**	40.89% ± 0.36	Adam	10^{-4}	EfficientNet	FEP(100)
Inception	CIFAR-10	1	76.91% ± 0.58	**77.34% ± 0.27**	77.13% ± 0.07	77.19% ± 0.17	Adam	10^{-4}	EfficientNet	FEP(300)
		2	79.06% ± 0.37	79.18% ± 0.12	**79.47% ± 0.52**	79.42% ± 0.15	Adam	10^{-4}	Inception	FEP(100)
	CIFAR-100	1	44.24% ± 0.70	**44.56% ± 0.44**	44.53% ± 0.42	44.07% ± 0.75	Adam	10^{-4}	Inception	FEP(200)
		2	45.86% ± 0.21	**46.50% ± 0.21**	45.13% ± 0.92	46.11% ± 1.28	Adam	10^{-4}	Inception	FEP(300)

*Branch Number
†Anti-Curriculum
§Random Curriculum
¶Learning Rate

3.5.3 Curriculum Learning for improving performance of Multi-Exit Convolutional Neural Networks

We assume that an already trained deep neural network is given at the beginning, which is augmented with a set of early exits. We keep the parameters of the trained neural network fixed and we train only the parameters of the early exit branches based on curriculum learning, in order to improve their accuracy. We use a pre-trained neural network as a teacher network and the categorical cross-entropy loss of this teacher for sorting the training samples based on their difficulty. We use two different teachers, InceptionV3 [141] which is the same teacher used in Hacohen et al. [139], and the more recent EfficientNetB7 [142]. Both these teacher networks are pre-trained on the ImageNet dataset [143] and then fine-tuned to the dataset of interest by removing the top layer and adding two dense layers with a Dropout layer [144] to match the number of classes in the output layer.

We use two variants of the *baby step* pacing function [139], the *fixed exponential pacing* function and the *single step pacing* function depicted in Fig. 20. These pacing functions introduce the entire dataset fairly quickly, leading to a curriculum that is introducing only simple examples to the neural network in the first few epochs.

We use four different backbone networks in our experiments, namely DenseNet201 [145], MobileNetV1 [146], ResNet152 [147] and InceptionV3 [141]. We train these networks on the CIFAR-10 and CIFAR-100 datasets [148] using transfer learning in the exact same way as the teacher networks. We place two early exit branches at two different layers on each backbone network, as shown in Table 15. All early exit branches have the same architecture, which is a convolution layer, followed by a maximum pooling layer, and three dense layers with a Dropout layer between each pair, as shown in Figure 21. We use the classifier-wise training strategy for training the multi-exit architecture. During the training of each branch, first we test both stochastic gradient descent and Adam [149] optimisers with different learning rates of $\{10^{-1}, 0.12, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ to obtain the highest accuracy for the normal training method without any curriculum, which we call *vanilla*. We chose to test the 0.12 learning rate in addition to 10^{-1} since it was the best case discovered for the experiments in Hacohen et al. [139]. With both optimisers, the learning rate is automatically reduced when the validation accuracy plateaus. Subsequently, using the same optimiser, we train the branch using the *curriculum* and *anti-curriculum* training methods.

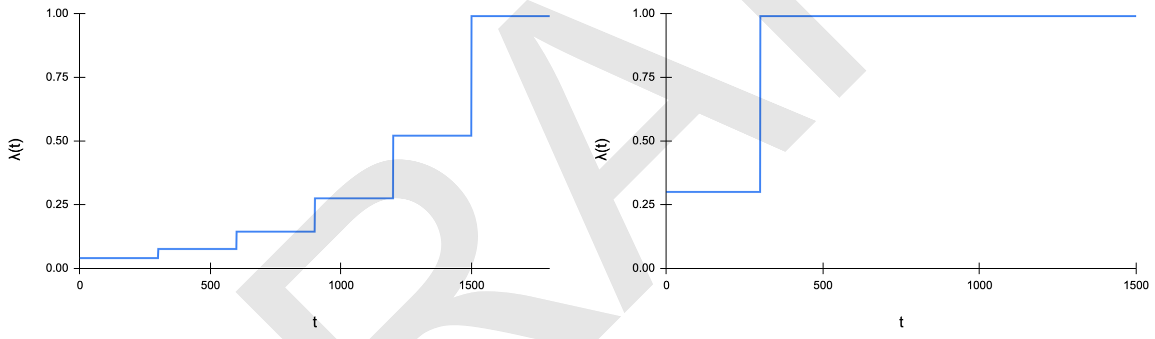


Figure 20: *Fixed exponential pacing* function (left) and *Single step pacing* function (right) used in our experiments.

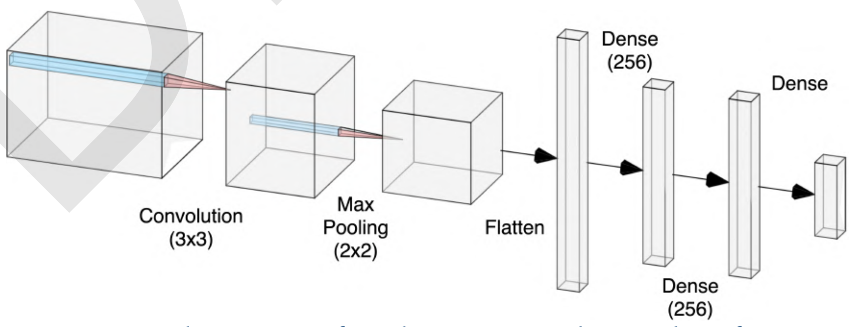


Figure 21: Architecture of Early Exit Branches (taken from [2]).

3.5.4 Copycat Fine-Tuning for improving performance of Multi-Exit Vision Transformers

We use Copycat [140] as a fine-tuning strategy for Transformer-based early exits [136]. In our experiments, we give images trained on the Tiny ImageNet dataset to a network

Table 15: Placement of Branches for Each Backbone Network.

Backbone	Dataset	BN*	Branch Placed After
DenseNet201	CIFAR-10	1	Layer 15 of 201
		2	Layer 40 of 201
	CIFAR-100	1	Layer 40 of 201
		2	Layer 137 of 201
MobileNet	CIFAR-10	1	Layer 8 of 28
		2	Layer 14 of 28
	CIFAR-100	1	Layer 8 of 28
		2	Layer 14 of 28
ResNet152	CIFAR-10	1	Layer 13 of 152
		2	Layer 38 of 152
	CIFAR-100	1	Layer 13 of 152
		2	Layer 38 of 152
InceptionV3	CIFAR-10	1	1st Filter Concat
		2	2nd Filter Concat
	CIFAR-100	1	1st Filter Concat
		2	2nd Filter Concat

*Branch Number

trained on CIFAR-10 and mix the resulting fake dataset with the original CIFAR-10 dataset with a 2-to-1 ratio, and use this new dataset to train Transformer-based early exit branches. The results of our experiments are shown in Table 16 where B1 and B2 represent branch locations 1 & 2. These results impact iKPI-3.3 (At least three (3) approaches tested for ML training algorithms) by providing an alternative approach for training.

Table 16: CC-SL-ViT: Performance on CIFAR-10, floating point operations (FLOPs), parameter count and maximum allocated memory at inference.

Model	Performance (%)	FLOPs (G)	Parameters (M)
SL-ViT, B1 (baseline)	71.06	1.64	0.59
CC-SL-ViT, B1 (ours)	71.61	1.64	0.59
SL-ViT, B2 (baseline)	81.18	5.26	0.79
CC-SL-ViT, B2 (ours)	83.41	5.26	0.79

3.6 Discriminant Learning-based initialisation of Feedforward Neural Networks

3.6.1 Introduction and Objectives

Initialisation of the parameters of neural networks is an important aspect in training them efficiently and achieving good performance. While a common practice is to initialise the neural network parameters by random sampling from an appropriate multi-dimensional distribution, it has been shown that the use of data-driven initialisation strategies can lead to faster training and some times to increased performance of the trained network. An explanation for such improved performance and training speed comes from the properties of the training processes used to train deep neural networks, which is commonly done by using a type of gradient-based iterative optimisation, like the error Backpropagation. Since the loss function used to train the neural network is defined on a set of parameters with an enormous size (i.e., equal to the number of the parameters of the neural network) and is non-convex, gradient-based optimisation leads to local minima solutions. Thus, starting the optimisation from a point in that high-dimensional space that considers the properties of the problem at hand can lead to faster convergence to a (hopefully better) local minimum, compared to a random point in the same high-dimensional space. Thus, the objective here is to define a way to use the training data for determining a good starting point to further perform gradient-based optimisation of the network parameter values. The methods discussed in this section are not directly contributing to any of the MARVEL components. The methods are contributing to the general development of visual representation learning and apply to the project from that point of view.

A summary of this work is provided hereafter. The corresponding paper is listed below, and can be found in Appendix 7.12:

- [3] K. Chumachenko, A. Iosifidis and M. Gabbouj, "Feedforward Neural Networks Initialization based on Discriminant Learning", Neural Networks, vol. 146, pp. 220-229, 2022.

3.6.2 Summary of the state-of-the-art

Widely adopted methods for initialising the parameters of a neural network based on random sampling use controlled parameters. For example, Glorot [150] initialises the parameters of a neural layer j by randomly sampling weights from a uniform distribution $U[-\alpha, \alpha]$, where $\alpha = \sqrt{6}/\sqrt{n_j + n_{j+1}}$, and n_j, n_{j+1} denote the number of neurons at layer j and $j+1$, respectively. Another commonly used random initialisation method using control parameters is He initialisation [151] which randomly samples weights from a Gaussian distribution with zero mean and $2/n_j$ variance, i.e., $\mathbf{W}_j \sim \mathcal{N}\left[0, \frac{\sqrt{2}}{\sqrt{n_j}}\right]$.

A popular approach for initialising the weights of a neural network is based on Transfer Learning [152]. In this case, the neural network is pre-trained on a larger dataset of similar properties, like ImageNet [153] for Computer Vision problems, and the trained network is further finetuned using the training data of the task at hand. Even though this is currently a very common practice, its effectiveness was questioned in [154]. In that study, it was shown that initialising the parameters of a neural network using those obtained by training on another task can lead to faster convergence, but not necessarily better performance when compared to using randomly initialised network parameters. Unsupervised network initialisation based on clustering was also shown to perform well [155, 156, 157], while the use of Principal Component Analysis (PCA) has also been proposed in [158]. For the case where the parameters of the network that we want to initialise are not vectors, e.g., when the parameters correspond to filters of Convolutional layers, appropriate vectorisation and tensorisation steps are performed.

The use of network weight initialisation is interesting, as it casts the weight initialisation problem as a subspace learning problem from one layer to the next one. This approach lead also to methods that employ supervised subspace learning methods for network weights initialisation, most of which employ Linear Discriminant Analysis

(LDA) [159, 160, 161]. However, all these approaches set limitations on the size of layers that can be used to form the neural network. Specifically, methods exploiting PCA to determine the parameters connecting layers j and $j+1$ the number of neurons in layer $j+1$ to be at most equal to the number of neurons in layer j, i.e., $n_{j+1} \leq n_j$. When LDA is used, the number of meaningful weights (i.e., those corresponding to the discriminant directions) determined by the method is restricted by the number of classes, i.e., $n_{j+1} \leq C-1$, where C is the number of the classes forming the classification problem defined at the image level. However, existing high-performing neural networks often do not follow these assumptions, as the appearance of bottleneck layers is very common. To account for this situation, these methods use random sampling schemes for initialising the remaining weights.

3.6.3 Description of work performed so far

A standard dense feedforward neural network receiving as input a vector $\mathbf{x} \in \mathbb{R}^D$ performs a hierarchical transformation through L layers

$$\mathbf{y} = f_a^L(\mathbf{W}_L^T f_a^{L-1}(\,\mathbf{W}_{L-1}^T \dots f_a^1(\mathbf{W}_1^T \mathbf{x} + b_1) + b_{L-1}) + b_L), \tag{2}$$

where $f_a^l(\cdot)$ is the (element-wise) activation function at layer l, $\mathbf{W}_l \in \mathbb{R}^{D_l \times D_{l+1}}$ is the corresponding weight matrix, and b_l is the bias term. A fully-Convolutional Neural Network performs a hierarchical data transformation of the form

$$\mathbf{y} = f_a^L(\hat{\mathbf{W}}_L * f_a^{L-1}(\,\hat{\mathbf{W}}_{L-1} * \;\dots\; * f_a^1(\hat{\mathbf{W}}_1 * \mathbf{x} + b_1) + b_{L-1}) + b_L), \tag{3}$$

where $\hat{\mathbf{W}}_l$ is a set of convolutional filters at layer l, b_l is the bias term, and $f_a^l(\cdot)$ is the activation function. For some Convolutional Neural Networks (CNNs), layers performing convolution data transformation of the form in (3) are combined with affine transformations of the form in (2) in a hierarchical manner.

For initialising the neural network parameters, we follow a progressive training process in which the parameters of one layer are initialised one after the other. In order to address limitations of existing network initialisation methods that employ supervised subspace learning, we propose the use of Subclass Discriminant Analysis (SDA) [162] in which each class (defined at the input image level) is allowed to form subclasses and the objective is to find a linear mapping to a new space where subclasses from different classes are well-discriminated. In the following, we use the abbreviation *fastSDA* to refer to this method. This approach has two advantages, the first being that by using this formulation one can define a mapping with an increasing number of dimensions as the dimensionality of the new space is restricted by the number of subclasses which is defined by the user. The second advantage is that the use of subclasses can allow for image patches corresponding to different locations, like different parts of the object/class of interest and different appearances of the background, in the input image to be grouped together. To make the process efficient, we employ a Spectral Regression-based solution of SDA which has been shown to lead to fast and accurate solutions [163]. Moreover, we proposed a vector batch normalisation process that allows for the use of the fast solution in [163] to be adopted for the network parameters optimisation.

3.6.4 Performance Evaluation

Experiments on image classification were conducted on three datasets, CIFAR-10 [148], MNIST [164], and Linnaeus-5 [165]. Experiments on vector classification problems, where dense feedforward neural networks are used can be found in the paper in Appendix 7.12. CIFAR-10 dataset contains images of 32 $\times$ 32 pixels with 3 channels and 10 object categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. MNIST dataset contains grayscale images of size 28$\times$28 posing a handwritten digit recognition problem. The Linnaeus-5 dataset contains RGB images of 32$\times$32 dimensionality of 5 object categories: berry, bird, dog, flower, and other. We use the provided train-test splits for evaluation. In the CIFAR-10 dataset, the training set is split into 48,000 images used for training, and 12,000 for validation. In the MNIST dataset, the training set is split into 40,000 and 10,000 images for training and validation, respectively. In both datasets, 10,000 images are used for testing. In Linneaus-5 datasets, 4,800 images are used for training, 1,200 for validation, and 2,000 for testing.

We used two CNN architectures with 5 and 6 hidden layers. We set $K_i = Z$ subclasses for all classes, leading to $CZ - 1$ filters in convolutional layers or neurons in dense layers at a position in the network. We construct the networks starting from 16 or 32 subclasses and reduce the number of subclasses by a factor of 2 with each subsequent layer. This results in two architectures with the layers having width of $\{319, 159, 79, 39, 19\}$ or $\{159, 79, 39, 19\}$ filters for MNIST and CIFAR datasets, and $\{159, 79, 39, 19, 9\}$ or $\{79, 39, 19, 9\}$ filters for Linnaeus-5 dataset. Another fully-connected layer of 128 neurons is added after the last convolutional layer. The output layer consists of 5 or 10 neurons depending on the dataset, and a softmax activation function. In convolutional layers, the bias terms are omitted in all models, and in fully-connected layers, they are initialised from zeros. To obtain the cluster labels during initialisation, mini-batch k-means clustering is performed [166]. A schematic illustration of the network structure is shown in Figure 22.

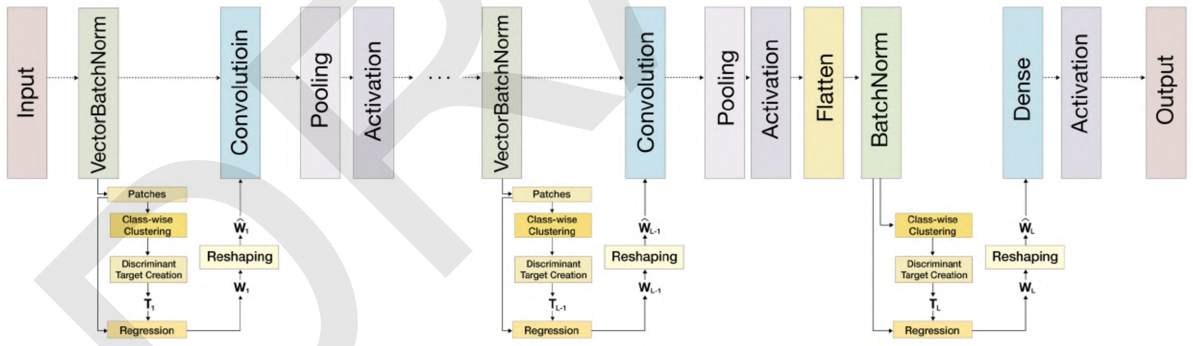


Figure 22: Structure of the CNN (taken from [3]).

We compare the proposed initialisation approach with Glorot initialisation [150], He initialisation [151], random initialisation from Gaussian distribution with $\mu = 0$ and $\sigma = 0.05$ (RNorm), random initialisation from uniform distribution in the range $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ (RUni), where n is the number of input neurons in the corresponding layers, K-Means initialisation, and PCA initialisation. We show that a small number of samples is generally sufficient to learn a good projection space that leads to competitive performance. To do this, we also test the case where only a limited number of training samples is used during the initialisation step. Specifically, we test the proposed approach with 200 and 500 samples per class (i.e., the total of 2000 or 5000 samples in CIFAR-10 and MNIST, and 1000 or 2500 samples in Linnaeus-5 dataset). After initialising the network parameters, we train the models with Stochastic Gradient Descent (SGD) with a learning rate of 0.001, a batch size of 32, and categorical cross-entropy as the loss function until the accuracy on the validation set stops improving for 10 epochs. The model that resulted in the best validation accuracy is then used for reporting the results on the test set. Images are mean-centered and re-scaled to match the range of 0 to 1.

Table 17 shows the accuracies obtained by using different initialisation methods. The best accuracy is highlighted in bold. The proposed method often outperforms competing methods. Considering the initialisation using a smaller number of samples, we observe that both 200 and 500 samples per class are often sufficient for outperforming the competing methods. Another fact worth noticing is that in a few cases, the use of a smaller number of samples leads to performance improvement compared to using the full dataset. A possible interpretation of this is that the model trained on a smaller number of samples overfits less to the training data, thus providing better generalisation properties.

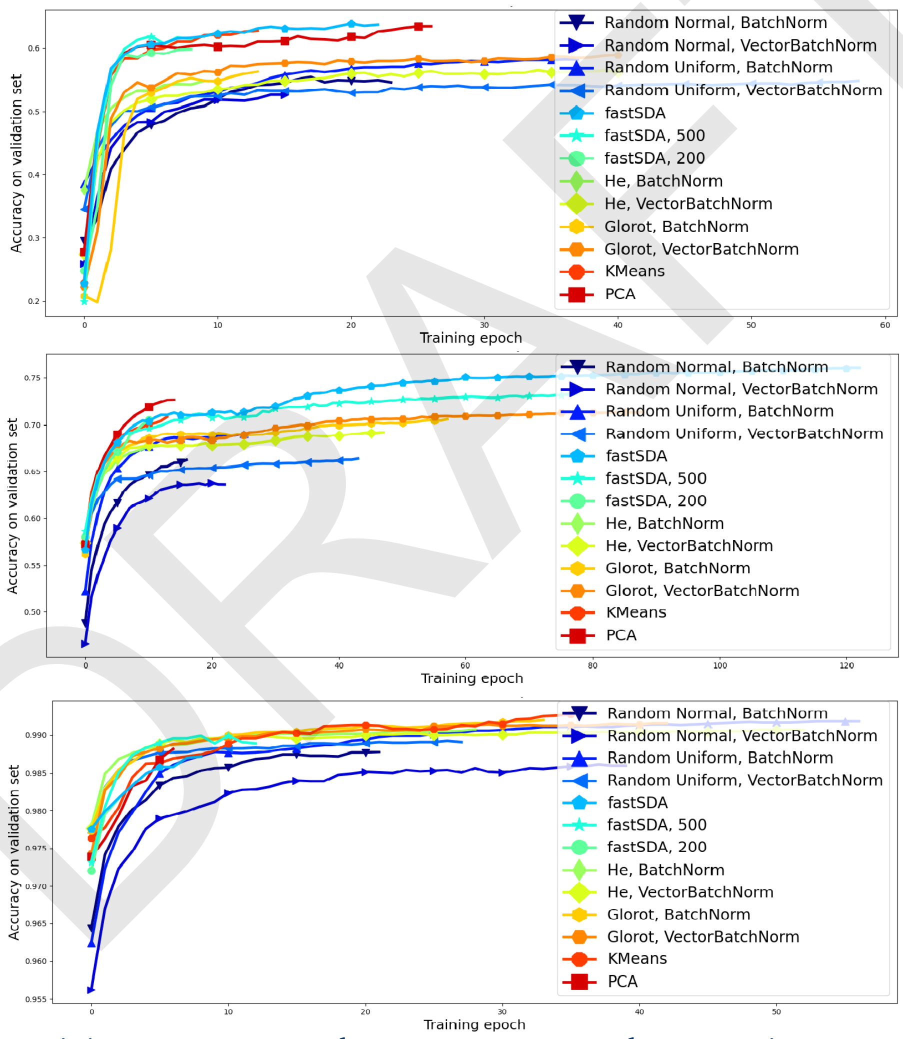


Figure 23: Training convergence plots. Datasets top to bottom: Linnaeus-5, CIFAR-10, MNIST (taken from [3]).

Figure 5 shows the accuracy of the validation set versus the number of training epochs when 16 subclasses and LeakyReLU activation function are used. We observe that the models initialised with the proposed method generally start in the first iteration with higher accuracy compared to the competing methods. The models initialised with the proposed method are able to achieve better overall accuracy. We also provide the initialisation times (in seconds) for the network architecture based on 32 subclasses in

Table 17: Classification accuracies on image classification problems.

Linnaeus-5

		LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
BNorm	RNorm	52.55	51.45	48.80	51.75	50.10	46.15
	RUni	59.00	55.25	49.65	58.50	58.30	47.70
	He	55.00	52.20	50.55	57.30	53.95	50.10
	Glorot	54.50	54.95	53.00	57.20	56.35	52.60
VecBNorm	RNorm	49.55	47.05	44.85	50.80	47.70	41.85
	RUni	53.90	51.55	44.30	51.85	52.90	43.80
	He	55.15	53.40	50.75	57.20	54.10	51.15
	Glorot	56.40	52.90	53.00	58.85	57.05	52.90
	KM	60.70	62.40	60.85	61.55	58.70	57.65
	PCA	60.90	62.90	60.00	**63.90**	60.35	59.75
	fSDA	**64.25**	62.30	61.75	59.75	62.70	**61.75**
	fSDA$_{500}$	62.00	**64.35**	**62.10**	61.45	**64.20**	61.70
	fSDA$_{200}$	60.65	62.90	59.70	60.95	64.05	59.20

CIFAR-10

		LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
BNorm	RNorm	66.17	63.67	59.35	64.89	62.05	63.78
	RUni	69.50	69.38	62.32	71.66	70.23	64.88
	He	68.49	68.74	65.89	70.18	71.34	64.57
	Glorot	71.71	72.04	68.41	73.77	73.51	66.94
VecBNorm	RNorm	63.65	61.89	59.96	64.50	60.68	62.71
	RUni	64.99	65.71	54.70	67.31	66.65	56.32
	He	69.17	68.17	64.24	71.54	70.11	64.88
	Glorot	71.45	71.75	65.79	73.73	72.88	68.56
	KM	72.03	75.01	68.52	**77.18**	76.20	67.03
	PCA	72.67	74.40	68.06	72.71	77.17	71.65
	fSDA	**75.02**	**75.36**	**70.59**	76.66	**77.79**	71.87
	fSDA$_{500}$	74.13	74.35	69.80	71.29	76.22	**72.60**
	fSDA$_{200}$	70.32	74.57	69.35	75.71	77.33	71.39

MNIST

		LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
BNorm	RNorm	98.82	98.78	98.45	98.94	98.70	98.41
	RUni	99.16	99.20	99.03	99.11	99.19	99.04
	He	99.00	99.17	99.03	99.19	99.30	99.10
	Glorot	99.10	99.30	99.23	99.22	**99.35**	99.24
VecBNorm	RNorm	98.76	98.35	98.30	98.76	98.63	98.19
	RUni	98.99	98.81	98.49	99.10	98.92	98.58
	He	99.03	99.14	98.95	99.13	99.14	98.87
	Glorot	**99.18**	99.21	99.08	99.15	99.16	99.22
	KM	99.10	99.07	99.12	99.21	99.20	99.18
	PCA	98.82	99.18	99.20	**99.25**	99.24	**99.34**
	fSDA	98.67	**99.26**	**99.24**	**99.25**	99.24	99.28
	fSDA$_{500}$	98.98	99.14	99.17	97.92	99.13	99.10
	fSDA$_{200}$	98.65	99.04	95.16	99.17	99.18	99.05

Table 18. As can be seen, the speed of initialisation depends both on dimensionality and dataset size. Using a smaller number of images for initialisation, the proposed method is generally faster when compared to the competing methods.

Table 18: Times for initialisation in 32-subclass architecture (in seconds).

	KM	PCA	fSDA	fSDA$_{500}$	fSDA$_{200}$
CIFAR	22020	12364	6315	807	532
MNIST	16301	6158	4516	521	294
LIN	958	1208	369	216	152

3.7 Neural network training with increased within-layer diversity

3.7.1 Introduction and Objectives

Neural networks perform complex data mappings from their input to their output which is obtained by performing multiple hierarchical data transformations. As described in Section 3.6, a feedforward neural network performs a transformation of the form:

$$\mathbf{y} = f_a^L(\mathbf{W}_L^T f_a^{L-1}(\mathbf{W}_{L-1}^T \dots f_a^1(\mathbf{W}_1^T \mathbf{x} + b_1) + b_{L-1}) + b_L), \tag{4}$$

where $f_a^l(\cdot)$ is the (element-wise) activation function at layer l, $\mathbf{W}_l \in \mathbb{R}^{D_l \times D_{l+1}}$ is the corresponding weight matrix, and b_l is the bias term.

After initialising the parameters of the network $\{\mathbf{W}_l, b_l\}_{l=1}^L$, an iterative optimisation of its parameters is usually performed by using a set of training data and label pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$ to optimise a loss function expressing the empirical loss. The empirical loss is expressed by the network error, i.e., a deviation between the network output on the training data with respect to the corresponding labels. At every iteration of the optimisation process, the gradient of the loss is used to update the network parameters. That is, at each optimisation step, neurons at a given layer receive feedback from neurons belonging to higher layers in the network topology. Often, the relationships between the neurons of the same layer are ignored in this optimisation. We showed that promoting diversity between the neurons of each layer can lead to better generalisation performance of the neural network. This is achieved by complementing the traditional 'between-layer' feedback with an additional 'within-layer' feedback to encourage diversity of the activations of the neurons within the same layer. The methods discussed in this section are not directly contributing to any component of the MARVEL project. The methods are contributing to the general development of visual representation learning and apply to the project from that point of view.

A summary of this work is provided hereafter. The corresponding paper is listed below, and can be found in Appendix 7.10:

- [167] F. Laakom, J. Raitoharju, A. Iosifidis and M. Gabbouj, "Within-layer Diversity Reduces Generalization Gap", International Conference on Machine Learning Workshop on Information Theoretic Methods for Rigorous, Responsible and Reliable Machine Learning, pp. 1-11, 2021.

3.7.2 Summary of the state-of-the-art

Deep neural networks are often over-parameterised, i.e., they have more parameters compared to the number of data samples used to train them. This often leads to overfitting to the data used in their training, which results in low generalisation performance on data that are not used during training [168]. Avoiding overfitting has been extensively studied in the literature [169, 170, 171] and several training strategies were proposed to avoid it. These include data augmentation [168, 172], the use of regularisation [173, 174, 175], and Dropout [176, 177, 178, 179].

Diversity has been used for improving the performance of ensemble methods [180, 181]. It has also been used deep learning for improving generalisation through diversifying the information extracted by different network neurons [182, 183]. A common practice to achieve this is by restricting the weights of each layer of the network to be diverse, e.g., through orthogonalisation [184, 185]. However, the diversity of the activations, i.e., the outputs of the neurons, has not received much attention. The method in [186] is the only one that considers the diversity of the activations directly by including an additional loss term using cross-covariance of hidden layer activations. This encourages the neurons to learn diverse or non-redundant representations. This approach was empirically shown to reduce overfitting and improve the generalisation performance of the neural network. However, a theoretical analysis to prove that increasing the diversity of the hidden layer neuron activations increases generalisation performance of the neural network is lacking.

3.7.3 Description of work performed so far

We proposed a neural network training strategy, where we encourage neurons within a layer to activate in a mutually different manner. Such an approach will lead the trained neurons to capture different patterns in the input data. To do this, we include an additional within-layer loss which penalises pairs of neurons that activate similarly. This is expressed by using an augmented training loss of the form:

$$\hat{L}_{\text{aug}}(f) = \hat{L}(f) + \lambda \sum_{i=1}^{P} J^i, \tag{5}$$

where $\hat{L}(f)$ is the standard empirical loss used for training the neural network, J^i expresses the overall similarity of the neurons within the i^{th} layer and λ is a scaling factor weighting the importance of the two terms. The loss Eq. (5) can be applied to a single layer, or to multiple layers of the network.

Let $\phi_n^i(\mathbf{x}_j)$ and $\phi_m^i(\mathbf{x}_j)$ be the outputs of the n^{th} and m^{th} neurons in the i^{th} layer of the network for the same input sample $\mathbf{x}_j$. The similarity s_{nm} between the the n^{th} and m^{th} neurons is defined as the average similarity of their outputs for N input samples. We use the radial basis function to express the similarity:

$$s_{nm} = \frac{1}{N} \sum_{j=1}^{N} \exp\big(- \gamma ||\phi_n^i(\mathbf{x}_j) - \phi_m^i(\mathbf{x}_j)||^2 \big), \tag{6}$$

where γ is a hyper-parameter value. The similarity s_{nm} can be computed over the whole dataset or batch-wise. When two neurons n and m have similar outputs for many samples, their similarity s_{nm} will be high. Otherwise, their similarity s_{mn} will be small. Based on these pair-wise similarities, we define three variants for the overall similarity J^i:

- **Direct:** $J^i = \sum_{n \neq m} s_{nm}$. In this variant, we model the global layer similarity directly as the sum of the pairwise similarities between the neurons.
- **Det:** $J^i = -\det(\mathbf{S})$, where $\mathbf{S}$ is defined as $\mathbf{S}_{nm} = s_{nm}$. This variant is inspired by the Determinantal Point Process (DPP) [187].
- **Logdet:** $J^i = -\text{logdet}(\mathbf{S})$. This variant has the same motivation as the second one. We use logdet instead of det as logdet is a convex function over the positive definite matrix space.

3.7.4 Performance Evaluation

We conducted experiments on two image datasets, the CIFAR10 and CIFAR100 [188]. They contain 60,000 images with a resolution of 32×32 pixels grouped into 10 and 100 distinct categories, respectively. The datasets are split into training and test subsets with 50,000 and 10,000 images, respectively. We split the original training set to create a training and a validation set. We use the first 40,000 images as the main training set and the last 10,000 as a validation set for optimising the hyperparameter values. We use three state-of-the-art CNNs:

- **ResNext 29-8-16**: we consider the standard ResNext Model [189] with a 29-layer architecture, a cardinality of 8, and a width of 16.
- **DenseNet-12**: we use DenseNet [190] with the 40-layer architecture and a growth rate of 12.
- **ResNet50**: we consider the standard ResNet model [191] with 50 layers.

We compare against the original networks as well as networks trained with the method in [186], which is referred to as DeCov.

All the models are trained using stochastic gradient descent (SGD) with a momentum of 0.9, weight decay of 0.0001, and a batch size of 128 for 200 epochs. The initial learning rate is set to 0.1 and is then decreased by a factor of 5 after 60, 120, and 160 epochs, respectively. We also adopt a standard data augmentation scheme that is widely used for these two datasets [191, 190]. For all models, the additional diversity term is applied on top of the last intermediate layer. The loss weight is chosen from $\{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01\}$ for both our approach and Decov and γ in the radial basis function is chosen from $\{0.01, 0.1.1, 10, 50, 100\}$. For each approach, the model with the best validation performance is used in the test phase. Each experiment is repeated three times and we report the average performance over three iterations.

Table 19 reports the average top-1 errors of the different methods. It can be seen that the use of diversity generally improves performance. Comparing the results of Deconv and our method, we can see that a variant of our method is consistently providing the best test error value.

We also conducted additional image classification experiments on the ImageNet-2012 classification dataset using ResNet50. For the hyperparameters, we use the best ones for each approach obtained from the experiments conducted on CIFAR10 and

Table 19: Average classification errors on CIFAR10 and CIFAR100 over three iterations.

		Top-1 test Error	
Model	Method	CIFAR10	CIFAR100
DenseNet-12	Standard	7.07	29.25
	DeCov	7.18	29.17
	Ours direct	**6.95**	29.16
	Ours det	7.04	**28.78**
	Ours logdet	6.96	29.15
ResNext-29-08-16	Standard	6.93	26.73
	DeCov	6.84	26.70
	Ours direct	6.74	**26.54**
	Ours det	**6.67**	26.67
	Ours logdet	6.70	26.67
ResNet50	Standard	8.27	34.06
	DeCov	8.03	32.26
	Ours direct	7.86	32.15
	Ours det	**7.73**	**32.12**
	Ours logdet	7.91	32.20

Table 20: Performance of ResNet50 with different diversity strategies on ImageNet dataset.

	Top-1 Errors		Generalisation
Method	Training	Testing	Gap
Standard	20.97	23.84	2.87
DeCov	20.92	23.62	2.70
Ours direct	20.88	**23.58**	2.70
Ours det	20.81	23.62	2.77
Ours logdet	**22.57**	**23.64**	**1.07**

CIFAR100 datasets. Table 20 reports the test errors of the different methods. We also report the final training errors and the generalisation gap, i.e., the difference between the training and test accuracies. As can be seen, the best test error is achieved by the direct variant of the proposed method, while the smallest generalisation gap is by the logdet variant of the proposed method.

4 Methodologies for multi-modal data analysis

In this Section, methodologies developed by MARVEL partners targeting multi-modal data analysis are described. These include a methodology for Audio-Visual Crowd Counting (Section 4.1), and a methodology for learning enhanced audio representations by following a cross-modal contrastive learning approach that combines multiple types of information related to music to learn audio representation (audio feature) from heterogeneous data, i.e., playlists-track interactions, genre metadata, and the tracks' audio (Section 4.2). Finally, a method for Dynamic Split Computing in Deep Learning models suitable for improving computing resources allocation between edge devices and fog or cloud computing is described in Section 4.3.

4.1 Efficient Audiovisual Crowd Counting with Single-Layer Vision Transformers

4.1.1 Introduction and objectives

When both visual and audio information is available, one can combine this enriched information in order to improve performance compared to using only information from one stream, e.g., the visual information used in the visual crowd counting method of Section 3.1. In audiovisual crowd counting, the ambient audio of the scene is used alongside its image in order to estimate the number of people appearing in the scene. The ambient audio can help improve the accuracy in situations where the quality of the input image is not ideal, for instance, in scenes with low illumination such as night-time, in cases where the capture device is noisy or outputs a low-resolution image, and in situations where some people are partially or completely occluded in the image.

Therefore, the input to an audiovisual crowd counting model is commonly a color image as well as a corresponding audio clip that spans the time period before and after the image was taken. Similar to visual crowd counting, the expected output is a number representing the total number of people present in that image, and the crowd counting model can optionally provide as output a density map that specifies the density of the crowd for each pixel of the input image.

Here we describe a different architecture for early exit branches placed on crowd counting models, that can lead to higher performance and can be combined with a neural network branch receiving as input audio information. A summary of this work is provided hereafter. The corresponding preprint is listed below, and can be found in Appendix 7.11:

- [136] A. Bakhtiarnia, Q. Zhang, and A. Iosifidis. Single-layer vision transformers for more accurate early exits with less overhead, accepted to Neural Networks journal (10.5281/zenodo.6737408), 2021.

4.1.2 Summary of the state-of-the-art

At the time of this writing, the DISCO dataset [192] is the only publicly available dataset for audiovisual crowd counting. DISCO consists of 1935 images of Full HD resolution (1920×1080) as well as a 1-second ambient audio clip for each image which starts 0.5 seconds before the image was captured and ends 0.5 seconds afterwards. For each image, annotations showing the location of each person's head in the image are also available. These head annotations can be converted to ground truth density maps and used to train and test the model. DISCO is a challenging dataset due to the variability of the images in terms of perspective, illumination, and crowd density.

AudioCSRNet [192], evaluated on DISCO, is a high-performing model for audiovisual crowd counting, which is an extension of the popular CSRNet model that is widely used for visual crowd counting. CSRNet utilises the features extracted by the first 10 layers of a VGG neural network [193] pre-trained on the ImageNet dataset [194], and processes these features provided using six dilated convolution layers. AudioCSRNet extends this architecture by extracting audio features from input audio waveform using a VGGish neural network [195] pre-trained on the AudioSet dataset [195]. Subsequently, these extracted audio features are merged with video features using six fusion blocks, each containing a dilated convolution. The architecture of AudioCSRNet is depicted in Figure 24.

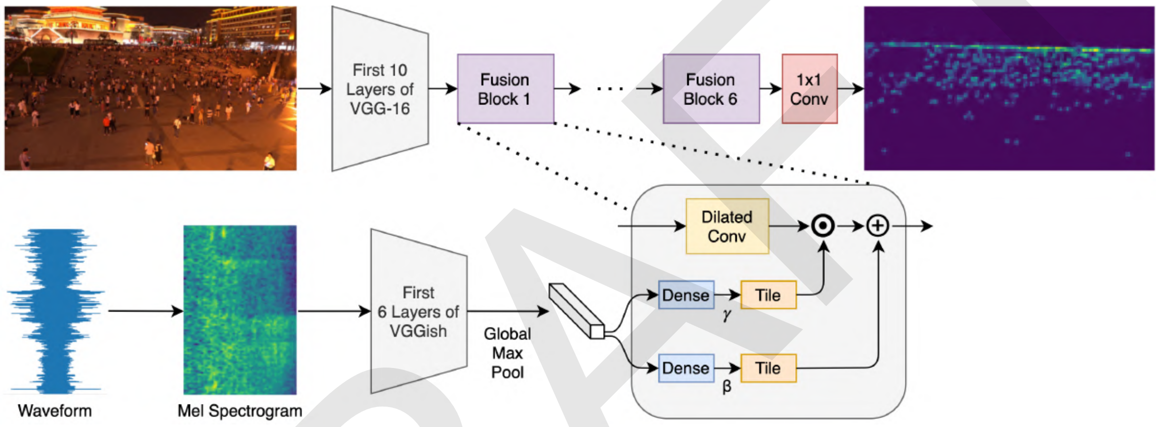


Figure 24: Architecture of AudioCSRNet.

Similar to the crowd counting method described in Section 3.1, audiovisual crowd counting also requires high computational power. For instance, AudioCSRNet requires nearly 500 billion floating point operations per second (FLOPS) during inference. Therefore, dynamic inference methods such as early exiting are used in the audiovisual case as well. As previously mentioned, results obtained from early exit branches are typically less accurate than the final result, thus improving the accuracy of early exits is an active area of research.

4.1.3 Description of work performed so far

Our method titled “Single-Layer Vision Transformer” (SL-ViT) [136] is an alternative architecture for early exit branches, inspired by the Vision Transformer architecture [109], which leads to more accurate early exit branches compared to conventional early exit branch architectures. Additionally, the overhead of our method is less than the conventional approach both in terms of the number of parameters and FLOPS. Figure 25 depicts the conventional CNN-based architecture for early exits, and Figure 26 shows the architecture of SL-ViT.

Besides a lower overhead and higher accuracy, another advantage of SL-ViT is that audio and video features can be directly fused inside the early exit branch, leading to audiovisual SL-ViT (AV-SL-ViT). The first benefit of AV-SL-ViT is its higher accuracy due to the fact that it is not required to fuse the modalities using element-wise addition and

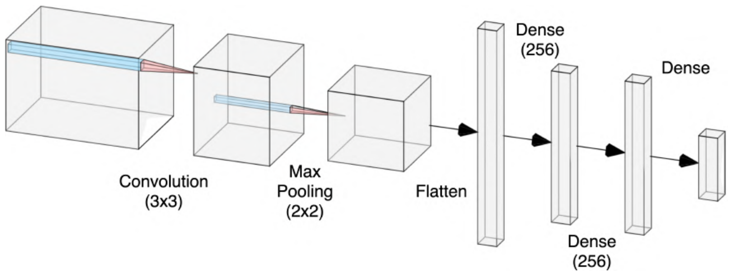


Figure 25: Conventional CNN-based early exit branch architecture. Figure created using the NN-SVG tool [4].

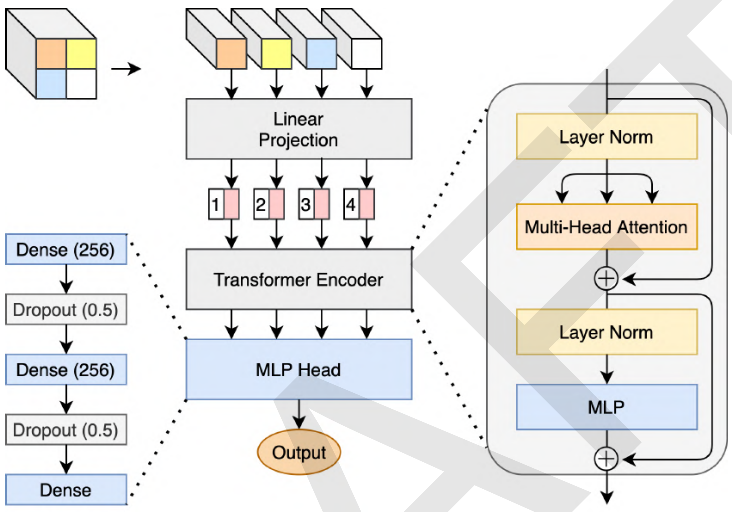


Figure 26: SL-ViT early exit branch architecture.

multiplication of tensors, which can introduce a lot of noise. Another benefit of AV-SL-ViT is that it provides more options for early exit locations since any intermediate layer of the visual part of the neural network can be combined with any intermediate of the audio part of the neural network to create audiovisual early exits, which leads to a more fine-grained dynamic inference.

4.1.4 Performance evaluation

Our results are summarised in Table 21. In this table, the accuracy is measured using mean absolute error (MAE) where lower is better. The SL-ViT early exit branch is placed before the first fusion block and the AV-SL-ViT early exit branch uses the outputs of VGG and VGGish. The number of parameters is calculated only for the early exit branch, however, the FLOPS indicate the total operations of the neural network up to and including the early exit branch.

These results contribute to the KPI-O2-E2-1 (average accuracy enhancement for audio-visual representations and models at least 20%) by achieving a 3.06% enhancement, and complete KPI-O2-E3-1 (increase the average accuracy for audio-visual event detection by at least 10%) and KPI-O2-E3-3 (decrease the time needed to identify an event by at least 30% of current time) by obtaining a 14.24% increase and 48% decrease, respectively.

Table 21: Single-Layer Vision Transformer (SL-ViT): Performance on the DISCO dataset, floating point operations per second (FLOPS), parameter count and maximum allocated memory of SL-ViT at inference.

Model	MAE	Parameters (M)	FLOPS (G)
Baseline (CNN)	16.99 ± 0.28	2.50M	329.77B
Proposed method (SL-ViT)	15.04 ± 0.71	2.35M	328.72B
Proposed method (AV-SL-ViT)	14.58 ± 0.64	2.36M	330.31B

4.2 Cross-modal contrastive learning

4.2.1 Introduction and objectives

The purpose of the work was to study how information from multiple sources could be combined into the same representation for music information retrieval applications such as musical genre classification, playlist continuation, and automatic tagging. Deep learning techniques can be used to obtain representations based on various sources of information. The cross-modal contrastive learning technique is proposed to combine multiple types of information related to music to learn audio representation (audio feature) from heterogeneous data. The methods discussed in this section are not directly contributing to any component of the MARVEL project. The methods are contributing to the general development of audio representation learning and apply to the project from that point of view. The paper to give details about the work is listed below and can be found in Appendix 7.14:

- [196] A. Ferraro, X. Favory, K. Drossos, Y. Kim and D. Bogdanov, "Enriched Music Representations with Multiple Cross-modal Contrastive Learning", IEEE Signal Processing Letters, vol. 28, pp. 733-737, 2021

4.2.2 Summary of the state-of-the-art

Information from multiple sources can be used in music information retrieval applications. Audio features are the best-predicting information for predicting musical genre [197], whereas users' listening data is the most suitable for music recommendation [198] and mood prediction [199]. Recent advances in deep learning allow for improving the performance on multiple tasks by combining different types of data (multimodal) [200, 201]. Deep learning techniques allow learning representation mappings from input data to an embedding space to be used for multiple downstream tasks [202]. Contrastive learning allows to learn representations by utilising learning objective to contrast similar (positive examples) and dissimilar items (negative examples) [203]. Triplet loss [204] based approaches have been recently used for music information retrieval [197] and zero-shot learning [205]. These approaches use triplets that are composed of an anchor, a positive, and a negative example, and the sampling strategy for these triplets is crucial to the learning process. The approaches using contrastive loss functions in a self-supervised learning scheme have recently provided robust image [206], environmental sound [207] and music audio [208] representations that are learned without annotated data. In a supervised learning scheme, contrastive learning has been used in a cross-modal approach where audio information and associated textual metadata are used to learn semantically enriched audio representations [209, 210].

4.2.3 Description of work performed so far

The proposed method aligns the latent representations obtained from playlists-track interactions, genre metadata, and the tracks' audio, by maximising the agreement between these modality representations using a contrastive loss [203]. The proposed method is evaluated in three downstream music information retrieval tasks, genre classification, playlist continuation, and automatic tagging.

The proposed method consists of encoders for the audio, musical genre, and music playlist information. These encoders are used to obtain latent representations, and the information from these encoders is mutually aligned using three contrastive losses between associated and non-associated examples. The joint minimisation of these losses is used in the optimisation to obtain embeddings of music signals. The performance is compared with a baseline audio-based CNN model trained for three tasks specifically.

The main contributions of the work are an updated audio encoder (embedding model) optimised for the music domain based on [209, 211] and the alignment of multi-modal data for exploiting the semantic metadata and collaborative filtering information. The work includes an ablation study by comparing the performance of each source of information independently to evaluate the importance of the different parts of the model.

4.2.4 Performance evaluation

The main results are summarised in Table 22. The results show that the proposed audio embedding trained using contrastive loss gives better performance than the baseline model trained directly for the task using only task-specific information. Detailed results can be found in [196] and in Appendix 7.14.

Table 22: Performance on genre classification, automatic audio tagging, and automatic playlist generation with a baseline method learned using only target information and proposed method learned using cross-modal contrastive learning.

Task	Metric	Baseline	Proposed
Genre classification	Mean accuracy±STD	63.28±1.19	**76.78±1.22**
Tagging, genre	ROC AUC±STD	0.840±0.004	**0.847±0.004**
Tagging, mood	ROC AUC±STD	0.722±0.004	**0.732±0.005**
Tagging, instrument	ROC AUC±STD	0.781±0.005	**0.797±0.005**
Playlist generation	normalised Discounted Cumulative Gain, nDCG@100	0.0044	**0.0020**
Playlist generation	Mean Average Precision, MAP@100	0.0007	**0.0020**

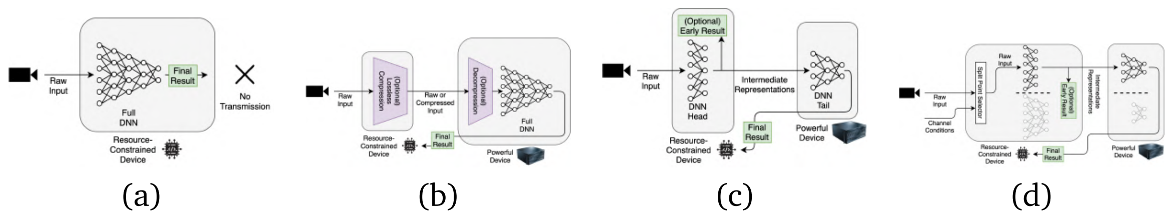


Figure 27: Overview of (a) no offloading; (b) full offloading; (c) split computing; and (d) dynamic split computing approaches.

4.3 Efficient Deep Learning on the Edge with Dynamic Split Computing

4.3.1 Introduction and objectives

Deep learning benefits many IoT applications [212], however, deep learning models are often computationally expensive which makes them difficult to deploy on resource-constrained IoT devices. Offloading the computation to an edge server or cloud server (Fig. 27 b) can solve this issue [213], however, it also introduces a few drawbacks. First, the size of the inputs to deep learning models is often large, particularly computer vision models which take images and videos as input. This means that offloading the computation to a server consumes a lot of bandwidth as well as energy, and introduces delays due to transmission. Secondly, even though IoT devices are restricted in terms of their computational capabilities, they still possess some amount of computational power which remains unused when the computation is fully offloaded to a server. Finally, since the server is collecting the raw input from several IoT devices, privacy concerns may arise.

Split computing is a new paradigm that partitions the deep neural network (DNN) into two sections [214]. The initial section called *head* is executed on the resource-constrained device. Subsequently, the output of the last layer of the head is transmitted to a server, where this intermediate representation is given as input to the second section of the DNN called *tail* and the final answer of the DNN is obtained. Fig. 27 (c) shows the split computing procedure.

When the size of the intermediate representation is smaller than the input size, split computing consumes less bandwidth and energy compared to full offloading. Moreover, the transmission delay is reduced and, since the intermediate neural network layer representations are transmitted to the server instead of raw inputs, privacy is preserved. Furthermore, the computational capability of IoT devices is utilised for performing part of the computations, which reduces the workload of the server. We proposed a dynamic version of split computing which optimises the split location based on the state of the communication channel and leads to faster inference compared to static split computing.

A summary of this work is provided hereafter. The corresponding preprint is listed below, and can be found in Appendix 7.15:

- [215] A. Bakhtiarnia, N. Milosevic, Q. Zhang, D. Bajovic and A. Iosifidis, "Dynamic Split Computing for Efficient Edge Intelligence", International Conference on Machine Learning (ICML) Workshop on Dynamic Neural Networks, 2022.

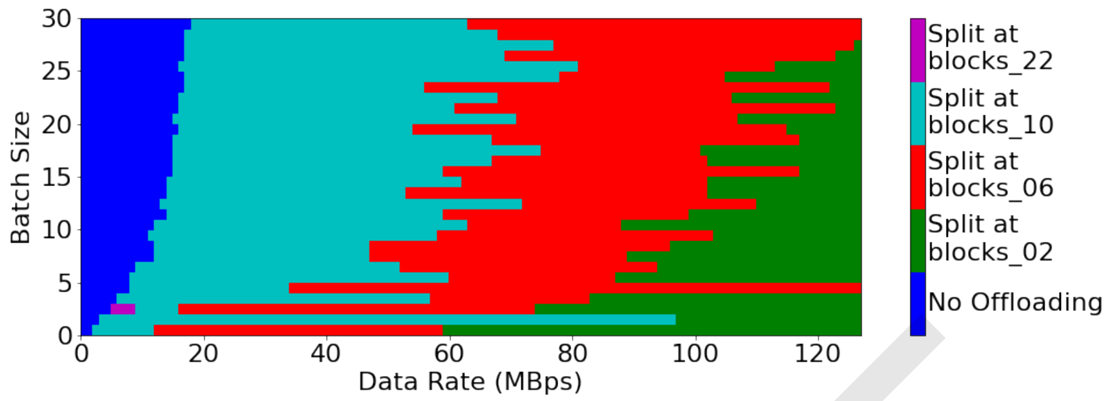


Figure 28: Optimal split location based on batch size and data rate for the EfficientNetV1-B4 architecture.

4.3.2 Summary of the state-of-the-art

Natural bottlenecks are layers of DNNs where the size of the intermediate representation is less than the input size. Not all DNNs contain such natural bottlenecks. In such cases, a bottleneck can be inserted into the architecture, for instance, by adding a few encoder and decoder layers in the middle of the DNN within and re-training the new DNN [214]. However, bottleneck insertion causes several issues. First, re-training the DNN is a time-consuming task that requires multiple steps such as identifying a proper split location, finding the optimal bottleneck size, and hyper-parameter optimisation. Furthermore, there is no guarantee that the re-trained DNN obtains a comparable performance, particularly with the added restriction of a bottleneck in the architecture.

4.3.3 Description of work performed so far

Dynamic split computing takes advantage of the fact that many natural bottlenecks exist in modern DNN architectures such as EfficientNet [216] and EfficientNetV2 [217], which means that dynamic split computing does not need to insert bottlenecks into the architecture and thus preserves the accuracy of the original DNN. Furthermore, dynamic split computing takes into account the data rate (bandwidth) of the communication channel as well as the batch size (workload), and calculates the expected end-to-end inference time for each possible split location. Based on these calculations, dynamic split computing can select the optimal split location that results in the lowest end-to-end inference time. The optimal split location varies over time whenever the state of the communication channel changes.

4.3.4 Performance evaluation

The results for the EfficientNetV1-B4 architecture are shown in Fig. 28, where for each data rate and batch size, the optimal split location is specified. For each possible state of the communication channel, the relative gain of dynamic split computing over static split computing for the EfficientNetV1-B4 architecture in terms of inference speed is shown in Fig. 29. Observe that each bottleneck can be an optimal split location in several scenarios, therefore, choosing the split location dynamically based on the state of the communication channel improves inference speed. This is also the case with the other variations of EfficientNet and EfficientNetV2 architectures.

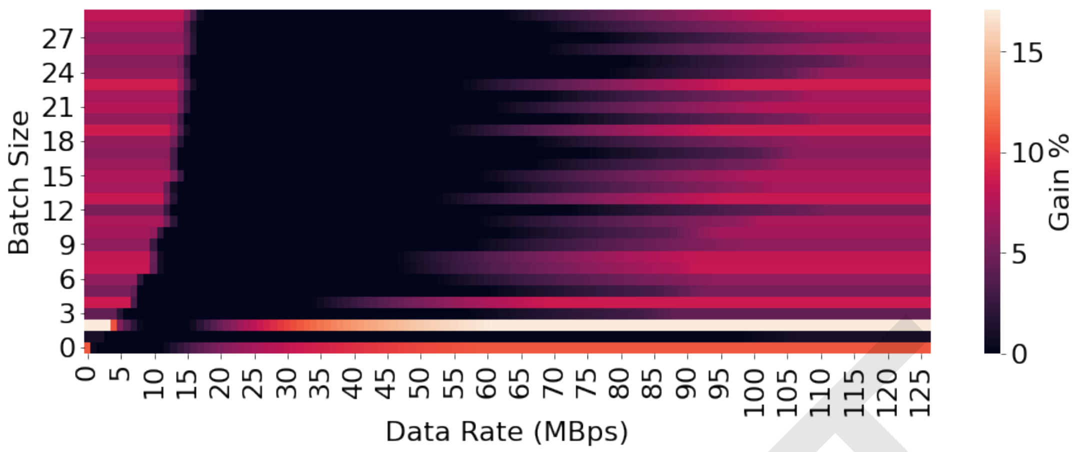


Figure 29: The relative gain of dynamic split computing in terms of end-to-end inference time over static split computing at block 10 in the EfficientNetV1-B4 architecture.

4.3.5 Future work

While in the results presented above only single-branch deep learning models were tested, we are currently working on testing the method on audio-visual data analysis processed by multiple branches followed by layer fusion block(s). This will allow for efficient split computing in multi-modal data analysis tasks.

5 Future plans

During the remaining part of the project, work in T3.3 will further focus on methodology development and training of the developed models in data coming from the MARVEL Data Corpus.

Sound event detection development will mainly focus in the future on optimising the performance in the MARVEL use cases. The focus will be on data unbalance problem leading to unbalanced detection across different sound event classes and learning robust models that generalise across multiple use case locations with a limited amount of learning examples. Both issues can be addressed by utilising transfer learning and data augmentation techniques. Work on these issues will further contribute to KPI-O2-E3-1. The development in sound event localisation and detection will focus as well on optimising the performance in MARVEL use cases. Fully annotated real-life data is difficult to produce for the task as this requires detailed location annotations for each sound source in the scene. Future research will focus on ways to utilise synthesised material generation to produce learning examples similar to the ones encountered in the use cases. This work will contribute to KPI-O2-E3-1 and KPI-O2-E2-3. Possibly future directions in the development of automated audio captioning methods include techniques coping with a low amount of learning examples with a narrow vocabulary.

Regarding visual data analysis, work related to visual crowd counting and visual anomaly detection will mainly focus on methodology development for improving performance and/or operation speed in connection to the MARVEL use cases. The Dynamic Split Computing methodology will be further developed and evaluated on visual and audio-visual data analysis related to the MARVEL use cases. Furthermore, we will investigate the case of high-resolution visual data analysis, which has the potential to improve performance. In this case, emphasis will be given to methodology development for improving inference speed, as this is the major bottleneck in analysing high-resolution visual data. Work on these issues will further contribute to KPI-O2-E2-1, KPI-O2-E3-2, and KPI-O2-E3-3.

Regarding multimodal data analysis, we will try to improve performance in audio-visual crowd counting and extend our visual anomaly detection to exploit both visual and audio data. Again, here the use of high-resolution visual information can have the potential in improving performance. Work on these issues will further contribute to KPI-O2-E2-1, KPI-O2-E3-2, and KPI-O2-E3-3. Furthermore, work on automated audio-visual captioning will be started by identifying the state-of-the-art methods, finding the possibilities for novel methods, and defining a benchmark dataset for the development.

For what concerns the SED methods for edge devices, the next efforts will focus on combining scalable models with more traditional model compression techniques (pruning). In addition, as new in-domain data will be released by the pilots, the sound event detection at the edge will be evaluated and further developed.

6 Conclusions

This document provided a description of the methodological contributions of MARVEL partners during the first half of the project duration (M08-M18) within T3.3 *Multimodal audio-visual intelligence*. The developed methodologies focus on six of the AI components included in the MARVEL Audio, Visual, and Multimodal AI Subsystem, and specifically, Sound Event Detection (SED), Sound Event Localisation and Detection (SELD), Automated Audio Captioning (AAC), Visual Crowd Counting (VCC), Visual Anomaly Detection (ViAD), and Audio-Visual Crowd Counting (AVCC). Moreover, methodologies for improving the training and efficiency of deep learning models when applied to audio, visual and multimodal analysis tasks were developed. The aim, along with the main contributions of the developed solutions are summarised in the following.

The aim of sound event detection is to provide a textual label, and the start and end time of targeted sound event instances in the audio signal. The development focused on vehicle type detection task. The main contribution of the proposed method is more balanced detection performance across vehicle types classes in comparison to the state-of-the-art baseline. The method targets the SED component in the MARVEL architecture and addresses a KPI related to detection accuracy (KPI-O2-E2-1).

Sound event detection can be performed on low-resource edge devices in order to reduce the bandwidth and energy required for data transfer and to ensure the preservation of the privacy of the citizens being recorded by the microphones. The main contribution of SED@Edge is the development of scalable neural architectures which allows an effective configuration of the neural models, given the available resources, while preserving the detection accuracy. This component contributes to KPI-O2-E3-3.

Sound event localisation and detection jointly detect the active sound events and their spatial locations (elevation and azimuth). The development focused on approaches suitable for real-time usage in the smart city domain and investigated the effect of replacing recurrent neural network blocks with self-attention. The method targets the SELD component in the MARVEL. The proposed neural network architecture decreases the inference time by 40% and this result completes KPI-O2-E3-3. Automated audio captioning aims at providing textual descriptions for a segment of audio. The development focused on neural network architectures that model spectro-temporal context, and continual learning for automated audio captioning. The methods contribute to the AAC component in MARVEL.

A system development from a large set of unlabelled data through active learning and domain adaptation was investigated in the case of speech emotion recognition application. The proposed methods are not directly contributing to any components in MARVEL, however, utilising large unlabelled datasets is an important issue for any AI system development. Another developed method studied how engagement in Citizen Science projects applied to Smart Cities infrastructure can be enhanced through contextual and structural game elements. The work addressed the issue of how to collect and annotate learning examples for AI development in the Smart City context by utilising users of a game. The proposed framework does not contribute directly to any MARVEL component.

Visual crowd counting aims at estimating the number of people visible in an image or video frame. The proposed method is based on multi-exit deep learning architectures allowing it to provide an estimate of the number of people under varying computing resources and network bandwidth allocation. The method targets the VCC component in the MARVEL architecture and addresses KPI related to accuracy enhancement (KPI-O2-E2-1). Visual anomaly detection aims at detecting novel situations in a visual scene. Two methods were proposed, one based on unsupervised learning through the reconstruction of the visual scene, and another one through detection of objects in the scene and employing user-defined anomaly rules. The methods target the ViAD component in the MARVEL architecture and address KPIs related to decrease event identification time (KPI-O2-E3-3) and increase average event detection accuracy (KPI-O2-E3-1). Another developed method studied how to estimate social distances in smart cities with non-invasive means. The work employs person detectors and provides the pair-wise distances of the existing persons in the scene. The proposed method does not contribute directly to any MARVEL component, but it is related to the visual scene analysis aspects of MARVEL.

A key aspect in achieving high performance in visual data analysis based on deep learning models is effective initialisation of their parameters and efficient training. Methods exploiting discriminant learning ideas for parameter initialisation, diversity of neurons forming each neural network layer, and data-driven training strategies were proposed. These methods do not contribute directly to any MARVEL component, but they constitute tools in the AI methodology design toolkit needed for training deep learning models.

Audio-visual crowd counting aims at estimating the number of people visible in an image or video frame by exploiting enriched audio-visual information. The proposed method is based on multi-exit deep learning architectures efficiently fusing learned visual and audio data representations which allow providing an estimate of the number of people under varying computing resources and network bandwidth allocation. The method targets the AVCC component in the MARVEL architecture and addresses a KPI related to accuracy enhancement (KPI-O2-E2-1). Finally, a cross-modal contrastive learning-based method was proposed to combine multiple types of information to learn audio representation from heterogeneous data. The method was investigated in the context of music information retrieval applications such as musical genre classification, playlist continuation, and automatic tagging. The proposed audio representation learning techniques can be utilised in many audio AI tasks, but they are not directly contributing to any MARVEL component.

References

[1] Francesco Paissan, Alberto Ancilotto, and Elisabetta Farella. PhiNets: a scalable backbone for low-power AI at the edge. *arXiv preprint arXiv:2110.00337*, 2021.

[2] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Improving the accuracy of early exits in multi-exit architectures via curriculum learning. In *International Joint Conference on Neural Networks*, pages 1–8, 2021.

[3] Kateryna Chumachenko, Moncef Gabbouj, and Alexandros Iosifidis. Feedforward neural networks initialization based on discriminant learning. *Neural Networks*, 146:220–229, 2022.

[4] Alexander LeNail. NN-SVG: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019.

[5] Dragana Bajovic and Nikola Simic. MARVEL - D1.3: Architecture definition for MARVEL framework. *Zenodo. https://doi.org/10.5281/zenodo.5463897*, 2021.

[6] Christos Dimou. MARVEL - D5.1: MARVEL Minimum Viable Product. *Zenodo. https://doi.org/10.5281/zenodo.5833310*, 2022.

[7] Toni Heittola and Tuomas Virtanen. MARVEL - D5.2: Technical evaluation and progress against benchmarks – initial version. *Zenodo. https://doi.org/10.5281/zenodo.6322699*, 2022.

[8] Francesco Paissan and Elisabetta Farella Alberto Ancilotto, Alessio Brutti. Scalable neural architectures for end-to-end environmental sound classification. *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[9] Francesco Paissan Alessio Brutti and Elisabetta Farella Alberto Ancilotto. Optimizing phinet architectures for the detection of urban sounds on low-end devices. *EUSIPCO*, 2022.

[10] DCASE website. https://dcase.community/. Accessed: 2022-06-03.

[11] Annamaria Mesaros, Aleksandr Diment, Benjamin Elizalde, Toni Heittola, Emmanuel Vincent, Bhiksha Raj, and Tuomas Virtanen. Sound event detection in the dcase 2017 challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(6):992–1006, 2019.

[12] Nicolas Turpault, Romain Serizel, Ankit Parag Shah, and Justin 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.

[13] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. 1986.

[14] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 131–135, 2017.

[15] Karol J. Piczak. Environmental sound classification with convolutional neural networks. *International Workshop on Machine Learning for Signal Processing*, pages 1–6, 2015.

[16] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[17] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *IEEE Spoken Language Technology Workshop*, 2018.

[18] Pablo Zinemanas, Pabo Cancela, and Martín Rocamora. End-to-end convolutional neural networks for sound event detection in urban environments. In *Conference of Open Innovations Association*, 2019.

[19] Yuji Tokozume and Tatsuya Harada. Learning environmental sounds with end-to-end convolutional neural network. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2721–2725, 2017.

[20] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. Audioclip: Extending clip to image, text and audio. *arXiv:2106.13043*, 2021.

[21] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. Esresne (x) t-fbsp: Learning robust time-frequency transformation of audio. *arXiv:2104.11587*, 2021.

[22] J. Salamon, C. Jacoby, and J. P. Bello. A dataset and taxonomy for urban sound research. In *ACM International Conference on Multimedia*, pages 1041–1044, 2014.

[23] David W Romero, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. Wavelet networks: Scale equivariant learning from raw waveforms. *arXiv:2006.05259*, 2020.

[24] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. Hello edge: Keyword spotting on microcontrollers. *arXiv:1711.07128*, 2017.

[25] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril. Efficient keyword spotting using dilated convolutions and gating. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6351–6355, 2019.

[26] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella. Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):654–664, 2020.

[27] Konstantinos Drossos, Stylianos I Mimilakis, Shayan Gharib, Yanxiong Li, and Tuomas Virtanen. Sound event detection with depthwise separable and dilated convolutions. In *International Joint Conference on Neural Networks*, pages 1–7, 2020.

[28] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[29] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7308–7316, 2019.

[30] Gianmarco Cerutti, Renzo Andri, Lukas Cavigelli, Elisabetta Farella, Michele Magno, and Luca Benini. Sound event detection with binary neural networks on tightly power-constrained iot devices. In *ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 19–24, 2020.

[31] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

[32] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *International Conference on Neural Information Processing Systems*, pages 6391–6401, 2018.

[33] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella. Neural Network Distillation on IoT Platforms for Sound Event Detection. In *Annual Conference of the International Speech Communication Association*, pages 3609–3613, 2019.

[34] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D. Plumbley. Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, 38:67–83, 2021.

[35] Toni Heittola. *Computational Audio Content Analysis in Everyday Environments*. PhD thesis, Tampere University, 6 2021.

[36] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020.

[37] Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE Transactions on Audio, Speech and Language Processing: Special issue on Sound Scene and Event Analysis*, 25(6):1291–1303, 2017.

[38] E. Çakir and T. Virtanen. End-to-end polyphonic sound event detection using convolutional recurrent neural networks with learned time-frequency representation input. In *International Joint Conference on Neural Networks*, pages 1–7, 2018.

[39] J. Salamon and J. P. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[40] Archontis Politis, Sharath Adavanne, and Tuomas Virtanen. A dataset of reverberant spatial sound scenes with moving sources for sound event localization and detection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, pages 165–169, Tokyo, Japan, November 2020.

[41] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *Annual Conference of the International Speech Communication Association*, pages 2613–2617, 2019.

[42] G. Parascandolo, H. Huttunen, and T. Virtanen. Recurrent neural networks for polyphonic sound event detection in real life recordings. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6440–6444, 2016.

[43] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[44] Matthias Dorfer and Gerhard Widmer. Training general-purpose audio tagging networks with noisy labels and iterative self-verification. In *roceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop*, pages 178–182, 2018.

[45] J. Cramer, H. Wu, J. Salamon, and J. P. Bello. Look, listen, and learn more: Design choices for deep audio embeddings. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3852–3856, 2019.

[46] S. Jung, J. Park, and S. Lee. Polyphonic sound event detection using convolutional bidirectional LSTM and synthetic data-based transfer learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 885–889, 2019.

[47] Anurag Kumar and Bhiksha Raj. Weakly supervised scalable audio content analysis. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2016.

[48] Yong Xu, Qiuqiang Kong, Wenwu Wang, and Mark 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)*, pages 121–125, 2018.

[49] Pablo Zinemanas, Pablo Cancela, and Martín Rocamora. Mavd: A dataset for sound event detection in urban environments. In *Detection and Classification of Acoustic Scenes and Events 2019 Workshop*, pages 263–267, 2019.

[50] Jakob Abeßer, Saichand Gourishetti, András Kátai, Tobias Clauß, Prachi Sharma, and Judith Liebetrau. Idmt-traffic: An open benchmark dataset for acoustic traffic monitoring research. In *European Signal Processing Conference*, pages 551–555, 2021.

[51] Justin Salamon, Duncan MacConnell, Mark Cartwright, Peter Li, and Juan Pablo Bello. Scaper: A library for soundscape synthesis and augmentation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 344–348, 2017.

[52] Konstantinos Drossos, Stylianos I Mimilakis, Shayan Gharib, Yanxiong Li, and Tuomas Virtanen. Sound event detection with depthwise separable and dilated convolutions. In *International Joint Conference on Neural Networks*, pages 1–7, 2020.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[55] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162, 2016.

[56] Sharath Adavanne, Archontis Politis, Joonas Nikunen, and Tuomas Virtanen. Sound Event Localization and Detection of Overlapping Sources Using Convolutional Recurrent Neural Networks. *IEEE Journal of Selected Topics in Signal Processing*, 13:34–48, 2018.

[57] Archontis Politis, Annamaria Mesaros, Sharath Adavanne, Toni Heittola, and Tuomas Virtanen. verview and evaluation of sound event localization and detection in DCASE 2019. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:684–698, 2021.

[58] Wolfgang Mack, Ullas Bharadwaj, Soumitro Chakrabarty, and Emanuël AP Habets. Signal-aware broadband DOA estimation using attention mechanisms. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4930–4934, Barcelona, Spain, 2020.

[59] Giuseppe Valenzise, Luigi Gerosa, Marco Tagliasacchi, Fabio Antonacci, and Augusto Sarti. Scream and gunshot detection and localization for audio-surveillance systems. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 21–26, 2007.

[60] Parthasaarathy Ariyakulam Sudarsanam, Archontis Politis, and Konstantinos Drossos. Assessment of self-attention on learned features for sound event localization and detection. In *Detection and Classification of Acoustic Scenes and Events 2021 Workshop*, pages 100–104, 2021.

[61] Yin Cao, Turab Iqbal, Qiuqiang Kong, Yue Zhong, Wenwu Wang, and Mark D. Plumbley. Event-independent network for polyphonic sound event localization and detection. In *etection and Classification of Acoustic Scenes and Events 2020 Workshop*, pages 1–15, 2020.

[62] Sooyoung Park. Trellisnet-based architecture for sound event localization and detection with reassembly learning. In *Detection and Classification of Acoustic Scenes and Events 2019 Workshop*, pages 179–183, 2019.

[63] Kazuki Shimada, Yuichiro Koyama, Naoya Takahashi, Shusuke Takahashi, and Yuki 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*, pages 915–919, 2021.

[64] Yin Cao, Qiuqiang Kong, Turab Iqbal, Fengyan An, Wenwu Wang, and Mark Plumbley. Polyphonic sound event detection and localization using a two-stage strategy. In *Detection and Classification of Acoustic Scenes and Events 2019 Workshop*, pages 30–34, 2019.

[65] Thi Ngoc Tho Nguyen, Douglas L. Jones, and Woon-Seng Gan. A sequence matching network for polyphonic sound event localization and detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 71–75, 2020.

[66] Huy Phan, Lam Pham, Philipp Koch, Ngoc Q. K. Duong, Ian McLoughlin, and Alfred Mertins. On multitask loss function for audio event detection and localization. In *Detection and Classification of Acoustic Scenes and Events 2020 Workshop*, pages 160–164, 2020.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[68] Koichi Miyazaki, Tatsuya Komatsu, Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, and Kazuya Takeda. Conformer-based sound event detection with semi-supervised learning and data augmentation. In *Detection and Classification of Acoustic Scenes and Events 2020 Workshop*, pages 100–104, 2020.

[69] Christopher Schymura, Tsubasa Ochiai, Marc Delcroix, Keisuke Kinoshita, Tomohiro Nakatani, Shoko Araki, and Dorothea Kolossa. Exploiting attention-based sequence-to-sequence architectures for sound event localization. In *European Signal Processing Conference*, pages 231–235, 2020.

[70] Christopher Schymura, Benedikt T. Bönninghoff, Tsubasa Ochiai, Marc Delcroix, Keisuke Kinoshita, Tomohiro Nakatani, Shoko Araki, and Dorothea Kolossa. PILOT: introducing transformers for probabilistic sound event localization. In *Annual Conference of the International Speech Communication Association*, pages 2117–2121, 2021.

[71] Archontis Politis, Sharath Adavanne, Daniel Krause, Antoine Deleforge, Prerak Srivastava, and Tuomas Virtanen. A dataset of dynamic reverberant sound scenes with directional interferers for sound event localization and detection. In *Detection and Classification of Acoustic Scenes and Events 2021 Workshop*, pages 125–129, 2021.

[72] An Tran, Konstantinos Drossos, and Tuomas Virtanen. Wavetransformer: An architecture for audio captioning based on learning temporal and time-frequency information. In *European Signal Processing Conference*, pages 576–580, 2021.

[73] Jan Berg and Konstantinos Drossos. Continual learning for automated audio captioning using the learning without forgetting approach. In *Detection and Classification of Acoustic Scenes and Events 2021 Workshop*, pages 140–144, 2021.

[74] Mengyue Wu, Heinrich Dinkel, and Kai Yu 0004. Audio caption: Listen and tell. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 830–834, 2019.

[75] Khoa Nguyen, Konstantinos Drossos, and Tuomas Virtanen. Temporal subsampling of audio feature sequences for automated audio captioning. In *Detection and Classification of Acoustic Scenes and Events 2020 Workshop*, pages 110–114, 2020.

[76] Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. Clotho: An audio captioning dataset. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 736–740, 2020.

[77] Yuma Koizumi, Ryo Masumura, Kyosuke Nishida, Masahiro Yasuda, and Shoichiro Saito. A Transformer-Based Audio Captioning Model with Keyword Estimation. In *Annual Conference of the International Speech Communication Association*, pages 1977–1981, 2020.

[78] Anna Shi. Audio captioning with the transformer. Technical report, DCASE2020 Challenge, 2020.

[79] 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*, pages 374–378, 2017.

[80] Emre Çakır, Konstantinos Drossos, and Tuomas 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)*, pages 6–10, 2020.

[81] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.

[82] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398, 2016.

[83] Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. Improved image captioning via policy gradient optimization of spider. In *IEEE international conference on computer vision*, pages 873–881, 2017.

[84] Daiki Takeuchi, Yuma Koizumi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. Effects of word-frequency based pre- and post- processings for audio captioning. In *Detection and Classification of Acoustic Scenes and Events 2020 Workshop*, pages 190–194, 2020.

[85] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. AudioCaps: Generating captions for audios in the wild. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 119–132, 2019.

[86] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

[87] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network (2015). *arXiv:1503.02531*, 2, 2015.

[88] Einari Vaaras, Sari Ahlqvist-Bjorkroth, Konstantinos Drossos, and Okko Rasanen. Automatic analysis of the emotional content of speech in daylong child-centered recordings from a neonatal intensive care unit. In *Annual Conference of the International Speech Communication Association*, pages 3380–3384, 2021.

[89] Bjorn Schuller, Bogdan Vlasenko, Florian Eyben, Martin Wöllmer, Andre Stuhlsatz, Andreas Wendemuth, and Gerhard Rigoll. Cross-corpus acoustic emotion recognition: Variances and strategies. *IEEE Transactions on Affective Computing*, 1(2):119–131, 2010.

[90] Jun Deng, Xinzhou Xu, Zixing Zhang, Sascha Frühholz, and Björn Schuller. Semisupervised autoencoders for speech emotion recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(1):31–43, 2017.

[91] Mohammed Abdelwahab and Carlos Busso. Active learning for speech emotion recognition using deep neural network. In *IEEE International Conference on Affective Computing and Intelligent Interaction*, pages 1–7, 2019.

[92] Hesam Sagha, Jun Deng, Maryna Gavryukova, Jing Han, and Björn Schuller. Cross lingual speech emotion recognition using canonical correlation analysis on principal component subspace. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5800–5804, 2016.

[93] Jun Deng, Xinzhou Xu, Zixing Zhang, Sascha Frühholz, and Björn Schuller. Universum autoencoder-based domain adaptation for speech emotion recognition. *IEEE Signal Processing Letters*, 24(4):500–504, 2017.

[94] Siddique Latif, Junaid Qadir, and Muhammad Bilal. Unsupervised adversarial domain adaptation for cross-lingual speech emotion recognition. In *International Conference on Affective Computing and Intelligent Interaction*, pages 732–737, 2019.

[95] Ziping Zhao and Xirong Ma. Active learning for speech emotion recognition using conditional random fields. In *ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 127–131, 2013.

[96] Jia Jia, Suping Zhou, Yufeng Yin, Boya Wu, Wei Chen, Fanbo Meng, and Yanfeng Wang. Inferring emotions from large-scale internet voice data. *IEEE Transactions on Multimedia*, 21(7):1853–1866, 2018.

[97] Weiquan Fan, Xiangmin Xu, Xiaofen Xing, Weidong Chen, and Dongyan Huang. Lssed: a large-scale dataset and benchmark for speech emotion recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 641–645, 2021.

[98] Zhao Shuyang, Toni Heittola, and Tuomas Virtanen. Active learning for sound event classification by clustering unlabeled data. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 751–755, 2017.

[99] Konstantinos Drossos, Paul Magron, and Tuomas Virtanen. Unsupervised adversarial domain adaptation based on the wasserstein distance for acoustic scene classification. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 259–263, 2019.

[100] emmanouel rovithis, nikolaos moustakas, konstantinos vogklis, konstantinos drossos, and andreas floros. design recommendations for a collaborative game of bird call recognition based on internet of sound practices. *Journal of the Audio Engineering Society*, 69(12):956–966, 2021.

[101] Rick Bonney, Jennifer L. Shirk, Tina B. Phillips, Andrea Wiggins, Heidi L. Ballard, Abraham J. Miller-Rushing, and Julia K. Parrish. Next steps for citizen science. *Science*, 343:1436–1437, 2014.

[102] Cathy C Conrad and Krista G Hilchey. A review of citizen science and community-based environmental monitoring: issues and opportunities. *Environmental monitoring and assessment*, 176(1):273–291, 2011.

[103] Jan L Plass, Bruce D Homer, and Charles K Kinzer. Foundations of game-based learning. *Educational Psychologist*, 50(4):258–283, 2015.

[104] Luca Turchet, Carlo Fischione, Georg Essl, Damián Keller, and Mathieu Barthet. Internet of musical things: Vision and challenges. *IEEE Access*, 6:61994–62017, 2018.

[105] Brian L Sullivan, Christopher L Wood, Marshall J Iliff, Rick E Bonney, Daniel Fink, and Steve Kelling. ebird: A citizen-based bird observation network in the biological sciences. *Biological Conservation*, 142(10):2282–2292, 2009.

[106] Luca Turchet, György Fazekas, Mathieu Lagrange, Hossein S Ghadikolaei, and Carlo Fischione. The internet of audio things: State of the art, vision, and challenges. *IEEE Internet of Things Journal*, 7(10):10233–10249, 2020.

[107] Vishwanath A. Sindagi and Vishal M. Patel. A survey of recent advances in CNN-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107(5):3–16, 2018.

[108] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Multi-exit vision transformer for dynamic inference. *British Machine Vision Conference*, 2021.

[109] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[110] Dingkang Liang, Xiwu Chen, Wei Xu, Yu Zhou, and Xiang Bai. Transcrowd: Weakly-supervised crowd counting with transformer. *arXiv preprint arXiv:2104.09116*, 2021.

[111] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. In *Advances in Neural Information Processing Systems*, 2021.

[112] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021.

[113] Hyunjong Park, Jongyoun Noh, and Bumsub Ham. Learning memory-guided normality for anomaly detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 14360–14369, 2020.

[114] Yeara Kozlov and Tino Weinkauf. Extracting and filtering minima and maxima of 1d functions. In *https://www.csc.kth.se/ weinkauf/notes/persistence1d.html (accessed: 11.05.2022)*.

[115] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[116] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008.

[117] Thomas Germer, Tobias Uelwer, Stefan Conrad, and Stefan Harmeling. Fast multi-level foreground estimation. In *International Conference on Pattern Recognition*, 2020.

[118] Alexander Gorbalenya, Susan Baker, Ralph Baric, Raoul de Groot, Christian Drosten, Anastasia Gulyaeva, Bart Haagmans, Chris Lauber, Andrey Leontovich, Benjamin Neuman, Dmitry Penzar, Stanley Perlman, Leo Poon, Dmitry Samborskiy, Igor Sidorov, Isabel Sola, and John Ziebuhr. The species severe acute respiratory syndrome-related coronavirus: classifying 2019-ncov and naming it sars-cov-2. *Nature Microbiology*, 5, 03 2020.

[119] Mert Seker, Anssi Mannisto, Alexandros Iosifidis, and Jenni Raitoharju. Automatic social distance estimation from images: Performance evaluation, test benchmark, and algorithm. *10.5281/zenodo.6737051*, pages 1–14, 2021.

[120] Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal. Monitoring covid-19 social distancing with person detection and tracking via fine-tuned yolo v3 and deepsort techniques, 2020.

[121] Maya Aghaei, Matteo Bustreo, Yiming Wang, Gian Luca Bailo, Pietro Morerio, and Alessio Del Bue. Single image human proxemics estimation for visual social distancing. In *IEEE Winter Conference on Applications of Computer Vision*, 2021.

[122] Dongfang Yang, Ekim Yurtsever, Vishnu Renganathan, Keith A. Redmill, and Ümit Özgüner. A vision-based social distancing and critical density detection system for covid-19. *arXiv:2007.03578*, 2020.

[123] Hong-Yuan Mark Liao Alexey Bochkovskiy, Chien-Yao Wang. Yolov4: Optimal speed and accuracy of object detection. *arXiv:2004.10934*, 2020.

[124] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv:1506.01497*, 2016.

[125] Lorenzo Bertoni, Sven Kreiss, and Alexandre Alahi. Perceiving humans: From monocular 3d localization to social distancing. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–18, 2021.

[126] Pietro Morerio, Matteo Bustreo, Yiming Wang, and Alessio Del Bue. End-to-end pairwise human proxemics from uncalibrated single images. In *IEEE International Conference on Image Processing*, pages 3058–3062, 2021.

[127] Imran Ahmed, Misbah Ahmad, Joel J.P.C. Rodrigues, Gwanggil Jeon, and Sadia Din. A deep learning-based social distance monitoring framework for covid-19. *Sustainable Cities and Society*, 65:102571, 2021.

[128] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018.

[129] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[130] Kathryn Watson. What's an Average Shoulder Width? [Online]. Available: https://www.healthline.com/health/average-shoulder-width [Accessed: 03- Mar- 2021].

[131] Laura Evans. What is pupillary distance and how do you measure it? [Online]. Available: https://www.allaboutvision.com/eye-care/measure-pupillary-distance/ [Accessed: 03- Mar- 2021].

[132] Backpack fitting. [Online]. Available: https://www.whitemountain.com.au/backpack-fitting/backpack-fitting-measure-torso-length.html [Accessed: 03- Mar- 2021].

[133] P. Sturm. Pinhole camera model. In *Computer Vision, A Reference Guide*, 2014.

[134] Katsushi Ikeuchi, editor. *Camera Parameters (Internal, External)*, pages 81–81. Boston, MA, 2014.

[135] S. Kabanikhin, N Tikhonov, V Ivanov, and M Lavrentiev. Definitions and examples of inverse and ill-posed problems. *Journal of Inverse and Ill-posed Problems*, 16:317–357, 2008.

[136] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Single-layer vision transformers for more accurate early exits with less overhead. *10.5281/zenodo.6737408*, 2021.

[137] X Wang, Y Chen, and W Zhu. A comprehensive survey on curriculum learning. *arXiv:2010.13166*, 2020.

[138] Y Bengio, J Louradour, R Collobert, and J Weston. Curriculum learning. *International Conference on Machine Learning*, 2009.

[139] G Hacohen and D Weinshall. On the power of curriculum learning in training deep networks. *International Conference on Machine Learning*, 2019.

[140] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *International Joint Conference on Neural Networks*, pages 1–8, 2018.

[141] C Szegedy, V Vanhoucke, S Ioffe, J Shlens, and Z Wojna. Rethinking the inception architecture for computer vision. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[142] M Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, 2019.

[143] J Deng, W Dong, R Socher, L Li, K Li, and L Fei-Fei. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[144] N Srivastava, A Hinton, A Krizhevsky, I Sutskever, and R Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 5:1929—-1958, 2014.

[145] G Huang, Z Liu, L Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[146] A. G. Howard, M Zhu, B Chen, D Kalenichenko, W Wang, T. Weyand, M Andreetto, and H Adam. Mobilenets: efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

[147] K He, X Zhang, S Ren, and J Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[148] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[149] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[150] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[151] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision*, 2015.

[152] M.T. Rosenstein, Z. Marx, L.P. Kaelbling, and T.G. Dietterich. To transfer or not to transfer. In *Neural Information Processing Workshop on Transfer Learning*, 2005.

[153] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *International Conference on Computer Vision and Pattern Recognition*, 2009.

[154] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *International Conference on Computer Vision*, 2019.

[155] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. 2012.

[156] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[157] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 2627–2635, 2014.

[158] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24(12):5017–5032, 2015.

[159] Yukun Ge, Jiani Hu, and Weihong Deng. Pca-ldanet: A simple feature learning method for image classification. In *Asian Conference on Pattern Recognition*, pages 370–375, 2017.

[160] Michele Alberti, Mathias Seuret, Vinaychandran Pondenkandath, Rolf Ingold, and Marcus Liwicki. Historical document image segmentation with lda-initialized deep neural networks. In *International Workshop on Historical Document Imaging and Processing*, pages 95–100, 2017.

[161] M. Seuret, M. Alberti, M. Liwicki, and R. Ingold. Pca-initialized deep neural networks applied to document image analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 877–882, 2017.

[162] M. Zhu and A. Martinez. Subclass discriminant analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28, 2006.

[163] Kateryna Chumachenko, Jenni Raitoharju, Alexandros Iosifidis, and Moncef Gabbouj. Speed-up and multi-view extensions to subclass discriminant analysis. *Pattern Recognition*, 111(107660):1–15, 2020.

[164] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[165] G Chaladze and L Kalatozishvili. Linnaeus 5 dataset for machine learning. Technical report, 2017.

[166] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[167] Firas Laakom, Jenni Raitoharju, Alexandros Iosifidis, and Moncef Gabbouj. Within-layer diversity reduces generalization gap. In *ICML Workshop on Information Theoretic Methods for Rigorous, Responsible and Reliable Machine Learning*, 2021.

[168] Ian Goodfellow, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT Press, 2016.

[169] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2018.

[170] Vaishnavh Nagarajan and J Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 11615–11626, 2019.

[171] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. Theory of deep learning III: explaining the non-overfitting puzzle. *arXiv preprint arXiv:1801.00173*, 2017.

[172] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[173] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv:1710.10686*, 2017.

[174] Alberto Bietti, Grégoire Mialon, Dexiong Chen, and Julien Mairal. A kernel perspective for regularizing deep neural networks. In *International Conference on Machine Learning*, pages 664–674, 2019.

[175] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pages 7413–7424, 2019.

[176] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[177] Haotian Wang, Wenjing Yang, Zhenyu Zhao, Tingjin Luo, Ji Wang, and Yuhua Tang. Rademacher dropout: An adaptive dropout for deep neural network via optimizing generalization gap. *Neurocomputing*, pages 177–187, 2019.

[178] Hae Beom Lee, Taewook Nam, Eunho Yang, and Sung Ju Hwang. Meta dropout: Learning to perturb latent features for generalization. In *International Conference on Learning Representations*, 2019.

[179] Zhe Li, Boqing Gong, and Tianbao Yang. Improved dropout for shallow and deep learning. In *Advances in Neural Information Processing Systems*, pages 2523–2531, 2016.

[180] Nan Li, Yang Yu, and Zhi-Hua Zhou. Diversity regularized ensemble pruning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 330–345, 2012.

[181] Yang Yu, Yu-Feng Li, and Zhi-Hua Zhou. Diversity regularized machine. In *International Joint Conference on Artificial Intelligence*, 2011.

[182] Bo Xie, Yingyu Liang, and Le Song. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics*, pages 1216–1224, 2017.

[183] Pengtao Xie, Yuntian Deng, and Eric Xing. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv:1511.07110*, 2015.

[184] Ke Yang, Vasilis Gkatzelis, and Julia Stoyanovich. Balanced ranking with diversity constraints. In *International Joint Conference on Artificial Intelligence*, pages 6035–6042.

[185] Jonathan Malkin and Jeff Bilmes. Multi-layer ratio semi-definite classifiers. In *International Conference on Acoustics, Speech and Signal Processing*, pages 4465–4468, 2009.

[186] Michael Cogswell, Faruk Ahmed, Ross B. Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. In *International Conference on Learning Representations*, 2016.

[187] Alex Kulesza and Ben Taskar. Structured determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2010.

[188] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[189] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[190] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[191] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[192] Di Hu, Lichao Mou, Qingzhong Wang, Junyu Gao, Yuansheng Hua, Dejing Dou, and Xiao Xiang Zhu. Ambient sound helps: Audiovisual crowd counting in extreme conditions. *arXiv:2005.07097*, 2020.

[193] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[194] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[195] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.

[196] Andres Ferraro, Xavier Favory, Konstantinos Drossos, Yuntae Kim, and Dmitry Bogdanov. Enriched music representations with multiple cross-modal contrastive learning. *IEEE Signal Processing Letters*, 28:733–737, 2021.

[197] Minz Won, Sergio Oramas, Oriol Nieto, Fabien Gouyon, and Xavier Serra. Multimodal metric learning for tag-based music retrieval. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 591–595, 2021.

[198] Òscar Celma and Perfecto Herrera. A new approach to evaluating novel recommendations. In *ACM Conference on Recommender Systems*, pages 179–186, 2008.

[199] Filip Korzeniowski, Oriol Nieto, Matthew McCallum, Minz Won, Sergio Oramas, and Erik Schmidt. Mood classification using listening data. In *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, October 11-16, 2020*, pages 542–549, 2020.

[200] Sergio Oramas, Francesco Barbieri, Oriol Nieto Caballero, and Xavier Serra. Multimodal deep learning for music genre classification. *Transactions of the International Society for Music Information Retrieval*, 1(1):4–21, 2018.

[201] Didac Surís, Amanda Duarte, Amaia Salvador, Jordi Torres, and Xavier Giró-i Nieto. Cross-modal embeddings for video and audio retrieval. In *Computer Vision – ECCV 2018 Workshops*, pages 711–716, Cham, 2019. Springer International Publishing.

[202] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR*, 2016.

[203] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

[204] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2):207–244, 2009.

[205] Jeong Choi, Jongpil Lee, Jiyoung Park, and Juhan Nam. Zero-shot learning for audio-based music classification and tagging. In Arthur Flexer, Geoffroy Peeters, Julián Urbano, and Anja Volk, editors, *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, pages 67–74, 2019.

[206] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[207] Eduardo Fonseca, Diego Ortego, Kevin McGuinness, Noel E O'Connor, and Xavier Serra. Unsupervised contrastive learning of sound event representations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 371–375. IEEE, 2021.

[208] Aaqib Saeed, David Grangier, and Neil Zeghidour. Contrastive learning of general-purpose audio representations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3875–3879. IEEE, 2021.

[209] Xavier Favory, Konstantinos Drossos, Tuomas Virtanen, and Xavier Serra. Coala: Co-aligned autoencoders for learning semantically enriched audio representations. In *International Conference on Machine Learning, Workshop on Self-supervision in Audio and Speech (ICML)*, 2020.

[210] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[211] Xavier Favory, Konstantinos Drossos, Tuomas Virtanen, and Xavier Serra. Learning contextual tag embeddings for cross-modal alignment of audio and tags. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 596–600, 2021.

[212] Xiaoqiang Ma, Tai Yao, Menglan Hu, Yan Dong, Wei Liu, Fangxin Wang, and Jiangchuan Liu. A survey on deep learning empowered iot applications. *IEEE Access*, 7:181721–181732, 2019.

[213] Xiaofei Wang, Yiwen Han, Victor C. M. Leung, Dusit Niyato, Xueqiang Yan, and Xu Chen. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22(2):869–904, 2020.

[214] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *arXiv:2103.04505*, 2021.

[215] Arian Bakhtiarnia, Nemanja Milosevic, Qi Zhang, Dragana Bajovic, and Alexandros Iosifidis. Dynamic split computing for efficient edge intelligence. *10.5281/zenodo.6737211*, 2022.

[216] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

[217] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *arXiv:2104.00298*, 2021.

7 Appendix

7.1 Scalable neural architectures for end-to-end environmental sound classification

The appended paper follows.

SCALABLE NEURAL ARCHITECTURES FOR END-TO-END ENVIRONMENTAL SOUND CLASSIFICATION

Francesco Paissan, Alberto Ancilotto*, Alessio Brutti, Elisabetta Farella*

Digital Society (DiGis) center - Fondazione Bruno Kessler

ABSTRACT

Sound Event Detection (SED) is a complex task simulating human ability to recognize what is happening in the surrounding from auditory signals only. This technology is a crucial asset in many applications such as smart cities. Here, urban sounds can be detected and processed by embedded devices in an Internet of Things (IoT) to identify meaningful events for municipalities or law enforcement. However, while current deep learning techniques for SED are effective, they are also resource- and power-hungry, thus not appropriate for pervasive battery-powered devices. In this paper, we propose novel neural architectures based on PhiNets for real-time acoustic event detection on microcontroller units. The proposed models are easily scalable to fit the hardware requirements and can operate both on spectrograms and waveforms. In particular, our architectures achieve state-of-the-art performance on UrbanSound8K in spectrogram classification (around 77%) with extreme compression factors (99.8%) with respect to current state-of-the-art architectures.

Index Terms— sound event detection, tinyML, scalable backbone, IoT;

1. INTRODUCTION

The task of Sound Event Detection (SED) consists in recognizing acoustic events in audio streams. This task is of interest both for industrial and smart cities applications. Although recently the research community has steadily improved the effectiveness and accuracy of SED solutions, current neural networks-based approaches are highly demanding in terms of memory footprint and computational complexity. As a consequence, these systems are not suitable for applications requiring pervasive low-power low-cost sensors. In addition, the high computational complexity affects the lifetime of the devices and results in increased energy absorption and related carbon emissions. Nonetheless, it has already been shown [1] how such architectures are not strictly necessary and can be compressed using, for example, knowledge distillation [2] and network pruning [3]. Moreover, when bringing these approaches to edge devices, it is essential to address the variability in computational resources existing between different platforms. Current compression approaches are generally tailored to a specific hardware platform; thus, they require an expensive process to adapt the neural network to new application scenarios. Conversely, neural architectures should scale efficiently to exploit the available computational resources under different operational constraints.

To address the issues above, in this paper we employ the *PhiNets* [4] architecture's family for the first time in an audio task, showing that these models are good candidates for deep-learning-based multimedia analytics at the edge. In addition, we propose a novel scalable backbone, which resembles the scalability principles of the *PhiNets*, while compressing audio models (down to 766 parameters)[1]. Excluding AudioCLIP [5], which however can not run in real-time on any embedded device (including edge GPUs), the proposed models achieve state-of-the-art performance on the UrbanSound8K benchmark [6], using only a fraction of the parameters and of the computations of current architectures (up to a 99.8% reduction in parameters and operations with respect to similarly performing models).

2. RELATED WORKS

In literature, many architectures for SED exist, recently driven by the DCASE series [7]. This section will review the state-of-the-art techniques grouped by input type (spectrogram or waveform) and target platform.

2.1. SED using spectrograms

The most common approach in detecting acoustic events is spectrogram classification. In this paradigm, the waveform is converted into a spectrogram, using stacked Fourier transforms [8], which are then processed by convolutional neural networks (CNNs). Among the approaches in literature, the best performing are VGGish [9], PiczakCNN [10], and SB-CNN [11]. For the sake of this study, we only consider the aforementioned architectures in terms of their classification accuracy on the Urbansound8K benchmark [6] and in terms of their computational requirements. In particular, these three architectures have different structures, but share a high parameter count, with the smallest one being SB-CNN that counts 241 k parameters. SB-CNN was presented in [11] alongside data augmentation techniques, which proved to be the most effective technique to train networks on the UrbanSound8K benchmark, given the small size of the dataset. Overall, these architectures are constituted by a massive number of parameters that could not fit on off-the-shelf MCUs.

2.2. SED using waveform

An emerging trend in audio classification is exploiting one dimensional convolutions (1DConvs) directly on the waveforms. In this case, neural networks learn the filters to be applied to the input signal directly. Many approaches that inject previous knowledge in the filter shape are proposed to ease the training. In SincNet [12], for example, the filters of the first convolution are forced to be band-pass filters. In [13] instead, the proposed architecture exploits the Gammatone filter initialization to boost the classification performance. In ENVNET-V2 [14], the authors use 1DConvs and then exploit bidimensional convolutions on the feature map. Despite the good performance, this comes with a high cost in computational requirements

*These authors contributed equally to this research. This work was partially funded by the EU H2020 project MARVEL (project number: 957337).

[1]Code available at `https://github.com/fpaissan/phinet_pl`

(more than 1 M parameters). AudioCLIP [5] learns a bi-dimensional representation of the waveform with a custom backbone [15] and is currently the best performing model on the UrbanSound8k benchmark. However, this high accuracy is achieved by employing extremely large architectures, counting up to 30 M parameters only for the feature extraction. In Wavelet Networks [16], instead, the architecture resembles the wavelet transform to maximize the sound event detection performance by reducing the impact of phase shifts in the signal. Overall, models working on the waveform are less accurate in classifying acoustic events, mainly due to the higher variability of the signals in the time domain. On the positive side, the 1DConv based models have a higher parameter efficiency given that they need to run in only one dimension.

2.3. SED at the edge

Audio processing at the edge (i.e. on embedded platforms) is relevant for both research and industrial applications. Many approaches targeting embedded platforms are already available in the literature for a variety of audio tasks, namely keyword spotting (KWS) [17, 18] and SED [1, 19]. In [1], a student-teacher approach is presented for model compression via knowledge distillation based on joint alignment of the latent representations and cost function optimization for classification. This approach shows promising results in compressing architectures. However, there is an implicit upper bound to the network's performance since it is empirically shown that the performance of the student network will not surpass that of the teacher. [19] proposes a novel architecture where a dilated convolution replaces the recurrent unit. Moreover, the implementation exploits depth-wise separable convolutions [20], which are well-known for their parameter efficiency. Despite this, the parameter count of the architecture is still higher than what could fit on an MCU. Network quantization offers another popular approach [21]. The most computationally efficient uses Binary Neural Networks (BNNs) [22] but compromises classification performance.

In our work, we present two efficient architectures whose lite computational complexity allows their implementation on extremely resource-constrained platforms, like MCUs. In addition, the proposed models are trained from scratch and, thus, are not limited by the knowledge distillation process.

3. HARDWARE-AWARE SCALING WITH PHINETS

When bringing neural architectures on MCUs, one of the most efficient approaches is *hardware-aware* scaling [4]. Using this paradigm, it is possible to optimize the neural architectures to fit on embedded platforms with a negligible drop in performance. However, in order to exploit this scaling principle, we need to avoid an exponential decay of the performance with respect to computational requirements, as often occurs [23]. For these reasons, we present two different architectures (*PhiNets 1D* and *PhiNets 2D*) that are in line with the *hardware-aware* scaling paradigm and work on two different multiply–accumulate (MAC) and memory ranges. We exploit the scalability principles of *PhiNets* [4], a scalable backbone based on a sequence of inverted residual blocks (depicted in Fig. 2), where the shape of each block depends on three hyper-parameters α, β, t_0 that control disjointly the MAC count and memory requirements (FLASH and RAM), respectively - as described in [4].

3.1. *PhiNets* on spectrograms

We introduced some modifications to the original *PhiNets* architecture to tailor it to the SED task and improve the classification performance. In particular, we propose down-sampling the feature map using max-pooling instead of strided convolutions. We also replace the original input block with a standard 2D convolution.
Max-pooling improves the network's overall performance by around 5% on the UrbanSound8K dataset, and changing the input block showed a similar trend. We observed that networks under 2 k parameters and 5 M MAC perform better using a strided convolution for downsampling and a depth-wise separable input block. Despite this small change, the computational load of the *PhiNets* architectures does not change and is analytically described in [4].

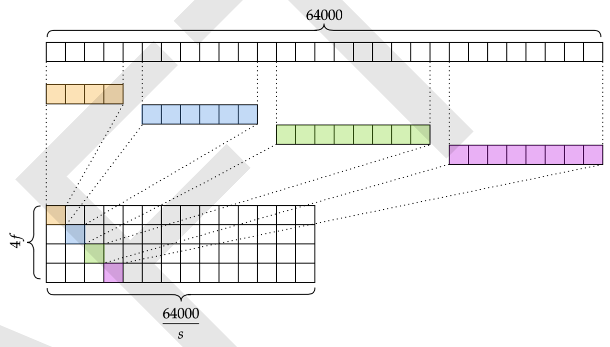


Fig. 1. Illustration of the input 1D convolutional block. The stacked convolutions work on the features, which are time-stretched with different phases. In the illustration, f represents the number of filters, while s represents the stride of the convolution.

3.2. *PhiNets* on waveform

To further reduce the computational cost of the *PhiNets*, we propose a variation of the architecture that works on waveforms, thus exploiting one-dimensional convolutions (1DConvs). By doing this, the relationship between the computational requirements and the number of filters used in each convolutional block is linear instead of quadratic. Moreover, the overall parameter count is lower.

The network architecture is split into three main blocks. The first block is a convolutional block that aims at reducing the shape of the input tensor allowing for a trade-off between accuracy and MAC count. This block consists of four vertically stacked 1DConvs with different kernel sizes (namely 32, 64, 128, 256 points), which work on time-stretched versions of the waveform to avoid losing information in the striding process, as described in Fig. 1. This first convolutional block is followed by a sequence of convolutional blocks composed by a point-wise convolution to up-sample the features, a depth-wise convolution, a squeeze-and-excite block and another point-wise convolution to restore the same number of features as the input. At the end of the network, a fully-connected layer for classification compresses the extracted features and outputs the logits for each class. To decrease the computational complexity and to help the convergence of the network [24], we exploit skip connections in the convolutional blocks. In particular, we either (i) concatenate the input and output tensors to double the number of features used in the following layers (instead of increasing channels by means of e.g. a point-wise convolution) or (ii) sum the input and output tensors.

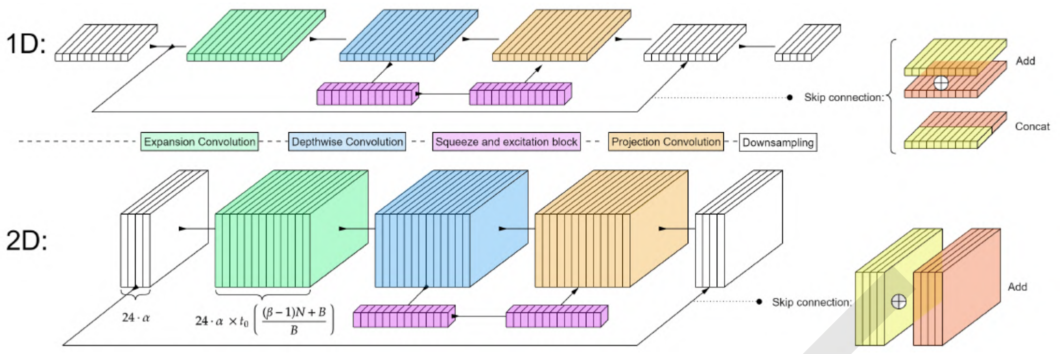


Fig. 2. Illustration of the 1D and 2D convolutional blocks. The input map is fed into the expansion convolution, which affects only the number of channels in the feature map. The feature map is fed into a depth-wise convolution followed by a squeeze-and-excite block. The output of the squeeze-and-excite is projected in a lower dimensionality space via a bottleneck layer. At the end of the 1D block there is an optional downsampling - implemented using average pooling - and a skip connection - either `ADD` or `CONCAT` depending if the layer realizes a downsampling operation or not. For the 2D block, the skip connection always uses an `ADD` operation. In the illustration, B is the number of blocks and N is the ID of the current block.

Fig. 2 shows the convolutional block of the *PhiNets* and how it is performed both in one and two dimensions.

To scale the computational requirements of *PhiNets 1D*, we can change: the number of convolutional blocks, the depth-multiplier, the number of filters in the first convolution or the stride of the input convolution. In particular, changing all of the above parameters has a linear impact on both computational cost (MAC count and parameter count) except for the number of filters in the first convolutional block. The latter has a quadratic impact on both parameter and MAC count. Such scalability features allow for extreme model compression and optimization, while decoupling parameter count and computational cost in alignment with the *harware-aware* scaling paradigm.

4. EXPERIMENTAL SETUP

We benchmarked the two proposed architectures, for waveforms and spectrograms, on the UrbanSound8K dataset [6]. The dataset consists of a collection of 8732 samples of 4 second long typical urban sound events, equally distributed among ten different classes (air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music). The sampling rate of the original audio sample varies, so we re-sampled each event at 16 kHz, resulting in 64 000 timepoints per sample. We used the standard 10-fold benchmarking procedure for this dataset by averaging the test score after training on eight folds and using one for validation. We augmented the dataset with pitch shift, time-stretching, and Gaussian noise. The model input consists of 40 mel-spectrograms computed on the 4 s sample using 2048 sample windows with a hop-length of 512, resulting in 120 frames for each sound event. For the waveform model instead, we used the re-sampled signal, thus leading to a 64 000 entries input vector.

We trained the models on spectrograms for 100 epochs, with a 10^{-3} learning rate, 10^{-2} weight decay and 0.05 dropout rate in the convolutional blocks. Moreover, we also added label smoothing to help the network avoid over-fitting. For the waveform model, we decreased the learning rate starting from 6×10^{-4} every time the validation accuracy was not improving for 15 consecutive epochs. Moreover, we used L2 regularization as for the other approach.

To demonstrate that scaling *PhiNets* has a marginal impact on the classification accuracy with respect to the compression factor, we benchmarked models in the 0.1-20 MMAC range and with 0.7-30 thousand parameters, which is a typical range for real-time operation with off-the-shelf MCUs. For reproducibility, the generated models are enumerated in Table 1.

5. RESULTS

Table 2 reports the performance achieved by the proposed models against a set of state-of-the-art solutions described in Sec. 2.1. We consider different configurations of our PhiNets implementation, which results in different parameter counts as shown in Table 1. The best performing model in spectrogram classification ([14]) has 101 M parameters and achieves 78% classification accuracy. *PhiNets*, instead, achieve a 76.3% accuracy with only 27 k parameters (i.e. 99.8% compression factor). This result pushes the state-of-the-art in SED on tiny architectures with a higher performance-compression ratio. Note that if we do not consider AudioCLIP [5], for the reasons already discussed, *PhiNets* have competitive results also in waveform classification, delivering an overall drop of around 15% in 10-fold accuracy while using only 2% of the best performing model's parameters.

5.1. Impact of scaling on classification accuracy

From the results reported above, it is clear that the two architectures cover well the operating range of MCUs. In fact, when the performance of the model that works on spectrograms starts decreasing drastically, the one-dimensional model helps extend the operative range while keeping the performance a bit higher, as depicted in Fig. 3. Encoding the information presented in the spectrograms is a much more computationally intensive task with respect to waveform analysis. However, this usually comes with an improvement in classification accuracy since spectrograms are less sensitive to noise and phase shift. Also, by scaling the *PhiNets* on spectrograms to a lower MAC and parameter count, we see that the performance is worse than the one of *PhiNets* when working on raw waveforms. In fact, the two models complement each other when scaling computational requirements, as shown in Fig. 3: spectrum yields the best complexity performance trade-off for high-end platforms whereas

Input	Model name	Input conv type	Max pooling	α	B	t_0	**MMAC**	**Parameters (k)**
Spectrogram	PhiNets M_{40}	Conv2D	True	0.5	3	4.0	43.00	27.1
	PhiNets M_{15}	SeparableConv2D	True	0.5	2	5.0	14.43	32.3
	PhiNets M_5	Conv2D	True	0.2	2	2.0	4.72	3.80
	PhiNets M_3	Conv2D	True	0.1	2	4.0	2.71	2.18
	PhiNets $M_{1.5}$	SeparableConv2D	True	0.1	2	2.0	1.59	2.00
		Input conv stride	Input conv filters	n	d		**MMAC**	**Parameters (k)**
Waveform	PhiNets 1D M_1	300	3	4	4.5		1.34	11.50
	PhiNets 1D $M_{0.5}$	500	2	4	4.5		0.40	5.91
	PhiNets 1D $M_{0.2}$	1600	2	3	2.5		0.06	2.11
	PhiNets 1D $M_{0.1}$	300	1	4	1.5		0.15	1.15
	PhiNets 1D $M_{0.07}$	500	1	3	1.5		0.07	0.766

Table 1. Parameters for generating the architectures presented in this paper. For the spectrogram classification model, the notation is the same as in [4]. For the waveform model, d refers to the depth multiplier while n refers to the number of blocks.

Input	Model	Params (K)	10-fold acc
Spectrogram	PICZAKCNN [10]	26 000	73.7
	SB-CNN [11]	241	73.11
	VGG [9]	77 000	70.74
	Cerutti M_{20k} [25]	30	69
	Cerutti M_{200k} [25]	200	72
	Cerutti M_{2M} [25]	2 000	75
	Cerutti M_{20M} [25]	70 000	76
	PhiNets M40	27.1	**76.3** ± 5.6
	PhiNets M15	32.2	76.1± 5.0
	PhiNets M5	3.80	68.8± 3.1
	PhiNets M3	2.18	65.3± 1.6
	PhiNets M1.5	2.00	62.3± 3.9

Input	Model	Params (K)	10-fold acc
Waveform	AudioCLIP [5]	>30 000	90.01
	ENVNET-V2 [14]	101 000	78
	W11-NET-WL [16]	1 806	68.47± 4.914
	W18-NET-WL [16]	3 759	65.01 ± 5.431
	W34-NET-WL [16]	4 021	66.77± 4.771
	1DCNN [13]	453	62± 6.791
	W-1DCNN-WL [16]	458	62.64± 4.979
	PhiNets 1D M1	**11.5**	59.3± 3.7
	PhiNets 1D M0.5	5.91	56.4± 6.4
	PhiNets 1D M0.2	2.11	48.4± 2.5
	PhiNets 1D M0.1	1.15	46.3± 4.2
	PhiNets 1D M0.07	0.766	43.3± 2.6

Table 2. Comparison between *PhiNets* and other state-of-the-art architectures, considering spectrograms and waveforms as input features. The central column reports the parameter count. The 10-fold accuracy is taken from the original papers. When standard deviation is not available in the paper, it is not reported in the Table. The notation for the models is taken from the original papers. In particular, M_{20k} in [25] refers to the order of magnitude of the parameter count.

using waveforms as input works better if computational constraints are more stringent.

As a rule of thumb, we saw that to maximize performance, the best combination of parameters involved the use of pooling layers, input block composed of a 2D convolution and a base expansion factor between 4 and 6. Smaller models, instead, performed better with a depth-wise separable input block as described in the original paper [4], strided convolutions for downsampling and a t_0 between 2 and 3. Instead, for the waveform model the test accuracy increases linearly with the depth-multiplier and decreases linearly with stride. Therefore, as expected, the best-performing models have a high number of blocks and a low stride; thus, they compress less the information in the input waveform.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented two novel architectures for SED at the edge. Our architectures are the most efficient in computational complexity for spectrogram classification without compromising the classification accuracy. In fact, the best performing model, which is also the biggest we benchmarked in the study, has only 27 k parameters and 43 MMAC and thus can easily fit in an MCU. Moreover, we studied the scalability features of our models in order to validate which architectures are the best performing ones for varying computational requirements. We highlight that *hardware-aware* scaling is the most computationally efficient way of bringing neural networks on MCUs (i.e., extremely low-resource devices). We plan to extend these architectures to other audio tasks, namely keyword spotting and speech recognition and to other training paradigms (e.g. cross-modal, knowledge distillation). Moreover, we will expand the 1DConv model with different input convolutional layers shapes (e.g., Sinc, Wavelet) to boost the models' performance.

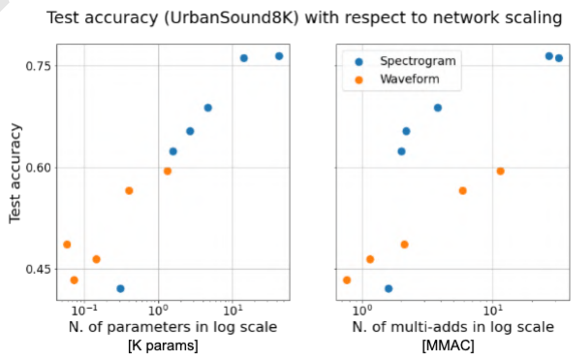


Fig. 3. Classification performance with respect to computational requirements and input type.

7. REFERENCES

[1] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella, "Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 654–664, 2020.

[2] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[3] Lorenzo Valerio, Franco Maria Nardini, Andrea Passarella, and Raffaele Perego, "Dynamic hard pruning of neural networks at the edge of the internet," *arXiv preprint arXiv:2011.08545*, 2020.

[4] Francesco Paissan, Alberto Ancilotto, and Elisabetta Farella, "PhiNets: a scalable backbone for low-power AI at the edge," *arXiv preprint arXiv:2110.00337*, 2021.

[5] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel, "Audioclip: Extending clip to image, text and audio," *arXiv preprint arXiv:2106.13043*, 2021.

[6] 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)*, Orlando, FL, USA, Nov. 2014, pp. 1041–1044.

[7] Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen, "Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions," *arXiv preprint arXiv:2005.14623*, 2020.

[8] Ronald Newbold Bracewell and Ronald N Bracewell, *The Fourier transform and its applications*, vol. 31999, McGraw-Hill New York, 1986.

[9] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan 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.

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

[11] Justin Salamon and Juan Pablo Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[12] Mirco Ravanelli and Yoshua Bengio, "Speaker recognition from raw waveform with sincnet," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 1021–1028.

[13] Pablo Zinemanas, Pabo Cancela, and Martín Rocamora, "End-to-end convolutional neural networks for sound event detection in urban environments," 04 2019.

[14] Yuji Tokozume and Tatsuya Harada, "Learning environmental sounds with end-to-end convolutional neural network," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2721–2725, 2017.

[15] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel, "Esresne (x) t-fbsp: Learning robust time-frequency transformation of audio," *arXiv preprint arXiv:2104.11587*, 2021.

[16] David W Romero, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn, "Wavelet networks: Scale equivariant learning from raw waveforms," *arXiv preprint arXiv:2006.05259*, 2020.

[17] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[18] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6351–6355.

[19] Konstantinos Drossos, Stylianos I Mimilakis, Shayan Gharib, Yanxiong Li, and Tuomas Virtanen, "Sound event detection with depthwise separable and dilated convolutions," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[20] François Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[21] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua, "Quantization networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7308–7316.

[22] Gianmarco Cerutti, Renzo Andri, Lukas Cavigelli, Elisabetta Farella, Michele Magno, and Luca Benini, "Sound event detection with binary neural networks on tightly power-constrained iot devices," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, pp. 19–24.

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

[24] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the loss landscape of neural nets," in *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2018, pp. 6391–6401.

[25] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella, "Neural Network Distillation on IoT Platforms for Sound Event Detection," in *Proc. Interspeech 2019*, 2019, pp. 3609–3613.

7.2 Optimizing PhiNet architectures for the detection of urban sounds on low-end devices

The appended paper follows.

Optimizing PhiNet architectures for the detection of urban sounds on low-end devices

Alessio Brutti
Digis Center
Fondazione Bruno Kessler
Trento, Italy
brutti@fbk.eu

Francesco Paissan
Digis Center
Fondazione Bruno Kessler
Trento, Italy
fpaissan@fbk.eu

Alberto Ancilotto
Digis Center
Fondazione Bruno Kessler
Trento, Italy
aancilotto@fbk.eu

Elisabetta Farella
Digis Center
Fondazione Bruno Kessler
Trento, Italy
efarella@fbk.eu

Abstract—**Sound Event Detection (SED) pipelines identify and classify relevant events in audio streams with application in the smart city domain (e.g., crowd counting, alarm triggering), thus is an asset for municipalities and law enforcement agencies. Given the large size of the areas to be monitored and the amount of data generated by the IoT sensors, large models running on centralised servers are not suitable for real-time applications. Conversely, performing SED directly on pervasive embedded devices is very attractive in terms of energy consumption, bandwidth requirements and privacy preservation. In a previous manuscript, we proposed scalable backbones from the PhiNets architectures' family for real-time sound event detection on microcontrollers. In this paper, we extend our analysis investigating how PhiNets' parameters scaling affects the model performance in the SED task while searching for the best configuration given the computational constraints.Experimental analysis on UrbanSound8K shows that while only the total number of parameters matters when training the model from scratch (i.e., it is independent of the scaling parameter configuration), knowledge distillation is more effective with specific scaling configurations.**

Index Terms—**Sound event detection, Neural Networks, PhiNets, tinyML**

I. Introduction

Sound Event Detection (SED) is an emerging task with many applications in fields like industries and intelligent cities [1], where multimedia analytics gained significant interest in the recent past [2], [3]. SED can benefit from the availability of pervasive embedded devices capable of continuously monitoring the environment looking for relevant events [1]. Driven by the release of novel datasets and challenges [4]–[7], recent advancements in the field have considerably improved the effectiveness and accuracy of SED solutions. However, this has been achieved using highly demanding models in terms of memory footprint and computational complexity [8]–[13]. Consequently, these systems are not suitable for applications requiring pervasive low-power, low-cost sensors. Nonetheless, it has been shown how strategies such as knowledge distillation (KD) [14], network pruning [15] or weight quantization [16], [17] can considerably reduce the size of the models, making them suitable to run on microcontroller units (MCU) [18]. Unfortunately, these techniques are typically tailored to the specific device and require an expensive process to adapt the original neural network to different processing units. Therefore, efforts have been recently focused on developing architectures specifically designed to operate on low-end devices [19].

Following this line of research, in a previous work [20], we applied *PhiNets* [21] to the SED task either using spectrograms or raw waveforms. The proposed model achieved state-of-the-art performance on the UrbanSound8k dataset [4] for spectrogram classification while using an extremely low number of parameters. The focus of this previous paper was on minimising as much as possible the memory footprint of the models, in order to make it fit on MCUs, while limiting the performance deterioration with respect to the state-of-the-art. Conversely, in this paper, we provide an experimental analysis on how the PhiNet's width-scaling parameters (namely the width multiplier α and the base expansion factor t_0 from the original paper [21]) impact the final classification performance by defining models of different sizes and architectures. In particular, we observed that different configurations of the scaling parameters leading to the same amount of model parameters give very similar performance. On the other hand, applying KD from a larger teacher model boosts the performance but only when large t_0 values are employed.

The paper is organised as follows. Section II describes the PhiNet backbones and their scaling parameters. Section III provides details about the experimental analysis whose results are reported and discussed in Section IV. Finally, Section V concludes the paper with final remarks.

II. Scalable backbones: PhiNets

This work employs the PhiNets networks [21]: a family of modular scalable backbones that can be easily tuned using few hyperparamters to match the memory and computational resources available on different embedded platforms. The main convolutional block used in the architecture is a modified version of the inverted residual block used in MobileNetV2 [22] and MobileNetV3 [23] architectures. This block is composed of a sequence of three operations, namely a pointwise *expansion* convolution, a depthwise convolution, a squeeze-and-excitation block [24] and a second pointwise *projection* convolution. The structure of the basic PhiNet convolutional block is shown in Fig. 1. The final model is obtained stacking

This work was partially funded by the EU H2020 project MARVEL (project number: 957337).

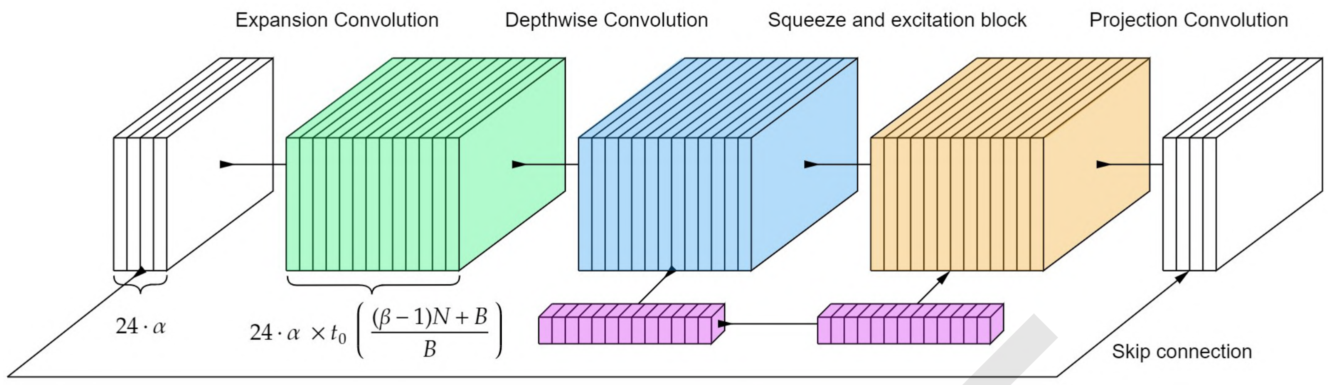

Expansion Convolution
Depthwise Convolution
Squeeze and excitation block
Projection Convolution
$24 \cdot \alpha$
$24 \cdot \alpha \times t_0 \left(\frac{(\beta - 1)N + B}{B} \right)$
Skip connection

Fig. 1. An overview of the PhiNets convolutional block structure. First, the number of channels is increased with a pointwise convolution, followed by a depthwise convolution (green) and SE block (blue). Finally, a second pointwise convolution (yellow) connects to the low dimensionality bottleneck block (purple). B is the number of blocks in the network, and N is the current block index. α, β and t_0 are the block hyperparameters.

B of these blocks (we use $B = 5$ for all the experiments in this work). Three hyperparameters can be used to modify the configuration of the convolutional blocks:

- **Width multiplier** α, which linearly adjusts the filter count of all convolutions in the network. As a result, it scales the operation count of the whole model. The number of operations in the network depends quadratically on this parameter, as shown in Fig. 2;

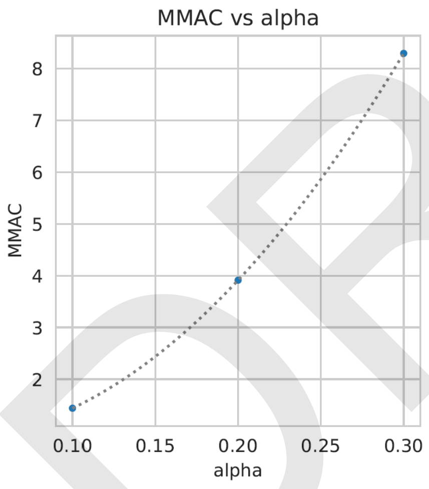


Fig. 2. Effects of varying the hyperparameter α on network operations (expressed in Multiply and Accumulate, MACC).

- **Base expansion factor** t_0, which affects the filter count in the expansion and depthwise convolutions inner blocks. This parameters can be used to optimise the RAM required by the network, which can be approximated as $R \approx C \cdot t_0$ with C denoting the RAM needed for the network with $t_0 = 1$. The effects of this parameter on the network working memory is shown in Fig. 3;
- **Shape parameter** β, that defines the filter count of the later blocks in the networks. These blocks are those the ones requiring the largest number of parameters, which can be approximated as $\#Params \approx C \cdot \frac{1}{2}(1+\beta)$, where C is number of parameters of the network with $\beta = 1$. The effects of this parameter on network parameters are shown in Fig 4.

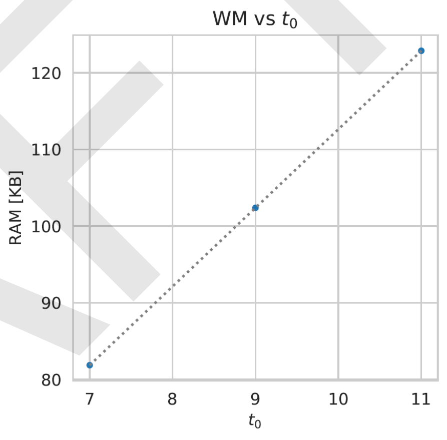


Fig. 3. Effects of varying the hyperparameter t_0 on the working memory (WM) or RAM required to store the intermediate network tensors.

The computational cost and memory footprint of the model can be easily adjusted to fit the constraints of the processing unit by varying these three hyper-parameters. Note that different configurations of the hyper-parameters may lead to highly similar parameter counts but rather different architectures. In this work, we want to investigate the effectiveness of these different configurations. The sequence of blocks is then followed by a fully connected classification layer with softmax activation in order to obtain a 10 class classification output.

III. Experimental Analysis

This work carries out an empirical study to highlight the effects of two width scaling parameters (α and t_0) on sound event detection performance. β will be kept at the default value ($\beta = 1$), as all networks tested require so few parameters that even the smallest MCUs can store them with a considerable

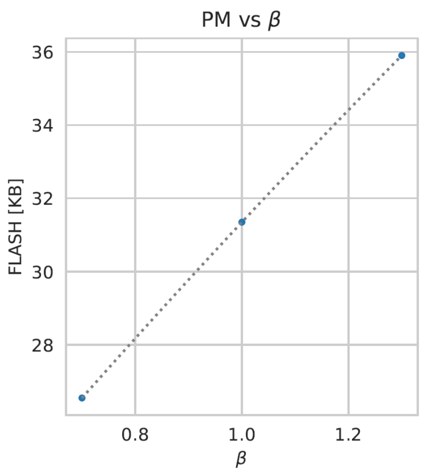


Fig. 4. Effects of varying the hyperparameter β on the parameter memory (PM) or FLASH required to store the network weights.

margin. We vary α considering [0.20, 0.35, 0.50] possible values and t_0 in [2, 4, 6]. Note that in this way, we cover different architectures with a very similar number of parameters. Given the small sizes of the resulting PhiNet models, besides training them from scratch, we also investigate the use of KD from a larger plain-conv2d model, using both soft and one-hot labels.

A. Dataset

We perform our analysis on the UrbanSound8K dataset [4], a collection of 8732 samples of 4 second long typical urban sound events, equally distributed among 10 different classes (air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music). We re-sampled each event at 16 kHz. Moreover, we augmented the dataset with pitch shifting with tone steps -2, -1, 1, 2, time-stretching with factors 0.81 and 1.07, and additional Gaussian noise. The task is single-label classification, and the performance is hence evaluated in terms of classification accuracy. We used the standard 10-fold benchmarking procedure for this dataset.

B. Implementation Details

The model input is 2D and consists of 40 mel-spectrum features computed on the 4 s segments using a 128 ms sample window with a hop-length 42 ms. Overall 120 frames are computed for each sound event. We trained all models for 200 epochs, with a 1×10^{-3} learning rate, 1×10^{-2} weight decay and 0.07 dropout rate in the convolutional blocks.

We used a plain conv-2d model consisting of 4 conv-2d layers with 16, 32, 64, 64 filters each for the KD-based training. The loss was a combination of the cross-entropy computed on the teacher's soft labels and the hard labels given by the ground-truth, with a ratio of 2/3-1/3. The temperature parameter was set to 2. Both hyper-parameters were empirically optimised.

IV. Results

Table I reports the sound event detection accuracy obtained training the models from scratch, as well as using knowledge distillation, considering different configurations. The table also reports the parameter count for each configuration.

TABLE I
ACCURACY ON URBANSOUND8K VARYING THE α AND $t0$ SCALING PARAMETERS, WITH AND WIHTOUT KD. THE TABLE REPORTS ALSO THE MODEL PARAMETER COUNT.

α	$t0$	Acc	Acc-KD	# Parameters
0.20	2	64.87	49.68	4,779
0.35	2	64.64	64.82	12,479
0.50	2	63.90	67.61	24,507
0.20	4	65.85	59.58	8,893
0.35	4	71.80	67.14	23,797
0.50	4	72.25	70.15	47,349
0.20	6	66.05	70.95	13,007
0.35	6	68.02	71.05	35,115
0.50	6	70.39	71.20	70,191

While the performance of the models trained from scratch decays rather linearly with the number of parameters (as shown in Fig. 5), the specific configuration of the hyper-parameters α and t_0 does not seem to have a direct and evident impact on the performance (see Fig 7, 6). Overall, this was expected as the PhiNet architectures are designed to scale efficiently in the MCU range without significantly compromising the network's performance. However, it is worth noting that, in some cases, using larger values of t_0 is preferable with respect to α given a target hyper-parameter count (compare for example the two models (0.5;2) and (0.35;34)). This could be related to the fact that larger convolutional blocks can better represent the information, easing the learning task. Finally, note that the best accuracy (72.25%) is achieved using a medium-size architecture (47K parameters obtained with $\alpha = 0.5$ and $t_0 = 4$). PhiNets are actually designed to be efficient in the MCU range. Therefore, they tend to overfit easily when the model size increases. This issue is further accentuated by the relatively small size of the dataset used in our experiments.

Conversely, models trained via knowledge distillation do not show the same linear performance decay that, instead, drastically decreases for small architectures. Nevertheless, the experimental analysis confirms that KD improves the performance of some configurations. However, this occurs only when the expansion factor is sufficiently high ($t_0 = 6$ in our experiments). In all the other cases, using KD leads to performance deterioration, which in some cases are extremely evident. Our hypothesis is that architectures with high values of t_0 tend to resemble the conv-2d nature of the teacher, thus helping the convergence of the model.

Overall, the scaling principles of PhiNets guarantee a competitive classification accuracy without requiring KD.

To complete our analysis, in Table IV we compare the best performance achieved with PhiNets with the state of the models (AudioCLIP [8]) and with the plain conv-2d model used as teacher. Note that the very high performance of Audio-

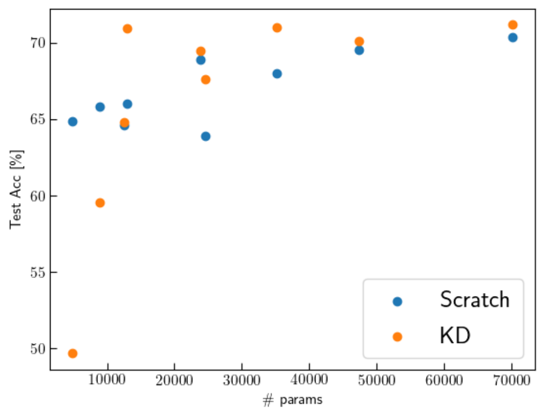


Fig. 5. Event detection accuracy as a function of the overall network parameter count.

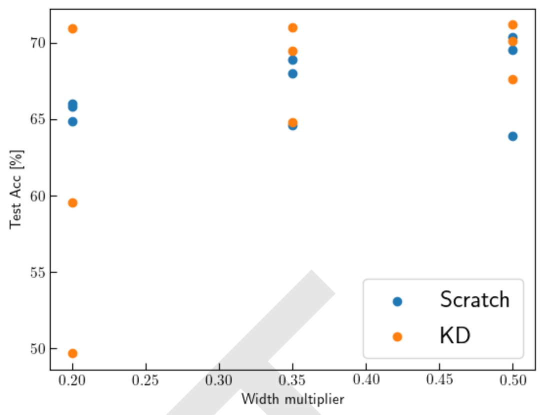


Fig. 7. Testing accuracy as a function of α. Again, we observe that higher values for the width multiplier is preferable when training using knowledge distillation.

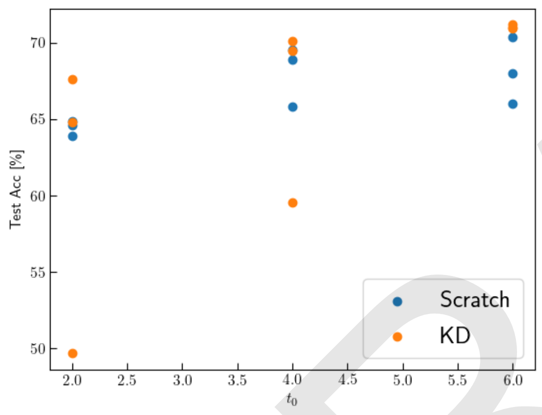


Fig. 6. Classification accuracy as a function of t_0. Note how a higher value for the base expansion factor favors networks trained using knowledge distillation.

CLIP is achieved with 60M parameters, which are definitely not suitable for low-end devices. In addition AudioCLIP has been trained on a much larger amount of data, while our models are trained direcltly on UrbanSound8K. Nevertheless, the proposed PhiNet architecture can reach a 72.2% accuracy with less the 50K parameters. It is interesting to observe that the plain conv-2d teacher considerably outperforms the PhiNet model with a similar parameter count (see the last row of Table I). This is mainly due to the fact that the teacher network composed of 2D convolutions requires largely more operations to run with respect to the largest PhiNet tested (90M vs 10M).

TABLE II
MODEL PERFORMANCE WITH RESPECT TO STATE-OF-THE-ART PLATFORM.

Model name	Test acc [%]	Parameter count
AudioCLIP [8]	90.01	60M
Teacher (Conv-2d)	81.15	66K
PhiNets	72.25	47K

V. CONCLUSIONS

In this paper, we investigated the impact of two width-scaling parameters of PhiNets towards identifying their effects on the performance of urban sound detection to simplify the model design given the available memory and computational resources. Experiments on UrbanSound8K show that while α and t_0 are interchangeable when the model is trained from scratch, and only the number of parameters matters, large values of t_0 are preferable if KD from a pre-trained teacher model can be applied.

In future work, we aim at applying the same KD approach on a video task to validate the results against a different sensing source. Moreover, we plan to compare this approach with adaptive pruning strategies for network compression.

REFERENCES

[1] D. Bajovic *et al.*, "Marvel: Multimodal extreme scale data analytics for smart cities environments," in *2021 International Balkan Conference on Communications and Networking (BalkanCom)*, 2021, pp. 143–147.
[2] F. Paissan, G. Cerutti, M. Gottardi, and E. Farella, "People/car classification using an ultra-low-power smart vision sensor," in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)*, 2019, pp. 91–96.
[3] F. Paissan, M. Gottardi, and E. Farella, "Enabling energy efficient machine learning on a ultra-low-power vision sensor for iot," *arXiv preprint arXiv:2102.01340*, 2021.
[4] 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)*, Orlando, FL, USA, Nov. 2014, pp. 1041–1044.
[5] 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 *ICASSP*, March 2017, pp. 776–780.
[6] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *ACM international conference on Multimedia*, 2015, pp. 1015–1018.
[7] 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 *DCASE*, 2017.
[8] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "Audioclip: Extending clip to image, text and audio," *arXiv preprint arXiv:2106.13043*, 2021.

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

[10] 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.

[11] 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.

[12] Y. Tokozume and T. Harada, “Learning environmental sounds with end-to-end convolutional neural network,” *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2721–2725, 2017.

[13] D. W. Romero, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn, “Wavelet networks: Scale equivariant learning from raw waveforms,” *arXiv preprint arXiv:2006.05259*, 2020.

[14] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.

[15] L. Valerio, F. M. Nardini, A. Passarella, and R. Perego, “Dynamic hard pruning of neural networks at the edge of the internet,” *arXiv preprint arXiv:2011.08545*, 2020.

[16] G. Cerutti, R. Andri, L. Cavigelli, E. Farella, M. Magno, and L. Benini, “Sound event detection with binary neural networks on tightly power-constrained iot devices,” in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, pp. 19–24.

[17] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-s. Hua, “Quantization networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7308–7316.

[18] G. Cerutti, R. Prasad, A. Brutti, and E. Farella, “Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 654–664, 2020.

[19] 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.

[20] F. Paissan and E. F. Alberto Ancilotto, Alessio Brutti, “Scalable neural architectures for end-to-end environmental sound classification,” *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, In press.

[21] F. Paissan, A. Ancilotto, and E. Farella, “PhiNets: a scalable backbone for low-power AI at the edge,” *arXiv preprint arXiv:2110.00337*, 2021.

[22] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” *CoRR*, vol. abs/1801.04381, 2018.

[23] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, “Searching for mobilenetv3,” *CoRR*, vol. abs/1905.02244, 2019.

[24] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” *CoRR*, vol. abs/1709.01507, 2017. [Online]. Available: http://arxiv.org/abs/1709.01507

7.3 Assessment of Self-Attention on Learned Features For Sound Event Localization and Detection

The appended paper follows.

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 probability 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

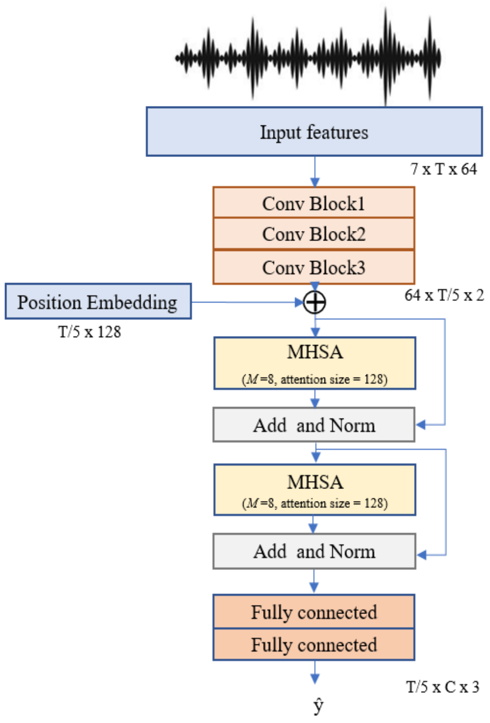


Figure 1: MHSA model configuration for SELD task.

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.

7.4 WaveTransformer: An Architecture for Audio Captioning Based on Learning Temporal and Time-Frequency Information

The appended paper follows.

WaveTransformer: An Architecture for Audio Captioning Based on Learning Temporal and Time-Frequency Information

An Tran
Audio Research Group
Tampere University
Tampere, Finland
an.tran@tuni.fi

Konstantinos Drossos
Audio Research Group
Tampere University
Tampere, Finland
konstantinos.drossos@tuni.fi

Tuomas Virtanen
Audio Research Group
Tampere University
Tampere, Finland
tuomas.virtanen@tuni.fi

***Abstract*—Automated audio captioning (AAC) is a novel task, where a method takes as an input an audio sample and outputs a textual description (i.e. a caption) of its contents. Most AAC methods are adapted from image captioning or machine translation fields. In this work, we present a novel AAC method, explicitly focused on the exploitation of the temporal and time-frequency patterns in audio. We employ three learnable processes for audio encoding, two for extracting the temporal and time-frequency information, and one to merge the output of the previous two processes. To generate the caption, we employ the widely used Transformer decoder. We assess our method utilizing the freely available splits of the Clotho dataset. Our results increase previously reported highest SPIDEr to 17.3, from 16.2 (higher is better).**

***Index Terms*—automated audio captioning, wavetransformer, wavenet, transformer**

I. Introduction

Automated audio captioning (AAC) is an intermodal translation task, where the system receives as an input an audio signal and outputs a textual description of the contents of the audio signal (i.e. outputs a caption) [1]. AAC is not speech-to-text, as the caption does not transcribe speech. In a nutshell, an AAC method learns to identify the high-level, humanly recognized information in the input audio, and expresses this information with text. Such information can include complex spatiotemporal relationships of sources and entities, textures and sizes, and abstract and high-level concepts (e.g. "several barnyard animals mooing in a barn while it rains outside").

There are different published approaches for AAC. Regarding input audio encoding, some approaches use recurrent neural networks (RNNs) [2], [3], [4], others 2D convolutional neural networks (CNNs) [5], [6], and some others the Transformer model [7], [8]. Though, RNNs are known to have difficulties on learning temporal information [9], 2D CNNs model time-frequency but not temporal patterns [10], and the Transformer was not originally designed for sequences of thousands time-steps [7]. For generating the captions, the Transformer decoder [6], [11], [8] or RNNs [1], [3], [5] are mostly employed, and the alignment of input audio and output captions is typically implemented with an attention mechanism [12], [11]. Also, some approaches adopt a multi-task approach, where the AAC method is regularized by the prediction of keywords, based on the input audio [6], [11], [13].

In this paper we present a novel AAC approach, based on a learnable representation of audio that is focused on encoding the information needed for AAC. We adopt existing machine listening approaches where sound sources and actions are well captured by time-frequency information [10], [14], and additionally exploit temporal information in audio using 1D dilated convolutions that operate on the time dimension [15], [16], for learning of high-level information (e.g. background vs foreground, spatiotemporal relationships). Additionally, we claim that these two types of information can be combined, providing a well-performing learned audio representation for AAC. To this end, we present an approach which is explicitly focusing on the above aspects. We employ three different encoding processes for the input audio, one regarding temporal information, a second that considers the time-frequency information, and a third that merges the previous two and its output is given as an input to a decoder which generates the output caption.

The contribution of our work is: i) we present *the first method that explicitly focuses on exploiting temporal and local time-frequency information for AAC*, ii) *we provide highest reported results* using only the *freely available splits of Clotho* dataset and *without any data augmentation and/or multi-task learning*, and iii) we show the impact on the performance of the different components of our method, i.e. the temporal and local time-frequency information, merging the previous two, or all of them. The rest of the paper is as follows. In Section II we present our method. Section III presents the evaluation process of our method, and the obtained results are in Section IV. Section V concludes the paper and proposes future research directions.

II. Proposed method

Our method takes as an input a sequence of T_{a} vectors with F audio features, $\mathbf{X} \in \mathbb{R}^{T_{\text{a}} \times F}$, and outputs a sequence of T_{w}

vectors having W one-hot encoded words, $\mathbf{Y}$. To do so, our method utilizes an encoder-decoder scheme, where the encoder is based on CNNs and the decoder is based on feed-forward neural networks (FFNs) and multi-head attention. Our encoder takes $\mathbf{X}$ as an input, exploits temporal and time-frequency structures in $\mathbf{X}$, and outputs the learned audio representation $\mathbf{Z} \in \mathbb{R}^{T_\text{a} \times F'}$, which is a sequence of T_a vectors of F' learned audio features. The decoder takes as an input $\mathbf{Z}$ and outputs $\mathbf{Y}$. Figure 1 illustrates our proposed method.

A. Encoder

Our encoder, $E(\cdot)$, consists of three learnable processes, $E_\text{temp}(\cdot)$, $E_\text{tf}(\cdot)$, and $E_\text{merge}(\cdot)$. E_temp learns temporal context and frame-level information in $\mathbf{X}$ [16], and is inspired by WaveNet [15] but with non-causal convolutions, since in AAC there is no restriction for causality in the encoding of input audio. E_tf learns time-frequency patterns in $\mathbf{X}$, and is inspired by SOTA methods for sound event detection [10], [14], and E_merge merges the information extracted by E_temp and E_tf.

N_t blocks of CNNs (called wave-blocks henceforth) in E_temp, sequentially process $\mathbf{X}$. Each wave-block consists of seven 1D CNNs, $\text{CNN}^{n_\text{t}}_{\text{t}_1}$ to $\text{CNN}^{n_\text{t}}_{\text{t}_7}$, with n_t to be the index of the wave-block. For example, $\text{CNN}^{2}_{\text{t}_3}$ is the third CNN of the second wave-block. The kernel size, stride, and dilation of $\text{CNN}^{n_\text{t}}_{\{\text{t}_1,\text{t}_4,\text{t}_7\}}$ are one and its padding zero. The kernel size of $\text{CNN}^{n_\text{t}}_{\{\text{t}_2,\text{t}_3\}}$ is three and its padding, dilation, and stride is one. The kernel size of $\text{CNN}^{n_\text{t}}_{\{\text{t}_5,\text{t}_6\}}$ is three, its padding and dilation are two, and stride is one. $\text{CNN}^{n_\text{t}}_{\text{t}_1}$ has $C^{n_\text{t}}_\text{in}$ and $C^{n_\text{t}}_\text{out}$ input and output channels, respectively, and the rest have $C^{n_\text{t}}_\text{out}$ input and output channels.

The above hyper-parameters are based on the WaveNet architecture [15]. The output of the n_t-th wave-block, $\mathbf{H}^{n_\text{t}}_\text{t}$, is obtained by

$$\mathbf{H}''^{n_\text{t}}_{\text{t}_1} = \text{CNN}^{n_\text{t}}_{\text{t}_1}(\mathbf{H}^{n_\text{t}-1}_\text{t}), \tag{1}$$

$$\mathbf{S}''^{n_\text{t}}_\text{t} = \tanh(\text{CNN}^{n_\text{t}}_{\text{t}_2}(\mathbf{H}''^{n_\text{t}}_{\text{t}_1})) \odot \sigma(\text{CNN}^{n_\text{t}}_{\text{t}_3}(\mathbf{H}''^{n_\text{t}}_{\text{t}_1})), \tag{2}$$

$$\mathbf{H}'^{n_\text{t}}_\text{t} = \text{CNN}^{n_\text{t}}_{\text{t}_4}(\mathbf{S}''^{n_\text{t}}_\text{t}) + \mathbf{H}''^{n_\text{t}}_{\text{t}_1}, \tag{3}$$

$$\mathbf{S}'^{n_\text{t}}_\text{t} = \tanh(\text{CNN}^{n_\text{t}}_{\text{t}_5}(\mathbf{H}'^{n_\text{t}}_\text{t})) \odot \sigma(\text{CNN}^{n_\text{t}}_{\text{t}_6}(\mathbf{H}'^{n_\text{t}}_\text{t})), \text{ and} \tag{4}$$

$$\mathbf{H}^{n_\text{t}}_\text{t} = \text{ReLU}(\text{BN}^{n_\text{t}}_\text{t}(\text{CNN}^{n_\text{t}}_{\text{t}_7}(\mathbf{S}'^{n_\text{t}}_\text{t}) + \mathbf{H}'^{n_\text{t}}_{\text{t}_1})), \tag{5}$$

where $\text{BN}^{n_\text{t}}_\text{t}$ is the batch normalization process at the n_t-th wave-block, ReLU is the rectified linear unit, $\sigma(\cdot)$ is the sigmoid non-linearity, $\odot$ is the Hadamard product, $\mathbf{H}^0_\text{t} = \mathbf{X}_\text{t}$, and $\mathbf{H}^{N_\text{t}}_\text{t} \in \mathbb{R}^{C^{n_\text{t}}_\text{out} \times T_\text{a}}_{\geq 0}$. The output of E_temp, $\mathbf{Z}_\text{t} = E_\text{temp}(\mathbf{X}_\text{t})$, is obtained by reshaping $\mathbf{H}^{N_\text{t}}_\text{t}$ to $\{1 \times T_\text{a} \times C^{n_\text{t}}\}$. All CNN^{n_t} operate along the time dimension of $\mathbf{X}_\text{t}$, allowing $\mathbf{H}^{N_\text{t}}_\text{t}$ to learn temporal information from $\mathbf{X}_\text{t}$ [15] and be used effectively in WaveTransformer for learning information that requires temporal context, e.g. spectro-temporal relationships. The time receptive field of each wave-block spans seven time-steps of its corresponding input, leading to a receptive field of $7N_t - 1$ time-steps of $\mathbf{X}$, for the output of the N_t-th wave-block.

E_tf employs N_{tf} blocks of 2D CNNs, called 2DCNN-blocks henceforth. Each 2DCNN-block consists of a 2D CNN

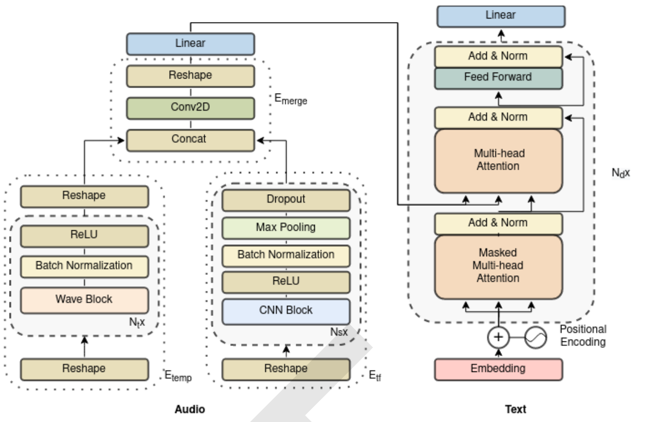


Fig. 1. The WaveTransformer, with the encoder on the left-hand side and the decoder on the right-hand side

($\text{S-CNN}^{n_\text{tf}}$), a leaky ReLU (LU), and a 2D CNN ($\text{P-CNN}^{n_\text{tf}}_\text{tf}$). Each 2DCNN-block is followed by a ReLU, a BN (BN^{n_tf}) process, a max-pooling (MP^{n_tf}) process that operates only on the feature dimension (hyper-parameters according to [10]), and a dropout (DR) with probability of p_{n_tf}. The 2DCNN-blocks are inspired by AAC and sound event detection and classification methods, and the recent, successful adoption of depth-wise separable convolutions [13], [10]. The 2DCNN-blocks learn spatial time-frequency information from their input [10], allowing $\mathbf{H}^{N_\text{d}}_\text{d}$ to be used effectively for the identification of sources and actions [10].

$\text{S-CNN}^{n_\text{tf}}$ consists of $C^{n_\text{tf}}_\text{in}$ different (5, 5) kernels with unit stride, and padding of 2, focusing on learning time-frequency patterns from each channel of its input. Each kernel of $\text{S-CNN}^{n_\text{tf}}$ is applied to only one channel of the input to $\text{S-CNN}^{n_\text{tf}}$, according to the depthwise separable convolution model and to enforce the learning of spatial time-frequency patterns [10]. $\text{P-CNN}^{n_\text{tf}}_\text{tf}$ consists of a square kernel of size $K_\text{P-CNN} > 1$, with unit stride, and padding of 2, focusing on learning cross-channel information from the output of $\text{S-CNN}^{n_\text{tf}}$, since the kernels of $\text{P-CNN}^{n_\text{tf}}_\text{tf}$ operate on all channels of the input to $\text{P-CNN}^{n_\text{tf}}_\text{tf}$.

While hyper-parameters of $\text{S-CNN}^{n_\text{tf}}$ and $\text{S-CNN}^{n_\text{tf}}$ are based on [10], the usage of $K_\text{P-CNN} > 1$ is not according to a typical point-wise convolution (i.e. with a (1, 1) kernel, unit stride, and zero padding), as it was experimentally found that it performs better, using the training and validation data, and the protocol described in Section 3. S-CNN^1 has $C^{n_\text{tf}}_\text{in} = 1$ and $C^{n_\text{tf}}_\text{out} = C^{n_\text{t}}_\text{out}$ input and output channels, respectively. $\text{S-CNN}^{n_\text{tf}>1}$ and $\text{P-CNN}^{n_\text{tf}}$ have input and output channels equal to $C^{n_\text{t}}_\text{out}$. The output of the n_tf-th 2DCNN-block, $\mathbf{H}^{n_\text{tf}}_\text{tf} \in \mathbb{R}^{C^{n_\text{tf}}_\text{out} \times T_\text{a} \times F'_\text{tf}}_{\geq 0}$, is obtained by

$$\mathbf{S}'^{n_\text{tf}}_\text{tf} = \text{P-CNN}^{n_\text{tf}}(\text{BN}^{n_\text{tf}}(\text{LU}(\text{S-CNN}^{n_\text{tf}}(\mathbf{H}^{n_\text{tf}-1}_\text{tf})))) \text{ and} \tag{6}$$

$$\mathbf{H}^{n_\text{tf}}_\text{tf} = \text{DR}(\text{MP}^{n_\text{tf}}(\text{BN}^{n_\text{tf}}(\mathbf{S}'^{n_\text{tf}}_\text{tf}))), \tag{7}$$

where $\mathbf{H}^0_\text{tf} = \mathbf{X}_\text{tf}$ and $\mathbf{H}^{N_\text{tf}}_\text{tf} \in \mathbb{R}^{C^{N_\text{tf}}_\text{out} \times T_\text{a} \times 1}_{\geq 0}$. Then, $\mathbf{Z}_\text{tf} =$

$E_{\text{tf}}(\mathbf{X}_{\text{tf}})$ is obtained by reshaping $\mathbf{H}_{\text{tf}}^{N_{\text{tf}}}$ to $\{1 \times T_{\text{a}} \times C_{\text{out}}^{N_{\text{tf}}}\}$.

E_{merge} consists of a 2D CNN, CNN_{m} and a feed-forward neural network (FNN), FNN_{m}, with shared weights through time. Specifically, CNN_{m} has a (5, 5) kernel with unit stride and dilation, padding of 2, and two input and one output channels. Both $\mathbf{Z}_{\text{t}}$ and $\mathbf{Z}_{\text{tf}}$ have the same dimensionality, are concatenated in their channel dimension, and given as an input to CNN_{m}, as $\mathbf{Z}'' = [\mathbf{Z}_{\text{t}}; \mathbf{Z}_{\text{tf}}]$ and $\mathbf{Z}' = \text{CNN}_{\text{m}}(\mathbf{Z}'')$, where $\mathbf{Z}'' \in \mathbb{R}_{\geq 0}^{2 \times T_{\text{a}} \times C_{\text{out}}^{N_{\text{tf}}}}$, and $\mathbf{Z}' \in \mathbb{R}^{1 \times T_{\text{a}} \times C_{\text{out}}^{N_{\text{tf}}}}$ is the output of CNN_{m}. $\mathbf{Z}'$ is then reshaped to $\{T_{\text{a}} \times C_{\text{out}}^{N_{\text{tf}}}\}$ and given as an input to FNN_{m}, as $\mathbf{Z} = \text{FNN}_{\text{m}}(\mathbf{Z}')$, where $\mathbf{Z} \in \mathbb{R}^{T_{\text{a}} \times F'}$, with $F' = C_{\text{out}}^{N_{\text{tf}}}$.

B. Decoder

We employ the decoder of the Transformer model [7] as our decoder, $D(\cdot)$. During training D takes as an input $\mathbf{Y}$ and $\mathbf{Z}$, and outputs a sequence of T_{w} vectors having a probability distribution over W words, $\hat{\mathbf{Y}} \in [0,1]^{T_{\text{w}} \times W}$. We follow the implementation in [7], employing an FFN as embedding extractor for one-hot encoded words, $\text{FNN}_{\text{emb}}(\cdot)$, a positional encoding process, $P_{\text{enc}}(\cdot)$, N_{dec} decoder blocks, $D^{n_{\text{dec}}}(\cdot)$, and an FFN at the end which acts as a classifier, $\text{FNN}_{\text{cls}}(\cdot)$. FNN_{emb} and FNN_{cls} have their weights shared across the words of a caption. Each $D^{n_{\text{dec}}}$ consists of a masked multi-head self-attention, a layer-normalization (LN) process, another multi-head attention that attends at $\mathbf{Z}$, followed by another LN, an FNN, and another LN.

We model each $D^{n_{\text{dec}}}$ as a function taking two inputs, $\mathbf{U}^{n_{\text{dec}}} \in \mathbb{R}^{T_{\text{w}} \times V_{\text{e}}^{n_{\text{dec}}}}$ and $\mathbf{Z}$, and having as output $\mathbf{H}_{\text{dec}}^{n_{\text{dec}}} \in \mathbb{R}^{T_{\text{w}} \times V_{\text{e}}^{n_{\text{dec}}}}$, with $\mathbf{H}_{\text{dec}}^{0} = \mathbf{H}'_{\text{dec}}$, $\mathbf{U}^{0} = \mathbf{Y}$, and $V_{\text{e}}^{0} = W$. All FNNs of each $D^{n_{\text{dec}}}$ have input-output dimensionality of $V_{\text{e}}^{n_{\text{dec}}}$. We use N_{att} attention heads and for the multi-head attention layers and p_{d} dropout probability. For the implementation details, we refer the reader to the paper of Transformer model [7]. FNN_{emb} takes as an input $\mathbf{Y}$ and its output is processed by the positional encoding process, as

$$\mathbf{H}'_{\text{dec}} = P_{\text{enc}}(\text{FNN}_{\text{emb}}(\mathbf{Y}))), \tag{8}$$

where P_{enc} is according to the original paper [7]. $\mathbf{H}'_{\text{dec}}$ is processed serially by the N_{dec} decoder blocks, as $\mathbf{H}_{\text{dec}}^{n_{\text{dec}}} = D^{n_{\text{dec}}}(\mathbf{H}_{\text{dec}}^{n_{\text{dec}}-1}, \mathbf{Z})$, and then we obtain $\hat{\mathbf{Y}}$ as

$$\hat{\mathbf{Y}} = \text{FNN}_{\text{cls}}(\mathbf{H}_{\text{dec}}^{N_{\text{dec}}}). \tag{9}$$

We optimize jointly the parameters of the encoder and decoder, by minimizing the cross-entropy loss between $\mathbf{Y}$ and $\hat{\mathbf{Y}}$.

III. Evaluation

To evaluate our method, we employ the dataset and protocol defined at the AAC task at the DCASE2020 challenge. The code and the pre-trained weights of our method are freely available online[1]. We also provide an online demo of our method, with 10 audio files, the corresponding predicted captions, and the corresponding ground truth captions[2].

[1]https://github.com/haantran96/wavetransformer

[2]https://haantran96.github.io/wavetransformer-web-demo/

A. Dataset and pre- and post-processing

We employ the freely available and well curated AAC dataset, Clotho, consisting of around 5000 audio samples of CD quality, 15 to 30 seconds long, and each sample is annotated by human annotators with five captions of eight to 20 words, amounting to around 25 000 captions [4], [17]. Clotho is divided in three splits: i) development, with 14465 captions, ii) evaluation, with 5225, and iii) testing with 5215 captions. We employ development and evaluation splits which are publicly and freely available. We extract $F = 64$ log mel-band energies using Hamming window of 46ms with 50% overlap from the audio files, resulting to $1292 \leq T_{\text{a}} \leq 2584$, for audio samples whose length is between 15 and 30 seconds. During training, to mitigate the length difference of the audio samples in Clotho, in each mini-batch we make all input audio samples to have same length by pre-pending zeros to the shorter ones.

We process each caption and we prepend and append the <sos> (start-of-sentence) and <eos> (end-of-sentence) tokens, respectively. Additionally, we process the development split and we randomly select and reserve 100 audio samples and their captions in order to be used as a validation split during training. These 100 samples are selected according to the criterion that their captions do not contain a word that appears in the captions of less than 10 audio samples. We term the resulting training (i.e. development minus the 100 audio samples) and validation splits as Dev_{tra} and Dev_{val}, respectively. We also provide the file names from Clotho development split used in Dev_{val}, at the online repository of WaveTransformer[2]. We post-process the output of WaveTransformer during inference, employing both greedy and beam search decoding. Greedy decoding stops when <eos> token or when 22 words are generated. During training, to mitigate the length difference of the captions in Clotho, in each mini-batch we make all captions to have same length by appending <eos> tokens to the shorter ones.

B. Hyper-parameters, training, and evaluation

We employ the Dev_{tra} (as training split) and Dev_{val} (as validation split) to optimize the hyper-parameters of our method, using an early stopping policy with a patience of 10 epochs. We employ Adam optimizer [18], a batch size of 12, and clipping of the 2-norm of the gradients to the value of 1. The employed hyper-parameters of our method are $N_{\text{t}} = 4$, $N_{\text{tf}} = 3$, $C_{out}^{n_{\text{t}}} = V_{\text{e}} = 128$, $F'_{tf} = 1$, $N_{\text{dec}} = 3$, $N_{\text{att}} = 4$, $p_{n_{\text{tf}}} = p_{\text{d}} = 0.25$, and beam size of 2. This leads to the modelling of $7N_{\text{t}} - 1 = 27$ frames, equivalent to 0.7 seconds for current $\mathbf{X}$, for E_{temp}.

To assess the performance of WaveTransformer (WT) and the impact of E_{temp}, E_{tf}, E_{merge}, and beam search, we employ the WT, WT without E_{tf} and E_{merge} (WT_{temp}), without E_{temp} and E_{merge} (WT_{tf}), and without E_{merge} (WT_{avg}), where we replace E_{merge} with an average between E_{temp} and E_{tf}. We evaluate the performance of WT with greedy decoding and with beam searching (indicated as WT-B) on Clotho evaluation split and using the machine translation metrics BLEU_1 to

TABLE I

RESULTS ON CLOTHO EVALUATION DATASET. B_n STANDS FOR BLEU_n. BOLDFACE FONTS INDICATE THE BEST VALUES FOR EACH METRIC

Model	$\mathbf{B_1}$	$\mathbf{B_2}$	$\mathbf{B_3}$	$\mathbf{B_4}$	**METEOR**	$\mathbf{ROUGE_L}$	**CIDEr**	**SPICE**	**SPIDEr**
TRACKE (w/o MT) [6]	50.2	29.9	18.3	10.2	14.1	**33.7**	23.3	09.1	16.2
NTT (w/o MT, DA, and PP) [11]	**52.1**	29.4	17.4	10.3	13.8	33.5	23.2	08.5	15.8
NTT (MT+PP, w/o DA) [11]	52.0	**31.2**	**20.0**	**12.7**	14.0	33.7	26.1	08.2	17.2
$\mathrm{WT}_{\mathrm{temp}}$	45.8	25.9	15.4	08.8	13.9	32.0	19.8	08.7	14.2
$\mathrm{WT}_{\mathrm{tf}}$	47.9	28.0	17.1	10.2	14.7	33.1	24.7	09.3	17.0
$\mathrm{WT}_{\mathrm{avg}}$	47.9	28.1	17.1	10.3	14.8	33.0	24.7	09.4	17.0
WT	48.4	28.2	17.4	10.2	**14.8**	33.2	24.7	**09.9**	17.3
WT-B	49.8	30.3	19.7	12.0	14.3	33.2	**26.8**	09.5	**18.2**

BLEU_4 scores, METEOR, and $\mathrm{ROUGE_L}$ [19], [20], [21], and the captioning metrics CIDEr, SPICE, and SPIDEr [22], [23], [24]. In a nutshell, BLEU_n measures a weighted geometric mean of modified precision of n-grams, METEOR measures a harmonic mean of recall and precision for segments between the two captions, and $\mathrm{ROUGE_L}$ calculates an F-measure using the longest common sub-sequence. On the other hand, CIDEr calculates a weighted cosine similarity of n-grams, using term-frequency inverse-document-frequency weighting, SPICE measures how well the predicted caption recovers objects, attributes, and their relationships, and SPIDEr is the average of CIDEr and SPICE, exploiting the advantages of CIDEr and SPICE.

Additionally, we compare our method with the two highest-performing AAC methods, NTT [11] and TRACKE [6], developed and evaluated using only Clotho development and evaluation splits. NTT uses different components, like multi-task learning (MT), data augmentation (DA), and post-processing (PP), but authors provide results without these components. TRACKE is the current SOTA, it also uses MT but the authors provide results without MT. We compare our WT against TRACKE without MT and NTT without (w/o) DA.

IV. RESULTS

Table I presents the results of WT, NTT, and TRACKE, where our comparison is limited to methods that are using only the publicly available splits of Clotho. It must be noted that both the systems presented at the papers of the NTT and TRACKE methods, employ data augmentation (DA) and/or multi-task learning (MT) schemes, achieving higher SPIDEr. Since WT is not employing MT and DA, in Table I we compare to the version of NTT and TRACKE methods that have similar set-ups as the WT. As can be seen, the learning of time-frequency information ($\mathrm{WT}_{\mathrm{tf}}$) can lead to better results than learning temporal information ($\mathrm{WT}_{\mathrm{temp}}$) instead. We hypothesize that this is because the decoder can learn an efficient language model, filling the connecting gaps (e.g. interactions of objects) between sound events learned from E_{tf}. However, from the results it can be seen that employing both E_{temp} and E_{tf} increases more the performance of the WaveTransformer (WT).

Comparing the different scores for the employed metrics and for the $\mathrm{WT}_{\mathrm{tf}}$ and WT cases, shows that the utilization of E_{temp} is not contributing much in the ordering of words, as indicated by the difference of BLEU metrics between $\mathrm{WT}_{\mathrm{tf}}$ and WT. We can see that with the E_{temp}, our method learns better attributes of objects and their relationships, as indicated by CIDEr and SPICE scores. Thus, we argue that E_{temp} contributes in learning attributes and interactions of objects, while E_{tf} contributes information about objects and actions (e.g. sound events). Also, by observing the results for $\mathrm{WT}_{\mathrm{avg}}$, we can see that a simple averaging of the learned information by E_{temp} and E_{tf} leads to a better description of objects, attributes, and their relationships (indicated by SPICE). Though, as can be seen by comparing $\mathrm{WT}_{\mathrm{avg}}$ and WT, the E_{merge} manages to successfully merge the information by E_{temp} and E_{tf}. The utilization of beam search (WT-B) gives a significant boost to the performance, reaching up to 18.2 SPIDEr. Compared to TRACKE and NTT methods, we can see that when excluding DA, MT, and PP, our method (WT) performs better. Additionally, WT-B performs better than NTT with MT and PP. Our post-processing consists only on using beam search, where the NTT method involves a second post-processing technique by augmenting the input data and averaging the predictions. Thus, WT surpasses the other methods that is compared against.

Finally, two, high SPIDEr-scoring, captions are for the files *Flipping pages.wav*, and *110422_village_dusk.wav* of the evaluation split of Clotho. Our predicted captions for each of these files, using WT-B, are: “a person is flipping through the pages of a book” and “a dog is barking while birds are chirping in the background”, respectively, and the best matching ground truth captions are “a person is flipping through pages in a notebook” and “a dog is barking in the background while some children are talking and birds are chirping”, respectively.

V. CONCLUSION

In this paper we presented a novel architecture for AAC, based on convolutional and feed-forward neural networks, called WaveTransformer (WT). WT focuses on learning long temporal and time-frequency information from audio, and expressing it with text using the decoder of the Transformer model. We evaluated WT using the dataset and the metrics adopted in the AAC DCASE Challenge, and we compared our method against previous SOTA methods and the DCASE AAC baseline. The obtained results show that learning time-frequency information, combined with a good language model, can lead to good AAC performance, but incorporating long temporal information can boost the obtained scores.

ACKNOWLEDGEMENT

The authors wish to thank D. Takeuchi and Y. Koizumi for their input on previously reported results, and to acknowledge CSC-IT Center for Science, Finland, for computational resources. Part of the needed computations was implemented on a GPU donated from NVIDIA to K. Drossos. 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.

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.

[2] M. Wu, H. Dinkel, and K. Yu, "Audio caption: Listen and tell," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019.

[3] K. Nguyen, K. Drossos, and T. Virtanen, "Temporal sub-sampling of audio feature sequences for automated audio captioning," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, 2020.

[4] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: an audio captioning dataset," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[5] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, and M. Cobos, "Task 6 dcase 2020: Listen carefully and tell: An audio captioning system based on residual learning and gammatone audio representation," Tech. Rep., DCASE2020 Challenge, Jun. 2020.

[6] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A transformer-based audio captioning model with keyword estimation," in *INTERSPEECH 2020*, 2020.

[7] A. Vaswani et al., "Attention is all you need," in *31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017.

[8] A. Shi, "Audio captioning with the transformer automated audio captioning," Tech. Rep., DCASE2020 Challenge, 2020.

[9] D. Serdyuk, N.-R. Ke, A. Sordoni, A. Trischler, C. Pal, and Y. Bengio, "Twin Networks: Matching the future for sequence generation," *CoRR*, vol. abs/1708.06742, 2017.

[10] 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)*, 2020.

[11] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre- and post- processings for audio captioning," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, 2020.

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

[13] E. Çakır, K. Drossos, and T. Virtanen, "Multi-task regularization based on infrequent classes for audio captioning," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, 2020.

[14] E. Çakır, 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.

[15] A. den Oord et al., "Wavenet: A generative model for raw audio," in *9th International Speech Communication Association (ISCA) Speech Synthesis Workshop*, 2016.

[16] Hyungui Lim, J. Park, K. Lee, and Yoonchang Han, "Rare sound event detection using 1d convolutional recurrent neural networks," Tech. Rep., DCASE2020 Challenge, 2017.

[17] 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.

[18] D. P Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representation (ICLR)*, 2014.

[19] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002.

[20] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *second workshop on statistical machine translation*, 2007.

[21] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004.

[22] 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.

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

[24] 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)*, 2017.

7.5 Continual Learning for Automated Audio Captioning Using The Learning Without Forgetting Approach

The appended paper follows.

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 $\text{M}_{\text{base}}(\cdot;\theta_{\text{base}})$, 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

$$\hat{\mathbf{Y}}'_k = \text{M}_{\text{base}}(\mathbf{X}'_k;\theta_{\text{base}}), \tag{1}$$

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

$$\mathcal{L}(\theta_{\text{base}}, \mathbb{D}_{\text{ori}}) = \sum_{k=1}^{K} \text{CE}(\mathbf{Y}'_k, \hat{\mathbf{Y}}'_k), \tag{2}$$

where CE is the cross-entropy loss between $\mathbf{Y}'_k$ and $\hat{\mathbf{Y}}'_k$.

Then, we create a copy of M_{base}, $\text{M}_{\text{new}}(\cdot;\theta_{\text{new}})$, 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

$$\mathbb{D}_{\text{new}} = \{(\mathbf{X}, \mathbf{Y})_b : (\mathbf{X}, \mathbf{Y}) \sim \mathcal{S} \wedge b = 1, \dots, B\}, \tag{3}$$

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

$$\hat{\mathbf{Y}}_b^{\text{base}} = \text{M}_{\text{base}}(\mathbf{X}_b;\theta_{\text{base}}), \text{ and} \tag{4}$$

$$\hat{\mathbf{Y}}_b^{\text{new}} = \text{M}_{\text{new}}(\mathbf{X}_b;\theta_{\text{new}}), \tag{5}$$

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

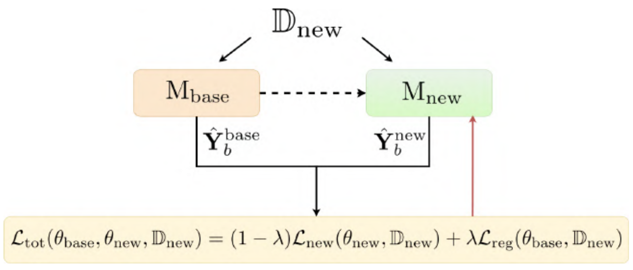

$\mathbb{D}_{\text{new}}$
M_{base}
M_{new}
$\hat{\mathbf{Y}}_b^{\text{base}}$
$\hat{\mathbf{Y}}_b^{\text{new}}$
$\mathcal{L}_{\text{tot}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}) = (1-\lambda)\mathcal{L}_{\text{new}}(\theta_{\text{new}}, \mathbb{D}_{\text{new}}) + \lambda\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \mathbb{D}_{\text{new}})$

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}.

$$\mathcal{L}_{\text{tot}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}) = (1-\lambda)\mathcal{L}_{\text{new}}(\theta_{\text{new}}, \mathbb{D}_{\text{new}}) + \lambda\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \mathbb{D}_{\text{new}}), \text{ where} \tag{6}$$

$$\mathcal{L}_{\text{new}}(\theta_{\text{new}}, \mathbb{D}_{\text{new}}) = \sum_{b=1}^{B} \text{CE}(\mathbf{Y}_b, \hat{\mathbf{Y}}_b^{\text{new}}), \tag{7}$$

$$\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}) = \sum_{b=1}^{B} \text{KL}(\hat{\mathbf{Y}}_b^{\text{base}}, \hat{\mathbf{Y}}_b^{\text{new}}), \text{ and} \tag{8}$$

λ is a factor that weights the contribution of $\mathcal{L}_{\text{new}}$ and $\mathcal{L}_{\text{reg}}$ to $\mathcal{L}_{\text{tot}}$, and $\text{KL}(a, b)$ 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

$$\theta_{\text{new}}^{\star} = \underset{\theta_{\text{new}}}{\text{argmin}}\ \mathcal{L}_{\text{tot}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}), \tag{9}$$

where $\theta_{\text{new}}^{\star}$ are the new, optimized parameters. After obtaining $\theta_{\text{new}}^{\star}$, we update θ_{new} as

$$\theta_{\text{new}} = \theta_{\text{new}}^{\star}. \tag{10}$$

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].

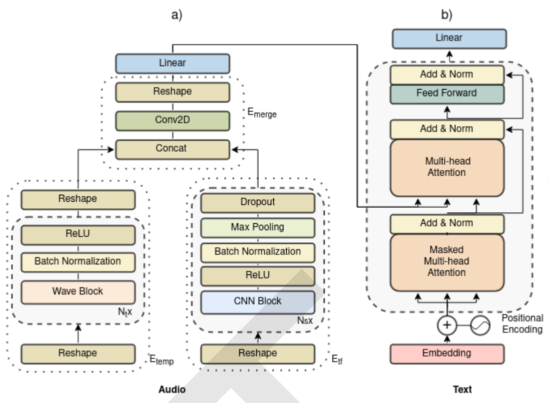


Figure 2: WT architecture, where a) is the encoder and b) the decoder, after [22].

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 sub-sampling 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.

7.6 Automatic Analysis of the Emotional Content of Speech in Day-long Child-Centered Recordings from a Neonatal Intensive Care Unit

The appended paper follows.

Automatic Analysis of the Emotional Content of Speech in Daylong Child-Centered Recordings from a Neonatal Intensive Care Unit

Einari Vaaras[1], *Sari Ahlqvist-Björkroth*[2], *Konstantinos Drossos*[1], *Okko Räsänen*[1,3]

[1]Unit of Computing Sciences, Tampere University, Finland
[2]Department of Clinical Medicine, University of Turku, Finland
[3]Department of Signal Processing and Acoustics, Aalto University, Finland

einari.vaaras@tuni.fi, sarahl@utu.fi, konstantinos.drossos@tuni.fi, okko.rasanen@tuni.fi

Abstract

Researchers have recently started to study how the emotional speech heard by young infants can affect their developmental outcomes. As a part of this research, hundreds of hours of daylong recordings from preterm infants' audio environments were collected from two hospitals in Finland and Estonia in the context of so-called APPLE study. In order to analyze the emotional content of speech in such a massive dataset, an automatic speech emotion recognition (SER) system is required. However, there are no emotion labels or existing in-domain SER systems to be used for this purpose. In this paper, we introduce this initially unannotated large-scale real-world audio dataset and describe the development of a functional SER system for the Finnish subset of the data. We explore the effectiveness of alternative state-of-the-art techniques to deploy a SER system to a new domain, comparing cross-corpus generalization, WGAN-based domain adaptation, and active learning in the task. As a result, we show that the best-performing models are able to achieve a classification performance of 73.4% unweighted average recall (UAR) and 73.2% UAR for a binary classification for valence and arousal, respectively. The results also show that active learning achieves the most consistent performance compared to the two alternatives.

Index Terms: speech emotion recognition, speech analysis, real-world audio, daylong audio, LENA recorder

1. Introduction

In speech emotion recognition (SER), the task is to recognize emotional states of speakers from speech signals [1, 2]. One potential application of SER is the study of babies' auditory environments, where the early emotional experiences of babies, including affective speech, can impact their later cognitive development. In order to study this relationship, *Auditory environment by Parents of Preterm infant; Language development and Eye-movements* (APPLE) study has collected a large audio corpus of child-centered daylong audio recordings from neonatal intensive care units (NICUs), recorded in Turku University Hospital, Finland, and Tallinn Children's Hospital, Estonia [3]. In order to analyze the emotional contents of speech in the recordings, a functional SER system for this new domain is required.

The purpose of the present study is to develop such a system to analyze these (initially unannotated) hospital-environment audio recordings for their emotional speech content. The absence of in-domain annotations and massive scale of the data raises the question of how to most effectively deploy a SER system for this real-world large-scale dataset.

In principle, *cross-corpus generalization* (CCG) is the most straightforward strategy to deploy SER for an unlabeled dataset, but can suffer from domain mismatch. In fact, [4] have shown through extensive multi-corpus and multilingual experiments that reliable CCG-based SER was only feasible with certain corpora and emotional classes, highlighting many issues with cross-domain SER model generalization to out-of-domain data (but see also, e.g., [5] for a potential remedy). In order to tackle the issue of domain mismatch, different *domain adaptation* (DA) methods have been utilized in SER. For instance, Deng et al. [6] extended an unsupervised deep denoising autoencoder (AE) by combining it with a supervised learning objective to create a semi-supervised DA method for SER. Another approach in [7] used an unsupervised deep neural network (DNN)-based adversarial DA approach for SER. The method learns a domain-invariant feature representation between labeled source data and unlabeled target-domain data while maintaining a good performance on the primary SER task. A number of other DA methods for SER have been proposed as well (e.g., [8–10]).

Active learning (AL) is another strategy and has been successfully applied to SER as well. Zhao and Ma [11] presented an iterative AL algorithm, which utilizes conditional random fields, to determine the level of uncertainty for each unlabeled sample. The most uncertain samples were then selected for human annotation. Another study [12] examined different AL methods based on uncertainty and diversity maximization in a simulation setup with DNN classifiers. The work showed that the tested AL methods outperformed random sampling-based methods with a constrained labeling budget.

Only a few SER studies have been conducted on large-scale datasets. Jia et al. [13] studied DNN-based SER with a massive 7-million-utterance internet voice corpus. They pretrained their novel DNN-based models with 90,000 unlabeled utterances, and fine-tuned and evaluated them on 3,000 randomly selected manually annotated utterances from the same dataset. Fan et al. [14] presented a SER dataset with a total duration of over 200 hours. They proposed a novel SER model containing pyramid convolutions which outperformed other models that were tested on the dataset. Additionally, they showed that existing models are prone to overfit to small-scale datasets, which limits the ability of these models to generalize for real-life data.

However, CCG, AL, and DA have rarely been compared to each other directly. Moreover, most of the existing work has been conducted using studio, telephone, or internet speech data. Therefore, our present daylong audio dataset from a hospital context, together with its practical significance, provides an excellent test bench to compare strategies for SER system development in a novel domain with challenging real-world speech data. More specifically, by using the Finnish subset of the data, we compare CCG and state-of-the-art DA and AL in the task to study their feasibility and SER performance in practice.

2. Methods

2.1. Medoid-based active learning

Zhao et al. [15] presented an AL method called medoid-based active learning (MAL) to effectively utilize a small number of annotations, which serves as the foundation of the AL method used in our experiments. The algorithm can be divided into three subsequent parts: 1) obtaining a distance matrix that contains the pairwise distances between all samples in the dataset, 2) performing k-medoids clustering using the distance matrix, and 3) starting from the largest cluster, querying human annotations for the medoids in a descending cluster size order.

The distance metric used in the present experiments was selected based on pilot experiments with MAL using existing SER datasets. A 600-dimensional utterance-level log-mel feature representation (see Section 4.1) was first used as the initial feature representation of each sample in a dataset. These features were then compressed into a 32-dimensional latent representation using a DNN-based AE with six layers. Pearson distances d_P [16] between the bottleneck features were then used to define the affinity matrix A across all the samples. Next, k-medoids clustering was applied to the data. First, one sample was randomly selected as the member of a set S, followed by an addition of $k-1$ more samples as centroids using the *farthest-first traversal* algorithm. Here, the distance from a sample, $\boldsymbol{a}$, to the set S was defined as

$$d_P(\boldsymbol{a}, S) = \min_{\boldsymbol{b} \in S} d_P(\boldsymbol{a}, \boldsymbol{b}) \ . \tag{1}$$

The samples in S were then used as the initial medoids for a k-medoids clustering algorithm (see e.g. [17] for an overview) to assign each sample in the dataset into one of the clusters.

In the final stage, the clusters were sorted in a descending order based on the number of samples in each cluster, and their medoids were presented to human annotators for labeling. In the experiments, we studied the use of these labels in two different ways: i) assigning each sample in a cluster with the annotated medoid label (as in [15]; here referred to as *"cluster labels"*), or ii) only using the medoid samples as labeled data for classifier training, which was not studied in the original MAL paper [15]. Based on pilot experiments on other datasets, k was set to $\frac{N}{3}$, where N is the number of samples in a corpus.

2.2. Wasserstein distance-based domain adaptation

The present DA approach was based on the Wasserstein distance-based domain adaptation (WDA) method proposed in [18]. In WDA, a neural network (NN) classifier, aka the *source model* M, is adapted to a target corpus, D_T, by using labeled data from source domain corpus/corpora, D_S. The source model M consists of two parts, a feature extractor, F_S, and a label classifier, C_L. The adaptation process of WDA involves two stages, which are demonstrated in Fig. 1.

The first stage (Fig. 1, top) consists of training M using samples X_S and their labels Y_S from D_S to obtain a trained F_S. This is done using binary cross-entropy [18] as the loss:

$$L_M(\boldsymbol{x}, \boldsymbol{y}) = - \sum_{(\boldsymbol{x},\boldsymbol{y}) \in (X_S, Y_S)} \boldsymbol{y}^T log_{10}(C_L(F(\boldsymbol{x}))). \tag{2}$$

In the second stage (Fig. 1, bottom), F_S is adapted to D_T to obtain an adapted feature extractor, F_T, by minimizing the Wasserstein-1 distance W_d between the distributions of D_S and D_T using an adversarial training process. Following a WGAN framework [19], F_S is adapted into F_T by finding a common feature representation for D_S and D_T by iteratively minimizing the two losses:

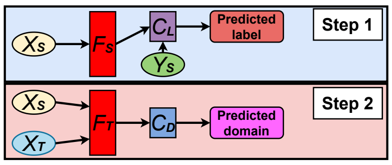


Figure 1: *The two-step the adaptation process of WDA. First, F_S and C_L are trained to classify source corpus samples into emotion categories. In the second step, F_S is adapted into F_T using a domain discriminator C_D with an adversarial loss.*

$$L_{C_D}(\boldsymbol{x}, \boldsymbol{z}) = \sum_{\boldsymbol{x} \in X_S} C_D(F_S(\boldsymbol{x})) - \sum_{\boldsymbol{z} \in X_T} C_D(F_T(\boldsymbol{z})) \tag{3}$$

$$L_{F_T}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in X_T} C_D(F_T(\boldsymbol{z})) + L_M(\boldsymbol{x}, \boldsymbol{y}) \ , \tag{4}$$

where C_D is the domain discriminator and X_T are the target corpus samples. The parameters for C_D and F_T are updated in turns, where Eqs. 3 and 4 are the loss functions for updating the parameters of C_D and F_T, respectively. The output features of F_T are the input features for C_D. Additionally, the parameters of F_S serve as the initial parameters of F_T. As pointed out in [18], the minimization of Eqs. 3 and 4 is shown to minimize W_d between the distributions of D_S and D_T. For a detailed formulation of the WDA algorithm, see Algorithm 1 in [18].

2.3. Cross-corpus generalization

As our baseline approach, we use CCG with different source corpora and their combinations. Labels of each corpus are first mapped to a common emotion category space, followed by a standard supervised classifier training (see Section 4.2).

3. Data

3.1. NICU-A

The FinEst NICU Audioset (NICU-A) was collected in the APPLE study, and is the primary audio material for which our SER system was aimed to be deployed on. We use the Finnish subset of the dataset, which was recorded at the NICU of Turku University Hospital using LENA-recorders (https://www.lena.org/) placed at the bedside of preterm babies (average age approx. 33 gestational weeks) in intensive care. The data consists of 43 x 16-hour recordings from different participating families (a total of 688 h of audio). The recordings were carried out in relatively calm single family rooms of the NICU, where only the baby, visiting parents (primary talkers), and, occasionally, nurses and doctors carrying out healthcare routines were present.

Broad-class diarization of LENA software [20] was used to split each 16-h recording into utterance-sized segments, and to assign a speaker tag (male, female, key child, other child) to the utterances. Based on the validity study reported for the same data in [21], adult speech from "male/female adult" "near" and "far" -categories were included in the analyses to capture caregiver speech (but see [22] for general guidelines with LENA "far" data). Utterances shorter than 600 ms were discarded from further analysis. This resulted in a total of 129,007 utterances with an average length of 1.57 s (approx. 56 h of speech).

Eight families were carefully selected as the test data and 35 as the training data based on the representativeness of both data sets in terms of covariates such as child health, parental presence etc. After pre-processing the data of NICU-A, both the training and test sets were partially annotated.

For the training data, samples were selected for annotation using MAL, as described in Section 4.2.1. Two annotators performed labeling for distinct subsets of the data, except for the first 200 samples that were annotated by both to measure inter-rater agreement rates. Each sample was annotated in two dimensions: in terms of binary arousal (high/low) and in terms of ternary valence (negative, neutral, positive). The two dimensions were annotated in a random order for each sample. A sample could also be labeled as erroneous, if the samples were corrupted by noise, had overlapping speakers, had very short speech fragments, or did not contain speech at all.

For the test data, gold standard (GS) annotations were obtained from three speech/clinical experts for a randomly selected subset of samples from the test set. All GS samples were independently annotated for their arousal and valence by all three annotators, followed by majority voting of labels. Samples without majority labels were removed from the test set. GS annotators had access to 10 s of the preceding audio context of each sample to better understand the communicative context.

After removing the erroneous files, the training and test sets had 5198 and 345 labeled samples, respectively. Training data inter-annotator agreement rates in terms of kappa scores were 0.78 for valence and 0.64 for arousal. For the GS data, the kappa scores were 0.48 for valence and 0.28 for arousal. The difference between the training and testing agreement rates is explained due to the use of MAL in the selection of the training samples, where the first 200 samples annotated by both annotators were also the most acoustically distinct samples in the training data. The finding also demonstrates the inherent difficulty in annotating a random sample of real-world speech for emotional content.

The 'neutral' and 'negative' classes for valence were merged for NICU-A, bearing in mind that the APPLE study was primarily interested in the proportion of positive valence over other speech. As a result, training sample counts were 1509 for positive and 3689 for neutral valence, and 3165 and 2033 for high and low arousal, respectively. The corresponding test set counts were 120 (positive) and 225 (neutral) for valence, and 89 (high) and 256 (low) for arousal.

3.2. Other corpora for CCG and DA experiments

In addition to NICU-A, four existing SER corpora (referred to as *source corpora*) were used in the CCG and DA experiments:

The Berlin Emotional Speech Database (EMO-DB) [23] is a widely used corpus and consists of 535 spoken utterances in German from 10 professional actors with seven emotional labels: anger, boredom, disgust, fear, joy, neutral, and sadness.

eNTERFACE [24] is an audiovisual database consisting of 1287 video samples in English from 42 test subjects from 14 nationalities in six categories: anger, disgust, fear, joy, sadness, and surprise. Only the audio tracks were used in this study.

The Finnish Emotional Speech Corpus (FESC) [25] consists of nine professional actors portraying emotions of five different categories: neutral, sadness, joy, anger, and tenderness. These portrayals were split into 4254 utterances based on long silences as defined by an energy threshold [26].

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [27] is a multimodal database including a total of 7356 recordings from 24 professional actors, out of which 1440 speech-only recordings were used in the present study. Eight different emotional labels were included: neutral, calm, happy, sad, angry, fearful, surprise, and disgust.

4. Experimental setup

4.1. Features

Log-mel, GeMAPS, and eGeMAPS [28] features were used in the CCG and AL experiments. For the DA experiments, only log-mel features were used due to their superior performance in pilot experiments. For the log-mel features, 40 mel filters were used with a Hann window using a 30-ms window size and 10-ms shifts. To get constant-dimensional utterance feature representations, seven functionals (the first four moments, min, max, and range) were taken from the time series of the log-mel features. In addition, four functionals (the first four moments) were applied to first and second order delta features. This resulted in a 600-dimensional feature vector for the log-mel features. The 62- and 88-dimensional GeMAPS and eGeMAPS features were extracted using the openSMILE toolkit [29]. The features for each corpus were z-score normalized at the corpus level.

4.2. Conducted experiments

For the source corpora, the emotional labels were mapped into the quarters of the valence-arousal plane following [4], with the exception of merging 'neutral' and 'negative' valence to 'neutral' in order to better correspond to the labels of NICU-A. The emotional mapping of [4] has been used in multiple SER studies (e.g. [8, 10, 30, 31]). All classification tests were conducted on the NICU-A GS data. We use the unweighted average recall (UAR %) as the primary evaluation measure.

4.2.1. Active Learning Experiments

In the AL experiments, MAL was performed for the full unlabeled training set of NICU-A (101,813 samples). To compress the log-mel features of the training set into a latent representation, an AE network was used. The training and validation data for the AE were based on a random split of the training set using a ratio of 80:20 utterances. The encoder of the AE consisted of three fully-connected (FC) ELU [32] layers of 512, 512, and 32 units, and the decoder of two 512-unit ELU layers and a linear reconstruction layer. The first two AE layers had a dropout of 0.1. The model was trained using MSE loss, Adam [33] optimizer ($lr = 10^{-4}$), batch size of 1024, and early stopping with a patience of 300. The best model according to the validation loss was then used to compress the data to 32 dimensions. Then, MAL was performed for each of the 35 training set families separately and the data were sent for annotation (Section 3.1).

The annotated samples were then used for training a support vector machine (SVM) with an RBF kernel. Each sample was weighted inversely proportional to its class frequency to counter class distribution imbalances. Optimal SVM hyperparameters were selected for each feature type and both classification tasks individually based on a grid search using 5-fold cross-validation over the training data. Then, the SVM was trained on the full training data using these hyperparameters and tested on the GS data. The process was performed separately for the labeled training set of 5,198 samples and for the extended training set of 33,979 samples using the cluster labels from MAL.

4.2.2. Cross-corpus Generalization Experiments

For the CCG experiments, two settings were explored: 1-to-1 and 4-to-1 CCG. In the 1-to-1 setting, each of the source corpora was used individually as the training set. In the 4-to-1 setting, all four source corpora were used for SVM training with similar specifications as with the AL experiments.

Table 1: *UAR (%) performance scores for alternative approaches on the target data. For AL and CCG, log-mel (log-m), GeMAPS (Ge), and eGeMAPS (eGe) features are compared. For DA, the unsupervised (US) and semi-supervised (S-S) variant of WDA is compared. The highest accuracies are* ***bolded***.

Experiment	UAR (%)						
		Valence			Arousal		
AL	Cluster labels	*log-m*	*Ge*	*eGe*	*log-m*	*Ge*	*eGe*
	No	70.9	71.0	71.9	68.5	**69.3**	65.8
	Yes	68.2	**73.4**	72.9	67.0	68.9	67.6
CCG	Training corpus	*log-m*	*Ge*	*eGe*	*log-m*	*Ge*	*eGe*
	EMO-DB	48.5	53.8	53.4	64.1	63.7	62.7
	eNTERFACE	56.8	52.7	50.2	63.1	64.3	64.1
	FESC	45.3	**57.3**	54.9	56.3	68.3	**70.8**
	RAVDESS	50.4	53.8	53.3	64.3	62.0	58.7
	All source corpora	42.9	54.9	56.8	61.3	64.4	65.5
DA	Source corpus	*US*	*S-S*		*US*	*S-S*	
	EMO-DB	49.7	51.3		71.0	**73.2**	
	eNTERFACE	57.0	**58.0**		67.2	68.6	
	FESC	46.9	47.4		61.5	63.1	
	RAVDESS	57.1	57.7		66.5	68.4	
	All source corpora	53.2	53.5		71.0	71.3	

4.2.3. *Domain Adaptation Experiments*

For the DA-based experiments, 1-to-1 and 4-to-1 adaptation conditions were examined with the same source corpora as in CCG. All DA experiments were conducted separately for valence and arousal. In the 1-to-1 settings, each source corpus was randomly split into a training and test set in a ratio of 85:15. For the 4-to-1 setting, the training and test sets were the combination of the respective corpus-specific splits. For the first stage of the adaptation process, the training set of each source corpus was used to train M by using the Adam optimizer ($lr = 10^{-4}$), early stopping with a patience of 100 based on test set accuracy, and batch size of 256. The log-mel features were used as the input features for F, consisting of three FC layers of 512, 512, and 256 units, each followed by batch normalization. The first two layers had LReLU [34] nonlinearities and a dropout of 0.4. C_L was an NN consisting of three FC layers of 256, 256, and 2 units. The first two layers had LReLU nonlinearities and a dropout of 0.3. The last layer was followed by a softmax function. For each variant of the source data, a separate M was trained for both valence and arousal.

For the second stage of the adaptation process, the full unlabeled data from the source corpus/corpora and the unlabeled training samples of NICU-A were used for training. Following [18], the unsupervised variant of WDA was trained until the first term in Eq. 4 was saturated. For the semi-supervised variant, the labeled training set of NICU-A was used to determine the model accuracy after each epoch, and the model with the highest accuracy was selected for testing. This set was also used to find optimal hyperparameters. C_D consisted of four FC layers of 512, 512, 256, and 1 units. The first three layers were followed by ReLU nonlinearities. The parameters of C_D and F_T were updated with the RMSProp [35] and Adam optimizers, respectively. In the 1-to-1 settings, $lr = 5 \cdot 10^{-5}$ was used, except with FESC for valence and with RAVDESS for arousal, where $lr = 7 \cdot 10^{-5}$. For the 4-to-1 settings, $lr = 7 \cdot 10^{-5}$ was used for valence and $lr = 6 \cdot 10^{-5}$ for arousal. The performance of the adapted model was then tested on the GS data.

All the DA and AL parameters were based on extensive piloting with leave-one-corpus-out simulations using the source corpora, and before any NICU-A data had been labeled.

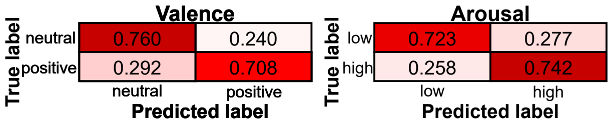


Figure 2: *Normalized confusion matrices for valence (left) and arousal (right) using the best models. Valence = SVM + GeMAPS + cluster labels from MAL (73.4% UAR). Arousal = NN + WDA using EMO-DB as the source corpus (73.2% UAR).*

5. Results

The main results are presented in Table 1. They show that AL (top rows) is the most consistent performer across the studied conditions, even though somewhat better arousal results are obtained by particular configurations of CCG and DA. The best DA-based model adaptation achieves 73.2% UAR on arousal, outperforming all other methods by a clear margin. However, adaptation from other corpora does not always work that well. In addition, CCG and DA have problems with valence classification on data from the new domain. The DA results (Table 1, bottom) are on average higher than the results of CCG, even though the WDA method does not provide a major improvement over CCG on valence. The semi-supervised variant of WDA is also consistently better than the unsupervised variant. The comparison of using either cluster or medoid labels for AL provides somewhat mixed results, depending on the exact condition.

In terms of features, the GeMAPS and eGeMAPS feature sets outperformed the log-mel features on valence with CCG. For CCG and arousal, the best-performing features varied largely between different training corpora, and the matching Finnish language FESC is a substantially better source for NICU-A than the others, reaching 70.8% UAR with eGeMAPS features. In the AL experiments (Table 1, top), the eGeMAPS and GeMAPS features achieved the best mean classification accuracy for valence and arousal, respectively.

The confusion matrices for the best-performing models (Fig. 2) indicate that these models do not systematically favor one label over the other when performing predictions.

6. Conclusions

In the present paper, we developed a SER system for large-scale analysis of emotional content of speech in initially unannotated real-life child-centered audio recordings from a NICU. CCG, AL, and DA were compared as alternatives for deploying a SER system for this novel dataset from scratch. Our results show that WGAN-based DA outperformed the baseline CCG approach, verifying its usefulness in the absence of any data labels. However, with a very moderate human labeling resource available, k-medoids based AL was superior compared to CCG and DA in valence classification and relatively competitive for arousal as well. However, when classifying arousal, DA resulted in slightly better results than AL. Overall, the results demonstrate that the earlier proposed MAL [15] and WDA [18] methods are also applicable to practical SER scenarios. The results also show that emotion analysis for LENA-based daylong audio recordings is possible with an accuracy comparable to those reported in earlier literature (e.g., 58.1% for valence and 66.8% for arousal across the multi-corpus tests in [31]).

7. Acknowledgements

OR was funded by Academy of Finland grant no. 314602 and KD partially by EU Horizon-2020 grant no. 957337 MARVEL. SAB was funded by Academy of Finland grant no. 332962. The authors thank the APPLE consortium for the help in the project.

8. References

[1] A. Batliner and B. Schuller, *Computational Paralinguistics: Emotion, Affect and Personality in Speech and Language Processing*. New York: John Wiley & Sons, Incorporated, 2013.

[2] A. Batliner, B. Schuller, D. Seppi, S. Steidl, L. Devillers, L. Vidrascu, T. Vogt, V. Aharonson, and N. Amir, "The Automatic Recognition of Emotions in Speech," in *Emotion-Oriented Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 71–99.

[3] E. Ståhlberg-Forsen, A. Aija, B. Kaasik, R. Latva, S. Ahlqvist-Björkroth, L. Toome, L. Lehtonen, and S. Stolt, "The validity of the Language Environment Analysis system in two neonatal intensive care units," *Acta Paediatrica*, 2021.

[4] B. Schuller, B. Vlasenko, F. Eyben, M. Wöllmer, A. Stuhlsatz, A. Wendemuth, and G. Rigoll, "Cross-Corpus Acoustic Emotion Recognition: Variances and Strategies," *IEEE Transactions on Affective Computing*, vol. 1, no. 2, pp. 119–131, 2010.

[5] B. Zhang, Y. Kong, G. Essl, and E. M. Provost, "f-Similarity Preservation Loss for Soft Labels: A Demonstration on Cross-Corpus Speech Emotion Recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 5725–5732, 2019.

[6] J. Deng, X. Xu, Z. Zhang, S. Frühholz, and B. Schuller, "Semisupervised Autoencoders for Speech Emotion Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 31–43, 2018.

[7] M. Abdelwahab and C. Busso, "Domain Adversarial for Acoustic Emotion Recognition," *IEEE/ACM Trans. Audio, Speech and Language Processing*, vol. 26, no. 12, p. 2423–2435, 2018.

[8] H. Sagha, J. Deng, M. Gavryukova, J. Han, and B. Schuller, "Cross lingual speech emotion recognition using canonical correlation analysis on principal component subspace," in *Proc. ICASSP*, 2016, pp. 5800–5804.

[9] J. Deng, X. Xu, Z. Zhang, S. Frühholz, and B. Schuller, "Universum Autoencoder-Based Domain Adaptation for Speech Emotion Recognition," *IEEE Signal Processing Letters*, vol. 24, no. 4, pp. 500–504, 2017.

[10] S. Latif, J. Qadir, and M. Bilal, "Unsupervised Adversarial Domain Adaptation for Cross-Lingual Speech Emotion Recognition," in *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2019, pp. 732–737.

[11] Z. Zhao and X. Ma, "Active Learning for Speech Emotion Recognition Using Conditional Random Fields," in *2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2013, pp. 127–131.

[12] M. Abdelwahab and C. Busso, "Active Learning for Speech Emotion Recognition Using Deep Neural Network," in *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2019, pp. 1–7.

[13] J. Jia, S. Zhou, Y. Yin, B. Wu, W. Chen, F. Meng, and Y. Wang, "Inferring Emotions From Large-Scale Internet Voice Data," *IEEE Transactions on Multimedia*, vol. 21, no. 7, pp. 1853–1866, 2019.

[14] W. Fan, X. Xu, X. Xing, W. Chen, and D. Huang, "LSSED: a large-scale dataset and benchmark for speech emotion recognition," *arXiv preprint arXiv: 2102.01754*, 2021.

[15] Z. Shuyang, T. Heittola, and T. Virtanen, "Active learning for sound event classification by clustering unlabeled data," in *Proc. ICASSP*, 2017, pp. 751–755.

[16] K. A. S. Immink and J. H. Weber, "Minimum Pearson Distance Detection for Multilevel Channels With Gain and/or Offset Mismatch," *IEEE Transactions on Information Theory*, vol. 60, no. 10, pp. 5966–5974, 2014.

[17] H.-S. Park and C.-H. Jun, "A Simple and Fast Algorithm for K-Medoids Clustering," *Expert Syst. Appl.*, vol. 36, no. 2, p. 3336–3341, 2009.

[18] 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.

[19] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," in *Proc. ICML*, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 214–223.

[20] D. Xu, U. Yapanel, S. Gray, J. Gilkerson, J. Richards, and J. Hansen, "Signal processing for young child speech language development," in *Proc. 1st Workshop on Child, Computer, and Interaction (WOCCI-2008)*, 2008.

[21] K. Siirilä, "Language Environment Analysis (LENA) - menetelmän validiteetti keskosvauvojen ääniympäristön arvioinnissa," Master's thesis, University of Turku, 2019.

[22] A. Cristia, M. Lavechin, C. Scaff, M. Soderstrom, C. Rowland, O. Räsänen, J. Bunce, and E. Bergelson, "A thorough evaluation of the Language Environment Analysis (LENA) system," *Behavior Research Methods*, 2020.

[23] F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, and B. Weiss, "A database of German emotional speech," in *9th European Conference on Speech Communication and Technology*, vol. 5, 2005, pp. 1517–1520.

[24] O. Martin, I. Kotsia, B. Macq, and I. Pitas, "The eNTERFACE' 05 Audio-Visual Emotion Database," in *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, 2006, pp. 1–8.

[25] M. Airas and P. Alku, "Emotions in Vowel Segments of Continuous Speech: Analysis of the Glottal Flow Using the Normalised Amplitude Quotient," *Phonetica*, vol. 63, pp. 26–46, 2006.

[26] E. Vaaras, "Automatic Emotional Speech Analysis from Daylong Child-Centered Recordings from a Neonatal Intensive Care Unit," Master's thesis, Tampere University, 2021.

[27] S. R. Livingstone and F. A. Russo, "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English," *PLoS ONE*, vol. 13, no. 5, pp. 1–35, 2018.

[28] F. Eyben, K. R. Scherer, B. W. Schuller, J. Sundberg, E. André, C. Busso, L. Y. Devillers, J. Epps, P. Laukka, S. S. Narayanan, and K. P. Truong, "The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing," *IEEE Transactions on Affective Computing*, vol. 7, no. 2, pp. 190–202, 2016.

[29] F. Eyben, F. Weninger, F. Gross, and B. Schuller, "Recent developments in openSMILE, the Munich open-source multimedia feature extractor," in *MM 2013 - Proceedings of the 2013 ACM Multimedia Conference*. ACM, 2013, pp. 835–838.

[30] B. Schuller, Z. Zhang, F. Weninger, and G. Rigoll, "Using Multiple Databases for Training in Emotion Recognition: To Unite or to Vote?" in *Proc. INTERSPEECH*, 2011, pp. 1553–1556.

[31] Z. Zhang, F. Weninger, M. Wöllmer, and B. Schuller, "Unsupervised learning in cross-corpus acoustic emotion recognition," in *2011 IEEE Workshop on Automatic Speech Recognition Understanding*, 2011, pp. 523–528.

[32] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," in *4th International Conference on Learning Representations*, 2016.

[33] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations*, 2015.

[34] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *Proc. ICML*, 2013.

[35] T. Tieleman and G. Hinton, "Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

7.7 Design Recommendations for a Collaborative Game of Bird Call Recognition based on Internet of Sound Practices

The appended paper follows.

Design Recommendations for a Collaborative Game of Bird Call Recognition Based on Internet of Sound Practices*

Emmanuel Rovithis, *AES Member*
(emrovithis@ionio.gr)
Nikolaos Moustakas *AES Member*
(a11mous@ionio.gr)
Konstantinos Vogklis
(voglis@ionio.gr)
Konstantinos Drossos
(konstantinos.drossos@tuni.fi)
Andreas Floros *AES Fellow*
(floros@ionio.gr)

Ionian University, Corfu, Greece *Tampere University, Tampere, Finland*

Citizen Science aims to engage people in research activities on important issues related to their well-being. Smart Cities aim to provide them with services that improve the quality of their life. Both concepts have seen significant growth in the last years, and can be further enhanced by combining their purposes with Internet of Things technologies that allow for dynamic and large-scale communication and interaction. However, exciting and retaining the interest of participants is a key factor for such initiatives. In this paper we suggest that engagement in Citizen Science projects applied on Smart Cities infrastructure can be enhanced through contextual and structural game elements realized through augmented audio interactive mechanisms. Our inter-disciplinary framework is described through the paradigm of a collaborative bird call recognition game, in which users collect and submit audio data, which are then classified and used for augmenting physical space. We discuss the Playful Learning, Internet of Audio Things, and Bird Monitoring principles that shaped the design of our paradigm, and analyze the design issues of its potential technical implementation.

0 INTRODUCTION

The concept of Smart Cities (SC) describes urban environments enriched with interaction modalities towards the improvement of city functioning and of its inhabitants' life [1]. Aiming to align the technological attainments of the digital era with the urban fabric of the physical world, SC utilize Information and Communication Technologies (ICT), such as mobile devices, embedded sensors, and data collection and analysis tools, and seamlessly integrate them in traditional infrastructure. Thus, the physical environment is transformed into a dynamic source of information, an "intelligent living space", which, based on the adoption of networking advances, provides citizens with the tools, resources, and services to exploit the benefits of this data flow [2]. The realization of intelligent systems to be embedded into SC environments can be broken down into three axes: i) the Internet of Things (IoT) facilitating the inter-connectivity of physical and virtual devices through communication protocols, ii) the Internet of Services (IoS) comprised of the amalgamation of different applications into explicable services, and iii) the Internet of People (IoP) encompassing the interactions between the citizens, who are ultimately the intended users of the system [3]. In order for SC to become truly beneficial innovation ecosystems capable of finding solutions to real-world problems, the citizens need to be excited in terms of creativity and collaboration [4]. To that scope, SC require applications that facilitate large-scale participatory projects, in which emphasis will be placed on the coordination of end-users towards dealing with the targeted issues [4].

The concept of Citizen Science (CS) describes projects, in which people volunteer to contribute to a scientific enquiry by gathering and managing information. The advent of the 21st century has signaled an unprecedented growth of CS initiatives bringing scientists and the public together with the aim of raising awareness and finding solutions about social and environmental issues. Volunteers can now quickly locate a project for CS on a subject of their interest and easily join its active community, whereas ad-

*Correspondence should be addressed to Emmanuel Rovithis
K. Drossos is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, project MARVEL.

vances in human-computer interaction have extended the access to such projects to groups that could not previously be reached [5]. Thus, citizens can take on a vital role in research activities that may vary from simply providing experts with the necessary information to consulting the responsible authorities or even participating in making and implementing decisions [6]. In that process, basic scientific principles must be followed, such as well designed data collection and validation methods, explicit instructions and research questions, and feedback as a reward for participation [7].

We believe that SC can host large-scale participatory CS projects in a mutually beneficial relationship, in which SC provide the technological infrastructure and CS the activities targeting the citizens' well-being. However, the design of such an endeavour must address important challenges related to participants' engagement and cognition. Regarding the former, volunteer dropout has been identified as one of the salient factors that impact the organizational consistency of CS [6, 8]. Regarding the latter, undertaking and accomplishing tasks in a CS context does not necessarily extend beyond the acquisition of knowledge to a deeper understanding of the scientific process [9]. So far, suggestions for exciting and retaining the interest of CS volunteers include providing positive reinforcement, and matching the tasks to their personal skills [6]. Cognitive impact has been approached mostly through student assessments in curriculum-based projects presenting limited evidence that participation enhances scientific knowledge and public awareness. Therefore, more evaluations are deemed necessary for extracting solid conclusions [9]. In this paper we suggest that motivation and understanding can be enhanced through an inter-disciplinary approach that combines structural and contextual game elements with Internet of Audio Thing (IoAuT) technologies [10] to realize CS projects in SC environments. We describe our recommendations for developing an appropriate design framework through the paradigm of a bird call recognition augmented reality audio-based game.

The rest of the paper is organized as follows. In Section 1 are discussed the principles underlying the paradigm's design in terms of its playful learning, audio interaction, and bird recognition aspects, Section 2 describes the conceptual and technical structure of the paradigm and Section 3 outlines the technical design specifications. Finally, Section 4 concludes the paper.

1 DESIGN PRINCIPLES

1.1 Playful Learning

Playful Learning refers to the incorporation of game elements into non-game learning environments [11].The ability to motivate players is the most frequently cited characteristic of games related to knowledge construction [12]. By utilizing a variety of interaction mechanisms games create the conditions for competition, cooperation, exploration, and reflection, and engage participants in immersive experiences [13]. Aiming to investigate the connection of motivation and engagement to the learning outcomes researchers have intensified their efforts in the last decade, whereas educators have been drawing upon the results to systematically use game-based learning practices in their classroom [12]. Non-schooling environments have been also following this trend: museums, libraries, corporations, and government agencies have been integrating game elements in personal or collective activities as the means to enrich users' experience and enhance their construction of knowledge. However, simply adding a leaderboard system based on points of progress may have negative effects, since players with low scores could become frustrated and lose their interest in the competition [14]. Similarly, stereotypical approaches will not necessarily result in increasing and sustaining participation when addressing the broader community [15]. Therefore, careful planning of game elements integrated into non-game systems is needed to ensure motivation at all times of the process.

Large-scale participatory CS projects require attention, coordination, cooperation, and commitment. The few cases, in which CS was organized in the form of a game, have delivered positive results: providing users with a playful interface and allowing them to collaborate with or compete against each other towards a common goal resulted in users coming up with novel ideas [8]. Another approach refers to CS projects, which are embedded in the form of mini-games within larger sand-box game environments, i.e. environments, in which players have freedom of action that is not restricted by a linear narrative. In the case of [16] players completing various stages of the mini-games are rewarded with in-game prizes.

Besides the motivational function, there are other game elements that can be useful. In order for CS to produce an output of equal-to-expert quality, the participants need guidance through protocols, training, and oversight [5]. A game's rules, tutorial, and feedback can address these issues respectively, whereas the addition of a compelling narrative can enhance immersion in the experience. A final issue that we considered is the link of high motivation and engagement to the intended learning outcomes. Drawing upon modern learning theories including Problem-based Learning [17], i.e. learning from the process of striving toward the resolution of a problem, Constructivist Learning [18], i.e. learning from the process of interacting with the environment, and Experiential Learning [19], i.e. learning from the process of reflecting on one's experience, provides the theoretical basis for realizing meaningful learning environments [20]. Yet, researchers stress the need for stronger evidence, before game mechanisms aiming at motivation and immersion are systematically used as the means to achieve the learning objectives [21, 22], a need that large-scale participatory projects can address.

1.2 Internet of Audio Things

IoAuT is an emerging field that refers to embedding computing devices in physical objects towards the reception, processing, and transmission of audio information [23]. It comprises different types of audio collectors, processors and transmitters, and facilitates their integration, local and remote accessibility, and multi-directional communication [10]. Despite the plethora of SC initiatives and the need for utilizing state of the art Human-Computer Interaction (HCI) technologies to realize new forms of participation, most approaches have focused on data visualisation techniques and mostly neglected the acoustic aspect of the urban environment [1, 24]. Existing SC applications of distributing information through the auditory channel include the generation of sound content based on urban related data in order to increase users' awareness about their city environment [24, 25], the generation of visual maps based on the perceptual attributes of submitted recordings for monitoring and managing the urban acoustic environment [26], and the augmentation of public spaces with audio information for engaging the audience in social experiences [27]. Furthermore, Wireless Acoustic Sensor Networks can be used for the surveillance and analysis of acoustic scenes, urban noise pollution, environmental anomalies, and wildlife [10].

In our paradigm design we focused on three specific aspects of IoAuT: i) collecting and submitting audio data for analysis, ii) generating a soundscape map, and iii) augmenting physical space with virtual audio components for navigation and interaction within the environment. Focusing on the latter, Augmented Reality Audio (ARA) systems have been applied for well-being purposes by acoustically enriching the working environment of employees [28], indicating the location of security threats [29], realizing non-visual spatial mappings for navigation [30], aurally signalling touristic points of interest [31], assigning audio recordings to locations of cultural importance [32], and aurally signifying city facilities for urban exploration [33]. Interaction in these implementations can be characterised as passive, i.e. users of the system essentially trigger sound events through their position and movement in the augmented space. However, more active modes of interaction can enhance users' communication in competitive or collaborative contexts. In [34] a positive connection was shown between challenging mechanics requiring the performance of gestures with the satisfaction gained from the experience. In [35] the behavior of the virtual sound sources that players need to locate is controlled by the movement of other antagonizing players, whereas in [36] players take up different roles and need to coordinate their actions in the augmented space to achieve the game goal. Sound recognition and audio based analytics [37, 38] can further expand the possibilities for interaction by advancing the responsiveness between the natural and the virtual acoustic environment [39], whereas user experience improvement techniques from the wider frame of AR can be utilized to enhance the system's context-awareness [40].

1.3 Bird Monitoring

The third field that we drew upon for designing our paradigm relates to Bird Monitoring. Bird related ecological projects usually fall into three categories: i)inventory, ii) monitoring, and iii) research [41]. Inventory projects aim to generate a list of species by identifying birds by visual observation and/or their song. Monitoring projects involve recording birds in a region or study site for a period of time. Such projects use geolocation information to pinpoint found birds on Geographic Information System (GIS) overlays. Research projects require experts to formalize and investigate a hypothesis about bird behaviour.

One of the leading active projects in collaborative Bird Monitoring is eBird, a project of the Cornell Lab of Ornithology [42, 43]. eBird evolved from a basic CS project into a collective enterprise through the novel approach of developing cooperative partnerships among experts in a wide range of fields including computer scientists, biologists, and data administrators. eBird data are overlaid on global GIS maps. They are openly available and constitute a major source of biodiversity data, increasing expert knowledge on the dynamics of bird species distributions and aiding the conservation of birds and their habitats. The project involves at the moment more that 100,000 registered users that deliver up-to-date results about bird populations. We suggest that future projects can motivate and retain participation through embedded game mechanisms as described in this paper.

2 PROPOSED FRAMEWORK

2.1 Scenario Design

In terms of structure our paradigm consists of four stages:

- In the first stage, users undertake the task of collecting bird songs for classification using the recording tool of the application. The recordings are checked internally in the mobile device regarding their authenticity and clarity, and, if they meet the criteria, they are matched to the corresponding bird species. Users are then provided with the respective information including the bird's name and photos.
- In the second stage, users submit their successful entry to the system's remote server along with the related meta-data including the recording's date and location. This information is used by the system for the creation of a virtual 2d map representing bird presence in the urban environment. Users receive a message informing that the map was updated to include their latest entry. The virtual map is dynamically shaped according to users' position in physical space and the meta-data gathered by all submissions.
- In the third stage, users activate the augmented version of the map. The system merges virtual components with physical space into an augmented environment, in which users can immerse. This augmentation relies purely on audio information. Essentially, the meta-data submitted by the users is periodically refreshed and translated into aural stimuli by shaping the parameters for the occurrence of sound events stored in the device. Each bird has been assigned its own sound, which acts as a symbol for the bird's presence, while at the same time demonstrates

its characteristic song for further study. Data sonification and sound spatialization techniques are used to express the targeted aspects of the aerial fauna: panning, playback volume, and playback rate are dynamically modified to hint at the birds' direction and proximity to the user, and at the amount of the submitted recordings respectively. Users interpret the audio information to navigate through the augmented urban landscape. Our suggested paradigm has a 24-hour storage cycle, through which the meta-data can be recalled on a day-by-day basis.
- In the fourth stage, users experience an artistic aspect of the project, in which all submitted recordings are streamed by the system to the users, aimed at exciting their enjoyment, engagement, and collaboration. This stage can be activated once users are within the range of an audio source. In case many recordings have been assigned to a specific location, the system performs processing and mixing of the material through reverberation and temporal allocation algorithms to distinguish the recordings from one another and create a clear and appealing virtual soundscape before superimposing it onto the real acoustic environment. Thus, users feel like they are participating in a collaborative artwork that enriches their scientific duties.

Our proposed paradigm suggests further enhancing the aforementioned stages through game elements applied on game scenarios. Thus, citizen scientists become players of a large-scale participatory game that aspires to boost their engagement through fun, challenge, attainment, competition, and collaboration.

Regarding game elements, the following are suggested:

- Level advance: players unlock different game scenarios according to their successful submissions and commitment to the project.
- Badges and rewards: players' progress is also made public through titles to be gained as rewards for their performance.
- Collaborative mode: players can work together towards achieving more complex goals that require cooperation and coordination with each other.

Regarding game scenarios, the following are suggested:

- Quiz: players are asked to recognize the bird songs to gain points for their correct answers. They can familiarize themselves with the topic by studying the stored patterns in the device, a process which enhances the project's educational aspect.
- Treasure Hunt: players focus on a specific bird and report different locations of its presence within a limited time frame. They can consult their maps (2d or augmented), as they are dynamically shaped by the submissions of other players.
- Time Travel: players follow the route of a specific bird within a certain time period by visiting past corresponding locations that are saved in the system, and thus study its migratory mobility.
- Adopt a bird: players monitor the activity of a specific bird for a certain amount of consecutive days. In the process more information about the species is disclosed, such as feeding and mating habits. This game scenario aims to engage players more deeply with caring for the subject of their study.

In terms of audio interaction, all aforementioned stages, elements, and scenarios rely on three different modes:

- Constant Listening: players are exposed to the complete augmented audio environment at any time, and can also select to isolate specific information.
- Focused Listening: players turn their device like opening a window to a specific direction and only listen to the audio information that the device is facing.
- Interactive Listening: players perform specific actions with the device, such as pressing a virtual button to record or tilting their device to activate the virtual soundscape and/or bring front specialized information.

2.2 Architecture Design

The design of our paradigm's architecture (fig. 1) is based on the four stages defined in the scenario, and involves a three-layer IoAuT setup. The Sensing Layer includes the sensors, and the recording and playback module in the user's mobile device, which allow for producing audio content and analyzing phenomena associated with auditory events. The Network Layer is responsible for data transfer from the Sensing Layer, and the Application Layer includes the web services and the virtual soundscape construction module.

Focusing on exploring the ARA environment, the architecture design segments the concept into two primary modes. The first mode is designed to facilitate passive interaction, as users walk through the real environment. After the desired filters are set through the menu of the mobile app, the sound monitoring mechanism reproduces the real acoustic environment in real-time, and the playback mechanism delivers the captured sound, when its source is in the user's proximity. All audio components are mixed together and delivered through the audio headset. An amplification coefficient, which is adjustable by the user, is applied to audio capture for improving the recognition of bird sound. Once users hear something of interest, they can enter the second mode and actively search the dynamically generated virtual 2d map or augmented audio map to locate points of interest and interact with them.

Once the preview mechanism is derived using the above procedure, audio recording is implemented into the existing capture procedure model. The user then responds to this procedure by annotating the part of the waveform, which contains the bird sound. A Convolutional Neural Network (CNN) model embedded in the mobile device checks to recognize specific bird classes. If the model classifies the annotating sound to a specific class, then the local save procedure is enabled. More specifically, the following data are stored: i) the annotation of the audio file, ii) the tag of the bird class, iii) the tag of the time of the event, including date, year, and hour, and iv) the GPS coordinates that are captured using the GPS features of the mobile device. As soon as the local saved data are ready, the final upload procedure to the web server can be made.

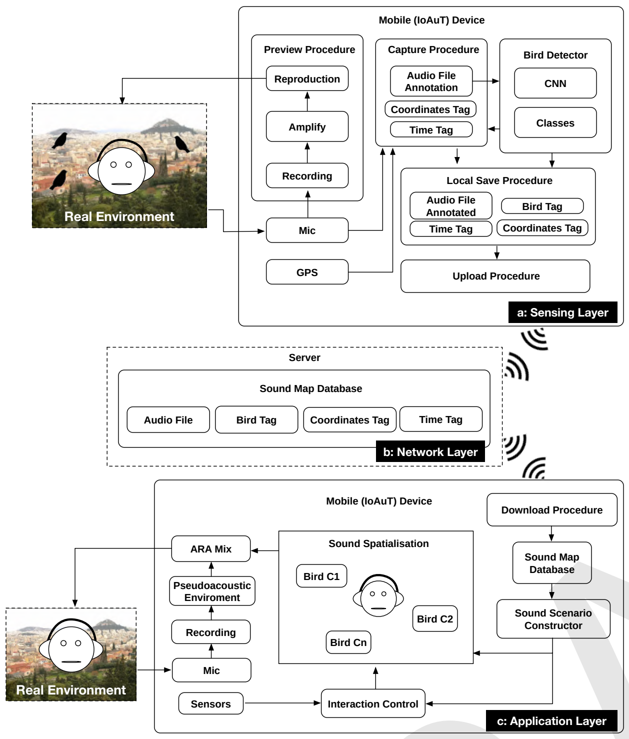


Fig. 1: The Concept Architecture: a) Sensing Layer, b) Network Layer, c) Application Layer

The Application Layer is organized in a client-server architecture. The mobile app, as the client, requests to download a sound scene by sending the phone position. The server part manages the data regarding the audio files of the annotated birds with corresponding tags of bird classification, GPS coordinates and time. The sound scene data are imported in the sound scenario constructor, which manages the processes of gestural interaction control, and the sound spatialization engine, which is responsible for placing the recordings in virtual space. The variations of sound scenario constructor and interaction controller shape each game mode.

2.3 Risk Assessment

Our paradigm stands for a preliminary approach that requires implementation and testing for evaluation and optimization. As a first step we have performed an assessment of potential risks and we suggest ways that they could be dealt with.

- Bad data: users could submit sounds that have been downloaded from the web, or recorded from other prerecorded playback. This risk can be countered by a) allowing users to make a recording only via the in-app recording tool, and b) performing a frequency range check to establish that the audio captured is a result of natural wild-life recorded in situ and not elsewhere. Special techniques used in voice anti-spoofing (see results from `https://www.asvspoof.org/`) can be also applied.
- Wildlife disturbance: naive users might disturb the birds' natural habitat in their attempt to perform the game actions. The fact that the proposed scenarios rely on capturing audio information, a process which does not require visual contact with the subject under examination, reduces that risk.
- Acoustic interference: in connection to the previous risk, there is the possibility that the playback of the application's audio content interferes with the natural acoustic environment and its inhabitants. This risk can be eliminated, if the application works only in headphones mode and does not emit sound from the speakers.
- Data overload: the streaming of too much information might cause system lag. Towards reducing that risk, the sounds used for user's navigation will be stored in the device, which will receive from the server only playback specifications. Furthermore, the streaming soundscape made from the all users' submissions, will be created periodically on the server. Each user entering the same physical area would download the same soundscape audio.

We understand that wildlife disturbance is a complex and sensitive issue and poses a major challenge in the gamification of the bird recognition process. Careful design must be applied to ensure that a set of appropriate instructions regarding user behavior is clearly communicated without thwarting the application's game aspect. Furthermore, an expert evaluation [44] is intended to take place and provide valuable insight towards the prevention of possible negative consequences.

3 TECHNICAL DESIGN SPECIFICATIONS

3.1 Audio Capture

The capture of the real acoustic environment is done by means of the sound recording features of the device. The user defines the appropriate recording option according to their equipment. The available options include mono and binaural recording technology. The capture procedure also involves outputting the monitor audio to the default playback device. Mono recording uses modules that are typically available in almost all modern smartphones: the built-in microphone for detecting sounds of interest, and a set of headphones as playback acoustic equipment. The binaural recording option is performed using in-ear microphones embedded on a stereo headset like Sennheiser Ambeo Smart Headset[1]. In both options, there is an optional

[1]Sennheiser AMBEO Smart Headset-Mobile binaural recording headset, URL `https://en-de.sennheiser.com/in-ear-headphones-3d-audio-ambeo-smart-headset`, (Accessed on 03/19/2021)

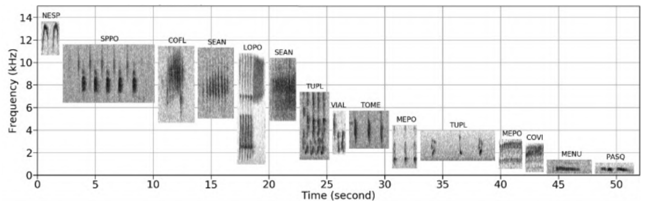


Fig. 2: Spectrograms of 14 distinct tropical birds from Puerto Rico. Taken from [45]

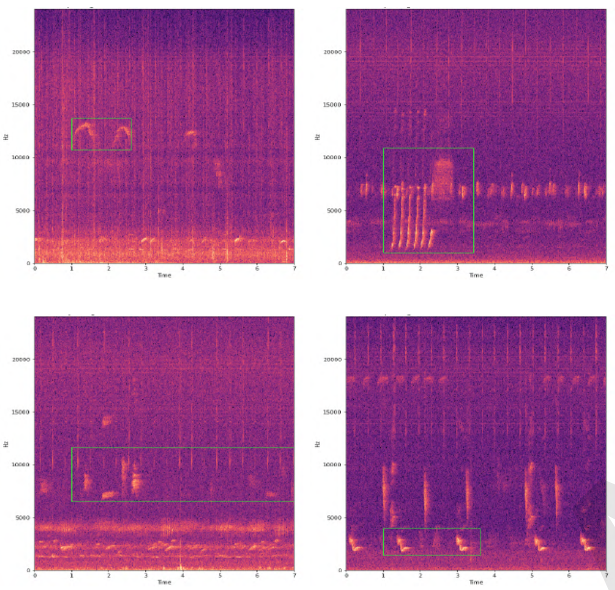


Fig. 3: Time-frequency representation of 4 distinct bird sounds from [45], with an annotation bounding box created by an expert

gain control of the environmental monitoring system for personalized sound detection procedure (fig.1).

3.2 Bird Call classification

Automatic bird sound classification plays an important role in monitoring and protecting biodiversity. Recent advances in machine listening and deep learning models for bird audio detection provide a novel way for improving bird call recognition to expert level. The fact that inter-species bird sounds exhibit such a distinctive spectral structure, motivated researches to employ typical hand-crafted time-frequency representations as an input to various deep learning models. The most prominent one is the usage of mel-scaled band energies as the time-frequency representation, which is given as an input to 2D CNNs [45]. Figure 2 shows the distinctive spectral structure of 14 tropical birds.

The most recent example of a Bird audio classification model is BirdNet, introduced by Cornell University [46, 47]. It involves a ResNet-like CNN model, containing 127 layers and 27 million parameters, and capable of identifying 984 North American and European bird species by sound. BirdNet achieves a Mean Average Precision (mAP) of 0.791 for single-species recordings. However, typical CNNs cannot model long temporal dependencies that are usually needed in machine listening tasks, such as bird-audio detection [48, 37]. To address this issue, different published papers have adopted the Convolutional Recurrent Neural Network (CRNN) model [48, 49]. The CRNN model consists of a series of 2D CNNs, followed by Recurrent Neural Networks (RNNs) and a linear layer. A time-frequency representation of audio is given as an input to the a CRNN model, and the 2D CNNs learn time-frequency patterns according to the targeted task (e.g. bird audio detection). Then, the RNNs take as input the output of the CNNs, and focus on learning temporal patterns. Finally, the linear layer is fed the output of the RNNs and performs the classification. The CRNN model achieved high performance in DCASE Challenge tasks on bird-audio detection, ranking among the top 5 systems [48, 50].

3.2.1 Deep learning workflow

In order to implement the bird call classification module the typical workflow for training image deep learning models will be followed. The first important step is to record the primary data sources, while keeping extensive metadata about time and location and recording quality. Sound data must be carefully curated by experts to reflect also the background sounds of the field under surveillance. The next steps comprise standard Deep Learning pipelines. CNNs are applied to classify time-frequency visualization of bird sounds (see Figure 3), using a hold out validation set to control the generalization to unknown cases. There are two major modeling options: i) clip-wise annotation/inference, where the model classifies a bunch of seconds at once, and ii) frame-wise annotation/inference, where the model outputs prediction for each discrete time step (e.g. milliseconds).

3.2.2 Delivering the model

A trained CNN model is delivered to the proposed application in two ways: i) recognition as a service where a back-end server (powered by GPU) is used to accept audio chunks, perform the pre-processing and the inference, and report the classification result and ii) recognition on the edge-device, where all process is performed on the user side. Each serving paradigm has pros and cons. Recognition as a service needs network access and a powerful back-end server, whereas recognition on the edge can accommodate relatively small sized models and has no network requirements.

3.3 GIS based repository

Since bird call detection will be accompanied by GPS based geographical coordinate and time stamp, all detection data can be presented in the form of cartographic GIS data model. This will allow users to easily locate their findings and the findings of others. GIS data can be aggregated by zoom levels and give finer grain detection the further the user zooms in. The basic user interface screen could be a map with overlaid information about the user's current geo-location and already existing bird call detection. A side effect of this online collaborative GIS repository would be the extraction of migratory journeys of species of bird as well as the abundance of specific species.

3.4 Augmented Reality Audio

The ARA environment consists of a real and a virtual acoustic component with the real sound recording being mixed with the spatial reproduction of the classified captured birds' sounds into a pseudoacoustic environment, a mix that needs to take place as seemlessly as possible. The importance of a dynamic ARA mix of the gain difference between the real and the virtual acoustic environment compared to the static mix gain in legacy ARA mix models [51] has been pointed out in [52]. The comparison of the legacy and the adaptive ARA mix model has shown that the latter demonstrated significantly better performance in terms of auditory perception [53]. Thus, in the proposed framework, we employ this dynamic and adaptive ARA mixing strategy that focuses on the impact of dynamic fluctuations of the real and the virtual environment to acoustic perception, taking in consideration acoustic phenomena, such as auditory masking.

Furthermore, the location awareness in ARA systems refers to the capability of a device to determine its location in terms of coordinates through active or passive human-computer interaction. Several ARA works have shown the necessity to utilize spatialization techniques, in order to combine data extracted from location awareness systems with virtual sound sources [28, 54]. Our proposed system includes a spatialization module for positioning the 3d virtual sound sources, a set of sensors including gyroscope, accelerometer, and GPS, that facilitates gestural interaction, and a headset for reproducing the augmented acoustic environment. With this setup users are free to move their head in both horizontal and median plane, and listen to the entire 3d acoustic space, while transmitting their location, movement, and gestural activity to the system's engine.

4 CONCLUSION

We have presented a framework for enriching a Citizen Science project in a Smart City environment with game elements using Internet of Sound technologies. The aim of our inter-disciplinary approach is to seek ways to enhance the public's motivation for participation, engagement in the experience, and deeper understanding of the subjects and processes at hand. We focused on Bird Call Recognition and Monitoring collaborative activities, in which participants can report and study the state and flux of the urban aerial ecosystem in the context of playful scenarios. In accordance with modern learning theories we propose the use of game elements including targeted quests, structured levels, and progress points and badges. User interaction relies on Internet of Audio Things mechanisms including recording and submitting audio data, and exploring the augmented environment through GIS-related navigation and gestural performance with the mobile device. Sound classification takes place through a CNN network, and the augmented soundscape is constructed by an adaptive ARA mixing system.

A core aspect of CS and SC is to focus on citizens' problems and needs. The users of our proposed system are seen, on the one hand, as active units that exploit the benefits of the enhanced world around them, and, on the other, as interconnected members of the community that collaborate with each other to improve that world. SC facilitate a safe environment to observe, reflect, and experiment, whereas CS provides with specific problems to solve and thus contribute to the scientific community. We hope that our proposed framework will serve as future reference towards enhancing the appeal of CS to the involved stakeholders, and providing novel ways to realize personal and collective interactive experiences based on a network of audio devices able to collect, evaluate, process and distribute acoustic data in urban environments.

5 REFERENCES

[1] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, Y. Portugali, "Smart cities of the future," *The European Physical Journal Special Topics*, vol. 214, no. 1, pp. 481–518 (2012), doi:10.1140/EPJST/E2012-01703-3.

[2] A. Urbieta, A. González-Beltrán, S. B. Mokhtar, M. A. Hossain, L. Capra, "Adaptive and context-aware service composition for IoT-based smart cities," *Future Generation Computer Systems*, vol. 76, pp. 262–274 (2017), doi:10.1016/j.future.2016.12.038.

[3] J. M. Hernández-Muñoz, J. B. Vercher, L. Muñoz, J. A. Galache, M. Presser, L. A. H. Gómez, J. Pettersson, "Smart cities at the forefront of the future internet." presented at the *Future internet assembly*, pp. 447–462 (2011), doi:10.1007/978-3-642-20898-0_32.

[4] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, A. Oliveira, "Smart cities and the future internet: Towards cooperation frameworks for open innovation," presented at the *The future internet assembly*, pp. 431–446 (2011), doi:10.1007/978-3-642-20898-0_31.

[5] R. Bonney, J. L. Shirk, T. B. Phillips, A. Wiggins, H. L. Ballard, A. J. Miller-Rushing, J. K. Parrish, "Next steps for citizen science," *Science*, vol. 343, no. 6178, pp. 1436–1437 (2014), doi:10.1126/science.1251554.

[6] C. C. Conrad, K. G. Hilchey, "A review of citizen science and community-based environmental monitoring: issues and opportunities," *Environmental monitoring and assessment*, vol. 176, no. 1, pp. 273–291 (2011), doi:10.1007/s10661-010-1582-5.

[7] J. Silvertown, "A new dawn for citizen science," *Trends in ecology & evolution*, vol. 24, no. 9, pp. 467–471 (2009), doi:10.1016/j.tree.2009.03.017.

[8] E. Hand, "Citizen science: People power," *Nature News*, vol. 466, no. 7307, pp. 685–687 (2010), doi:10.1038/466685a.

[9] R. Bonney, T. B. Phillips, H. L. Ballard, J. W. Enck, "Can citizen science enhance public understanding of science?" *Public Understanding of Science*, vol. 25, no. 1, pp. 2–16 (2016), doi:10.1177/0963662515607406.

[10] L. Turchet, G. Fazekas, M. Lagrange, H. S. Ghadikolaei, C. Fischione, "The Internet of Audio Things: State of the Art, Vision, and Challenges," *IEEE internet of things journal*, vol. 7, no. 10, pp. 10233–10249 (2020), doi:10.1109/JIOT.2020.2997047.

[11] J. L. Plass, B. D. Homer, C. K. Kinzer, "Foundations of game-based learning," *Educational Psychologist*, vol. 50, no. 4, pp. 258–283 (2015), doi:10.1080/00461520.2015.1122533.

[12] C. Costa, K. Tyner, S. Henriques, C. P. G. Sousa, "A Review of Research Questions, Theories and Methodologies for Game-Based Learning," *Journal of Content, Community and Communication*, vol. 4 (2016).

[13] D. Dicheva, C. Dichev, G. Agre, G. Angelova, "Gamification in education: A systematic mapping study," *Journal of Educational Technology & Society*, vol. 18, no. 3, pp. 75–88 (2015).

[14] T. Reiners, L. C. Wood, J. Dron, "From chaos towards sense: A learner-centric narrative virtual learning space," in *Gamification for human factors integration: Social, education, and psychological issues*, pp. 242–258 (IGI Global) (2014), doi:10.4018/978-1-4666-5071-8.CH015.

[15] S.-K. Thiel, M. Reisinger, K. Röderer, P. Fröhlich, "Playing (with) democracy: A review of gamified participation approaches," *JeDEM-eJournal of eDemocracy and Open Government*, vol. 8, no. 3, pp. 32–60 (2016), doi:10.29379/JEDEM.V8I3.440.

[16] CCP, "Project Discovery: citizen science begins with you," (2020), URL `https://www.eveonline.com/discovery`.

[17] D. Boud, G. Feletti, *The challenge of problem-based learning* (Psychology Press) (1997).

[18] T. M. Duffy, D. H. Jonassen, "Constructivism: New implications for instructional technology?" *Educational Technology*, vol. 31, no. 5, pp. 7–12 (1991 May).

[19] D. A. Kolb, *Experiential learning: Experience as the source of learning and development* (FT press) (2014).

[20] K. Kiili, *On Educational Game Design: Building Blocks of Flow Experience*, Tampere University of Technology. Publication (Tampere University of Technology) (2005 Dec.), awarding institution:Tampere University of Technology¡br/¿Submitter:Made available in DSpace on 2008-09-24T10:58:32Z (GMT). No. of bitstreams: 1 kiili.pdf: 2269572 bytes, checksum: a198e0fd0dec407e8b1f9aecfdeb1fb9 (MD5) Previous issue date: 2005-12-14.

[21] C. Perrotta, G. Featherstone, H. Aston, E. Houghton, "Game-based learning: Latest evidence and future directions," *Slough: NFER* (2013).

[22] J. Hamari, D. J. Shernoff, E. Rowe, B. Coller, J. Asbell-Clarke, T. Edwards, "Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning," *Computers in human behavior*, vol. 54, pp. 170–179 (2016), doi:0.1016/J.CHB.2015.07.045.

[23] L. Turchet, C. Fischione, G. Essl, D. Keller, M. Barthet, "Internet of musical things: Vision and challenges," *IEEE Access*, vol. 6, pp. 61994–62017 (2018), doi:10.1109/ACCESS.2018.2872625.

[24] P. Sarmento, O. Holmqvist, M. Barthet, "Musical Smart City: Perspectives on Ubiquitous Sonification," *arXiv preprint arXiv:2006.12305* (2020).

[25] K. Drossos, A. Floros, N.-G. Kanellopoulos, "Affective Acoustic Ecology: Towards Emotionally Enhanced Sound Events," presented at the *Proceedings of the 7th Audio Mostly Conference: A Conference on Interaction with Sound*, AM '12, pp. 109–116 (2012), doi:10.1145/2371456.2371474, URL `http://doi.acm.org/10.1145/2371456.2371474`.

[26] J. Kang, F. Aletta, E. Margaritis, M. Yang, "A model for implementing soundscape maps in smart cities," *Noise Mapping*, vol. 5, no. 1, pp. 46–59 (2018), doi:10.1515/noise-2018-0004.

[27] P. K. Nikolic, H. Yang, "Designing playful cities: audio-visual metaphors for new urban environment experience," *Mobile Networks and Applications*, pp. 1–7 (2020), doi:10.1007/s11036-020-01514-6.

[28] E. D. Mynatt, M. Back, R. Want, R. Frederick, "Audio Aura: Light-weight audio augmented reality," presented at the *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pp. 211–212 (1997), doi:10.1145/263407.264218.

[29] V. Sundareswaran, K. Wang, S. Chen, R. Behringer, J. McGee, C. Tam, P. Zahorik, "3D audio augmented reality: implementation and experiments," presented at the *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pp. 296–297 (2003), doi:10.1109/ISMAR.2003.1240728.

[30] S. Holland, D. R. Morse, H. Gedenryd, "Audio-GPS: Spatial audio navigation with a minimal attention interface," *Personal and Ubiquitous computing*, vol. 6, no. 4, pp. 253–259 (2002), doi:10.1007/s007790200025.

[31] D. McGookin, S. Brewster, P. Priego, "Audio bubbles: Employing non-speech audio to support tourist wayfinding," presented at the *International Conference on Haptic and Audio Interaction Design*, pp. 41–50 (2009), doi:10.1007/978-3-642-04076-4_5.

[32] J. Reid, E. Geelhoed, R. Hull, K. Cater, B. Clayton, "Parallel worlds: immersion in location-based experiences," presented at the *CHI'05 extended abstracts on Human factors in computing systems*, pp. 1733–1736 (2005), doi:10.1145/1056808.1057009.

[33] J. R. Blum, M. Bouchard, J. R. Cooperstock, "What's around me? Spatialized audio augmented reality for blind users with a smartphone," presented at the *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pp. 49–62 (2011), doi:10.1007/978-3-642-30973-1_5.

[34] E. Rovithis, N. Moustakas, A. Floros, K. Vogklis, "Audio Legends: Investigating Sonic Interaction in an Augmented Reality Audio Game," *Multimodal Technologies and Interaction*, vol. 3, no. 4, p. 73 (2019), doi:10.3390/mti3040073.

[35] N. Moustakas, A. Floros, N. Grigoriou, "Interactive audio realities: An augmented/mixed reality audio game

prototype," presented at the *Audio Engineering Society Convention 130* (2011 may).

[36] R. Pellerin, N. Bouillot, T. Pietkiewicz, M. Wozniewski, Z. Settel, E. Gressier-Soudan, J. R. Cooperstock, "Soundpark: Exploring ubiquitous computing through a mixed reality multi-player game experiment," *Studia Informatica Universalis*, vol. 8, no. 3, p. 21 (2009).

[37] K. Drossos, S. I. Mimilakis, S. Gharib, Y. Li, T. Virtanen, "Sound Event Detection with Depthwise Separable and Dilated Convolutions," presented at the *2020 International Joint Conference on Neural Networks (IJCNN)* (2020 Jul.), doi:10.1109/IJCNN48605.2020.9207532.

[38] K. Drossos, S. Lipping, T. Virtanen, "Clotho: An Audio Captioning Dataset," presented at the *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020 May), doi: 10.1109/ICASSP40776.2020.9052990.

[39] V. Pulkki, S. Delikaris-Manias, A. Politis, *Parametric time-frequency domain spatial audio* (Wiley Online Library) (2018 October), doi:10.1002/9781119252634.

[40] J. Aliprantis, G. Caridakis, "A survey of augmented reality applications in cultural heritage," *International Journal of Computational Methods in Heritage Science (IJCMHS)*, vol. 3, no. 2, pp. 118–147 (2019), doi: 10.4018/IJCMHS.2019070107.

[41] M. E. Hostetler, "Bird Monitoring Projects for Youth: Leader's Guide1," (2002 October).

[42] B. L. Sullivan, C. L. Wood, M. J. Iliff, R. E. Bonney, D. Fink, S. Kelling, "eBird: A citizen-based bird observation network in the biological sciences," *Biological conservation*, vol. 142, no. 10, pp. 2282–2292 (2009), doi: 10.1016/J.BIOCON.2009.05.006.

[43] B. L. Sullivan, J. L. Aycrigg, J. H. Barry, R. E. Bonney, N. Bruns, C. B. Cooper, T. Damoulas, A. A. Dhondt, T. Dietterich, A. Farnsworth, *et al.*, "The eBird enterprise: an integrated approach to development and application of citizen science," *Biological Conservation*, vol. 169, pp. 31–40 (2014), doi:10.1016/J.BIOCON.2013.11.003.

[44] H. Petrie, N. Bevan, "The Evaluation of Accessibility, Usability, and User Experience." *The universal access handbook*, vol. 1, pp. 1–16 (2009), doi:10.1201/9781420064995-c20.

[45] J. LeBien, M. Zhong, M. Campos-Cerqueira, J. P. Velev, R. Dodhia, J. L. Ferres, T. M. Aide, "A pipeline for identification of bird and frog species in tropical soundscape recordings using a convolutional neural network," *Ecological Informatics*, vol. 59, p. 101113 (2020), doi: https://doi.org/10.1016/j.ecoinf.2020.101113.

[46] M. Arif, R. Hedley, E. Bayne, "Testing the Accuracy of a birdNET, Automatic bird song Classifier," *ERA - University of Alberta's open access digital archive* (2020 July), doi:10.7939/R3-6KHB-KZ18.

[47] S. Kahl, C. M. Wood, M. Eibl, H. Klinck, "BirdNET: A deep learning solution for avian diversity monitoring," *Ecological Informatics*, vol. 61, p. 101236 (2021), doi:10.1016/j.ecoinf.2021.101236.

[48] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," presented at the *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1744–1748 (2017), doi:10.23919/EUSIPCO.2017.8081508.

[49] E. Çakir, G. Parascandolo, T. Heittola, H. Huttunen, 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), doi:10.1109/TASLP.2017.2690575.

[50] S. Adavanne, K. Drossos, E. Çakir, T. Virtanen, "Stacked convolutional and recurrent neural networks for bird audio detection," presented at the *2017 25th European signal processing conference (EUSIPCO)*, pp. 1729–1733 (2017), doi:10.23919/EUSIPCO.2017.8081505.

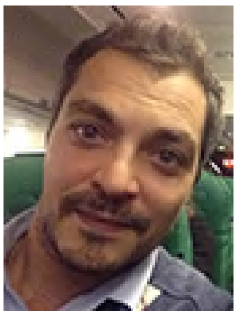
[51] M. Karjalainen, V. Riikonen, M. Tikander, "An augmented reality audio mixer and equalizer," presented at the *Audio Engineering Society Convention 124* (2008 May).

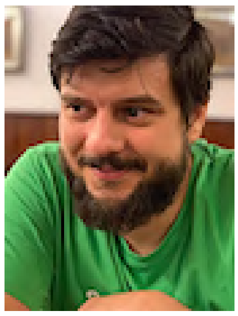
[52] N. Moustakas, A. Floros, E. Rovithis, K. Vogklis, "Augmented audio-only games: A new generation of immersive acoustic environments through advanced mixing," presented at the *Audio Engineering Society Convention 146* (2019 March).

[53] N. Moustakas, E. Rovithis, K. Vogklis, A. Floros, "Adaptive Audio Mixing for Enhancing Immersion in Augmented Reality Audio Games," presented at the *Companion Publication of the 2020 International Conference on Multimodal Interaction*, pp. 220–227 (2020), doi:10.1145/3395035.3425325.

[54] Y. Vazquez-Alvarez, I. Oakley, S. A. Brewster, "Auditory display design for exploration in mobile audio-augmented reality," *Personal and Ubiquitous computing*, vol. 16, no. 8, pp. 987–999 (2012), doi: 10.1007/s00779-011-0459-0.

THE AUTHORS





Dr. Emmanouel Rovithis was born in Athens, Greece in 1978. He holds an MA in Music Composition from the Anglia Polytechnic University in Cambridge, UK., and a PhD (first-class honours) in Electronic Music Composition from the Department of Music Studies of the Ionian University in Corfu, Greece. He is currently employed as Laboratory Research and Teaching Staff at the Department of Audio & Visual Arts of the Ionian University, teaching subjects related to Art and Technology in Education, Digital Signal Processing, and Music Programming. His research focuses on the design of Audio Games, and Augmented Reality Audio applications for educational, entertaining, and creative purposes. He has designed numerous interactive installations, educational games and software, seminars and workshops on behalf of prominent cultural institutions, and has a strong background in music composition and sound design for theatre and cinema.

●

Nikolaos Moustakas was born in Piraeus, Greece in 1987. He received his Master Degree in Audiovisual Arts from the department of Audio & Visual Arts, Ionian University in 2011, and in 2021 his PhD from the same department. His research focus on digital audio signal processing, adapting techniques, auditory perception and mixed reality environments. He was involved in the development of augmented reality audio technologies and audio-only games.

●

Konstantinos Vogklis was born in Ioannina, Greece, in 1978. He received the diploma degree in Computer Science from the University of Ioannina, Greece, in 1999, and M.Sc. and Ph.D degrees in computer science, in 2002 and 2010 respectively, from the same department. His research interests include the development of high-performance parallel algorithms and machine learning techniques with emphasis on big data, image and sound processing. In the last couple of years he has been involved in the design and implementation of augmented reality applications on mobile devices aiming on both entertainment and the emergence of cultural heritage. His published work includes 30 papers in peer-reviewed scientific journals and conferences.

●

Dr. Konstantinos Drossos was born in Thessaloniki, Greece. He holds a BEng in Sound Technology (first-class honours), a BSc in Informatics, an MSc in Sound & Vibration Research, and a PhD (first-class honours) in the field of machine listening. Currently, he is a senior researcher at the Audio Research Group (ARG), Finland. He has been a postdoc researcher at ARG and a postdoc fellow at Montreal Institute for Learning Algorithms, Canada, and at Music Technology Group, Spain. He has authored over 45 research papers, has pioneered the field of audio captioning, has organized scientific challenges, special sessions, and workshops in international conferences, serves as a reviewer for top journals and conferences, and has served as the Chairman of the Finnish IEEE Joint Chapter of SP&CAS. His research interests include audio captioning, domain adaptation, multimodal translation, source separation, detection and classification of acoustic scenes and events, and machine listening.

●

Andreas Floros was born in Drama, Greece in 1973. In 1996 he received his engineering degree from the department of electrical and computer engineering, University of Patras, and in 2001 his Ph.D. degree from the same department. His research was mainly focused on digital audio signal processing and conversion techniques for all-digital power amplification methods. He was also involved in research in the area of acoustics. H serves as Professor of Audio Technology and Electroacoustics at the department of Audiovisual Arts, Ionian University. His current research interests focus on Analysis, processing and conversion of digital audio signals, intelligent digital audio effects and sound synthesis, creative intelligence, audio-only games, auditory interfaces and displays, augmented reality audio foundations and applications as well as the investigation of the impact of sound events to human emotions.

7.8 Multi-Exit Vision Transformer for Dynamic Inference

The appended paper follows.

Multi-Exit Vision Transformer for Dynamic Inference

Arian Bakhtiarnia
arianbakh@ece.au.dk
Qi Zhang
qz@ece.au.dk
Alexandros Iosifidis
ai@ece.au.dk

DIGIT,
Department of Electrical and Computer Engineering,
Aarhus University,
Aarhus, Denmark

Abstract

Deep neural networks can be converted to multi-exit architectures by inserting early exit branches after some of their intermediate layers. This allows their inference process to become dynamic, which is useful for time critical IoT applications with stringent latency requirements, but with time-variant communication and computation resources. In particular, in edge computing systems and IoT networks where the exact computation time budget is variable and not known beforehand. Vision Transformer is a recently proposed architecture which has since found many applications across various domains of computer vision. In this work, we propose seven different architectures for early exit branches that can be used for dynamic inference in Vision Transformer backbones. Through extensive experiments involving both classification and regression problems, we show that each one of our proposed architectures could prove useful in the trade-off between accuracy and speed.

1 Introduction

Deep neural networks have achieved immense success in recent years [15], however, they commonly consist of many interconnected layers containing millions of parameters which require high computational resources and cause slow inference speed. Dynamic inference methods [8] allow deep models to modify their computation graph during inference in order to alleviate this problem. One such method is *early exiting* [20, 21], leading to *multi-exit architectures*, where early exit *branches* are inserted after intermediate hidden layers of the *backbone* network and provide early results, albeit with less accuracy compared to the final result of the backbone network.

Early exits are useful in computationally restricted settings such as mobile and edge computing, where early results can be used for "easy" inputs to save resources. Additionally, multi-exit architectures can be helpful in *anytime prediction* settings where the inference process may be interrupted at any time and the network is expected to provide a response even if it was interrupted before completion. Examples of anytime prediction settings are distributed environments such as edge computing systems and IoT networks, where the latency depends on the communication channels, which means the exact computation time budget is not known beforehand and varies over time. Here, the latest result provided by a multi-exit architecture can be given as output whenever the network is interrupted.

Vision Transformer [4] is a recently proposed architecture for computer vision which has since been applied to various problems, such as image classification, object detection, depth estimation, and many more [12]. To the best of our knowledge, multi-exit Vision Transformer architectures have not yet been studied in the literature, which limits the application of Vision Transformers in mobile and edge computing. In this work, we propose seven different architectures for early exit branches that can be inserted into Vision Transformer backbones. Through extensive experiments on both image classification and crowd counting, the latter being a regression problem, we show that depending on the particular problem at hand, each of these architectures has the potential to be useful in the trade-off between classification accuracy and inference speed. Our code will be made publicly available at `https://gitlab.au.dk/maleci/multiexitvit`.

2 Related Work

2.1 Multi-Exit Architectures

A deep neural network (DNN) can be formulated as a function $f(X) = f_L(f_{L-1}(...f_1(X)))$ where X is the input, L is the number of layers in the DNN and f_i is the differentiable operator at layer i. The output of layer i is denoted by $h_i = f_i(h_{i-1})$ and θ_i refers to the trainable parameters of $f_i(\cdot)$. The training process for this DNN can be formulated as shown in Equation (1) where $\theta = \bigcup_{i=1}^{L} \theta_i$ is the set of all trainable parameters of the DNN, $\{X_n, y_n\}_{n=1}^{N}$ is the set of training samples and $l(\cdot)$ is a loss function.

$$f^* = \arg\min_{\theta} \sum_{n=1}^{N} l(y_n, f(X_n)) \tag{1}$$

In order to convert a DNN to a multi-exit architecture, an early exit branch $c_b(h_b) = y_b$ is placed at every selected branch location $b \in B \subseteq \{1,..,L\}$, where c_b is the classifier or regressor producing the early result y_b. The schematic illustration of a multi-exit architecture is shown in Figure 1 (a). Since there are multiple outputs in a multi-exit architecture, its training procedure is not as straightforward as Equation (1). Three major strategies for training multi-exit architectures exist in the literature [21]. The *classifier-wise* strategy freezes the backbone, meaning the parameters θ will not be modified, and trains the branches separately and independent of each other or the backbone. In the *end-to-end* strategy, the loss function $l_t = l + \sum_{b \in B} \lambda_b l_b$ combines the losses l_b of the early exit branches with the backbone's loss and trains the entire multi-exit architecture simultaneously. In this strategy, the contribution of the loss of the branch at location b is captured by weight score λ_b. Finally, the *layer-wise* strategy first trains the layers up to and including the first early exit branch. Subsequently, the previous layers are frozen and the rest of the layers up to and including the second branch are trained, and this operation is repeated until the entire backbone has been trained.

In the end-to-end and layer-wise strategies, the number of branches and their placement create trade-offs between the accuracy of different exits. In addition, with the end-to-end strategy, the weight scores introduce new hyper-parameters. In contrast, no trade-offs or new hyper-parameters need to be considered with the classifier-wise strategy. However, since in this case the parameters of the backbone remain unchanged, fewer parameters are affected during the training of the branches. In this work we investigate all three training strategies.

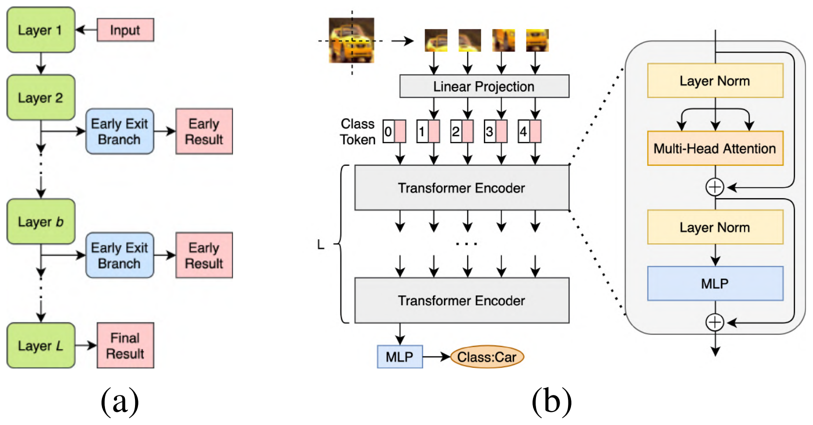


Figure 1: (a) Schematic illustration of a multi-exit; and (b) Vision Transformer architecture.

It is important to note that branches placed later in the networks do not necessarily result in a higher accuracy compared to previous branches. We use the term *impractical* in order to refer to such branches, and the term *practical* for branches with a higher accuracy than all previous branches. The usage of impractical branches would not be sensible since earlier branches with a higher accuracy exist.

2.2 Vision Transformer

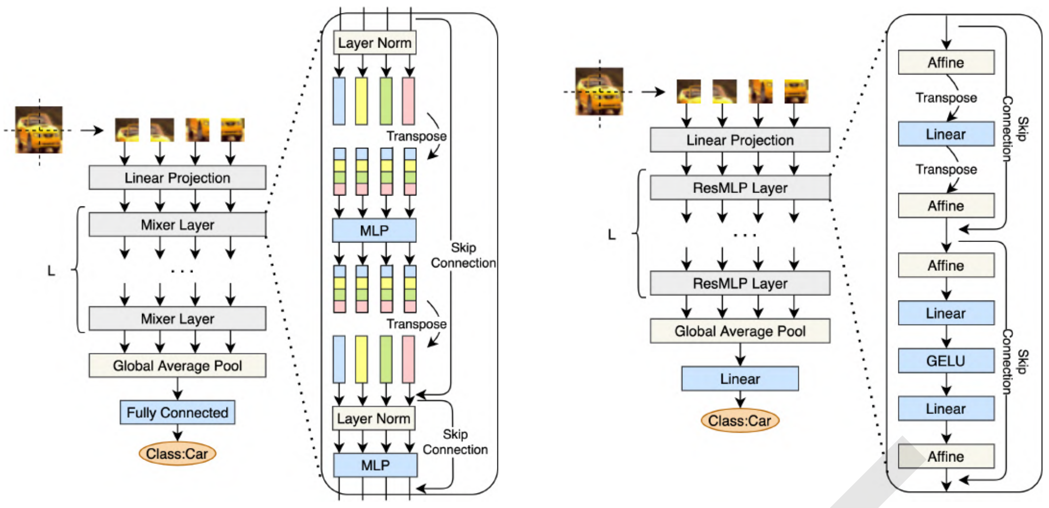
Vision Transformer (ViT) [4] is an adaptation of the Transformer architecture [25] for computer vision problems. At the core of the Transformer is the *self-attention* layer, which takes a sequence $X = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$ as input and outputs the sequence $Z = (z_1, \ldots, z_n) \in \mathbb{R}^{n \times d_v}$, which can be formulated as Equation (2), where $Q = XW^Q$, $K = XW^K$ and $V = XW^V$ are query, key and value matrices, respectively, in which W^Q, W^K and W^V are learnable weight matrices [4]. $d_k = d_q$ are the size of the vectors in query and key matrices.

$$Z = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

In order to capture more than one type of relationship between the entities in the sequence, self-attention is extended to *multi-head attention* by concatenating the output of several self-attention blocks, each with its own set of learnable parameters. Figure 1 (b) depicts the Vision Transformer architecture, where initially an input image is cut into several image patches. A sequence of patch embeddings is then formed by projecting each patch and concatenating a positional embedding to the resulting vector. An extra learnable *classification token* is also appended to the sequence. The sequence passes through L Transformer encoder layers, each containing multi-head attention layers among other operations. Finally, the output vector corresponding to the classification token is passed on to an MLP dubbed *classification head* to obtain the final result.

2.3 Attention-Free, MLP-Based Architectures

Several MLP-based architectures for computer vision that also operate on sequences of image patches have been recently proposed [6]. The aim of these architectures is to reduce the computational cost of ViT by removing the attention mechanism, while achieving a comparable performance by preserving a global receptive field similar to that of ViT. Since the



(a) (b)

Figure 2: (a) MLP-Mixer architecture; and (b) ResMLP architecture.

intermediate representations in the hidden layers of ViT is in the form of a sequence of patches, it is simple to use the building blocks of these MLP-based architectures as early exit branches placed on ViT backbones. These building blocks create more lightweight branches compared to the Transformer encoders in ViT.

One such architecture called *MLP-Mixer* [23] is shown in Figure 2 (a). Each *mixer layer* in MLP-Mixer consists of *token mixing* and *channel mixing* operations, which are formulated as Equations (3a) and (3b), where $f_1(\cdot)\dots f_4(\cdot)$ are linear layers and $\sigma(\cdot)$ is the GELU activation function. The output of the final mixer layer is passed on to a global average pooling layer and then a fully connected layer.

$$U = X + f_2(\sigma(f_1(Norm(X)^T)))^T \tag{3a}$$
$$Y = U + f_4(\sigma(f_3(Norm(U)))) \tag{3b}$$

A similar architecture called *ResMLP* [24] is shown in Figure 2 (b). Each *ResMLP layer* consists of a *cross-patch sublayer* and a *cross-channel sublayer*, which are formulated as Equations (4a) and (4b). In ResMLP, normalization is carried out using an affine transformation instead of layer normalization, as shown in Equation (4c) where α and β are learnable vectors that scale and shift the input. Similarly, the output of the final ResMLP layer is passed on to a global average pooling layer and then a fully connected layer.

$$U = X + Norm(f_1(Norm(X)^T)^T) \tag{4a}$$
$$Y = U + Norm(f_3(\sigma(f_2(Norm(U))))) \tag{4b}$$
$$Norm(X) = Aff_{\alpha,\beta}(X) = Diag(\alpha)X + \beta \tag{4c}$$

3 Multi-Exit Vision Transformer

We assume a high-performing ViT backbone is available for the problem at hand, and the goal is to convert this backbone to a multi-exit architecture in order to allow for dynamic inference. We propose seven different architectures for the early exit branches added after intermediate layers of a ViT backbone. The most intuitive approach, which we call *MLP-EE*, is to add an MLP to the classification token of the intermediate layer, similar to the classification head in the ViT backbone. Even though MLP-EE is very lightweight, it may not contain enough parameters and layers to extract useful features, particularly for exits placed early. Moreover, it does not process tokens other than the classification token.

Another approach is to convert the sequence of token vectors in the intermediate layers of the ViT backbone to a 2D grid and further process them using convolutional filters, leading to 3 different architectures we call *CNN-Ignore-EE*, *CNN-Add-EE* and *CNN-Project-EE*, each handling the classification token in a different way. Note that even though the intermediate layer is in the form of a sequence, each vector in the sequence corresponds to a patch of the input image, therefore putting the vectors back in a 2D grid simulates their original neighborhood which is essential when using convolutional filters that have a local receptive field. The motivation behind this approach is that convolutional filters are the current approach in the literature for early exiting [10, 21, 22] and can act as a baseline for the other proposed architectures. Furthermore, convolutional filters introduce low overhead in terms of parameters and computation. Additionally, a fusion of CNNs that can capture local structure very well but can not handle long range interactions, with ViTs which can process long range interactions, seems natural and may combine the advantages of both [6].

On the other hand, the local receptive field of CNN-based early exits may prove to be a drawback. An alternative that can overcome this limitation is using the Transformer encoder layer instead of the convolutional filters, which we call *ViT-EE*. Indeed, it has been shown that Transformer encoder layers can create superior early exits for CNN backbones by introducing a global receptive field [2]. However, since the layers of ViT backbones already have a global receptive field, it is not clear whether ViT-EE will have the same advantage over CNN-based early exits in ViT backbones as well. Another advantage of using Transformer encoder is the simplicity of its structure, which means it can handle various other data types such as point-clouds and even cross-modal data [2, 6]. The main drawback of ViT-EE is its high overhead, however, the building blocks of the recently proposed attention-free MLP-based architectures can serve as more lightweight alternatives that still maintain a global receptive field and structure simplicity, leading to *ResMLP-EE* and *MLP-Mixer-EE*.

Formally, the output of Transformer encoder b, denoted by P^b, consists of patch embeddings $p_1^b, \dots, p_N^b$ corresponding to the input image patches, as well vector p_0^b corresponding to the classification token. Since the shape of the intermediate representations is the same for all of the hidden layers, without loss of generality, we assume that the early exit branch is to be placed after Transformer encoder b. In MLP-EE, shown in Figure 3 (a), P^b is normalized to obtain $\bar{P}^b = Norm(P^b)$. Subsequently, an MLP consisting of three dense layers with two dropout layers in between takes $\bar{p}_0^b$ as input, where $\bar{P}^b = (\bar{p}_0^b, \dots, \bar{p}_N^b)$, and outputs the early result. The MLP layers in all our proposed architectures have the same three layers. In ViT-EE, shown in Figure 3 (b), P^b is given as input to a Transformer encoder layer [2]. The output of the Transformer encoder is then normalized and passed on to an MLP, similar to the previous architecture.

In CNN-based early exits, the N patch embeddings $p_1^b, \dots, p_N^b$ can be reshaped into a tensor $C^b \in \mathbb{R}^{\sqrt{N} \times \sqrt{N} \times d_v}$, akin to an intermediate representation in a CNN backbone, with height and width of $\sqrt{N}$ and d_v channels, and then passed on to a convolution layer, a max pooling layer and an MLP to obtain the early result. However, it is not clear what should be done with classification token $\bar{p}_0^b$. A similar situation arises in dense prediction using Vision Transformers, where three ways for dealing with the classification token are proposed [19]. In CNN-Add-EE, the classification token is added to every patch embedding, leading to $\bar{C}^b = (p_1^b + p_0^b, p_2^b + p_0^b, \dots, p_N^b + p_0^b)$; in CNN-Project-EE, the classification token is concatenated to ev-

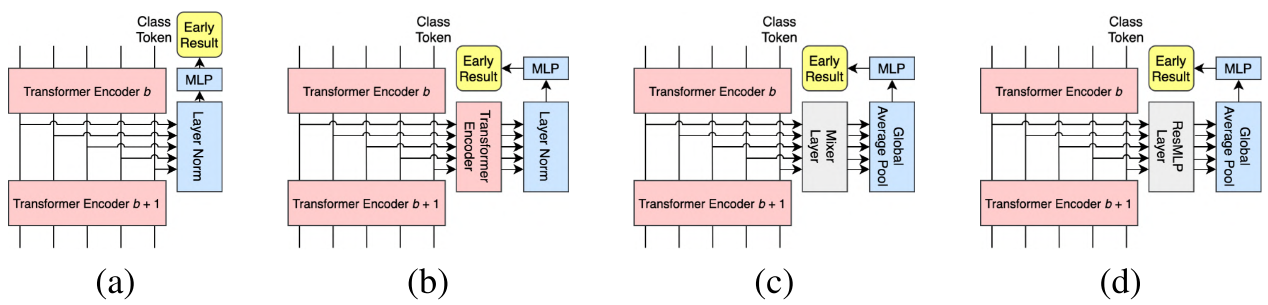


Figure 3: (a) MLP-EE; (b) ViT-EE; (c) MLP-Mixer-EE; and (d) ResMLP-EE early exit branch architectures.

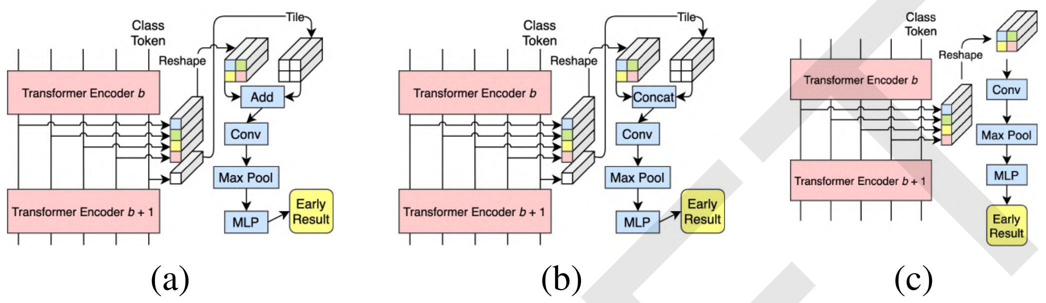


Figure 4: (a) CNN-Add-EE; (b) CNN-Project-EE and (c) CNN-Ignore-EE early exit branch architectures.

ery patch embedding, leading to $\bar{C}^b = (\mathit{concat}(p_1^b, p_0^b), \mathit{concat}(p_2^b, p_0^b), \dots, \mathit{concat}(p_N^b, p_0^b))$; and in CNN-Ignore-EE, the classification token is ignored and discarded, leading to $\bar{C}^b = C^b$. These three alternative architectures are depicted in figure 4.

As previously mentioned, the building blocks of attention-free MLP-based architectures can be low-overhead alternatives for ViT-EE which uses a Transformer encoder layer. Figure 3 (c) and (d) shows the MLP-Mixer-EE and ResMLP-EE early exit branch architectures, respectively. Note that similar to the original MLP-Mixer and ResMLP architectures, the output of the mixer layer and the ResMLP layer are passed on to a global average pooling layer.

4 Results

For the image classification experiments, we use CIFAR-10, CIFAR-100 and Fashion MNIST datasets [14, 26]. We use ViT-B/16 architectures with the original pre-trained weights provided by the authors [4] as backbones, and we train them on our target datasets using a cross-entropy loss function. For the regression experiments, we investigate crowd counting, which is the problem of counting the total number of people present in a given image [5]. We use DISCO [9] as the dataset and TransCrowd [17] which is a ViT-based architecture as the backbone. Mean absolute error (MAE) is commonly used to evaluate the accuracy of crowd counting models [3]. All three backbones have 12 Transformer encoder layers.

All our models were trained using the Adam optimizer [13] with learning rates of $\{10^{-3}, 10^{-4}, 10^{-5}\}$ and the best result is selected. The learning rate is reduced by a factor of 0.6 on plateau with a tolerance of 2 epochs, and an early stopping mechanism with a tolerance of 5 epochs is used.

Note that while early exits have been recently attached to high-performing CNN backbones [1, 2, 7], there is no prior work for early exits on Vision Transformer backbones. Since the performance obtained by Vision Transformer backbones is improved by a large margin, we omit listing the comparison with early exits on CNN backbones in the following results.

4.1 Classifier-Wise

With the classifier-wise training strategy, since the branches do not affect each other or the backbone, we place and train early exit branches with each of our proposed architectures at every single layer in the backbone. For each early exit branch, we record the accuracy of its early results as well as the total FLOPs up to and including the branch. The results are depicted in Figure 5, where all practical early exits are circled. In addition, the accuracy of the final exit of the backbone is shown in these figures.

These results can be used to select a collection of lightweight high-performing branches. With dynamic inference, it is desirable for the model to be as fine-grained as possible, therefore, with the classifier-wise strategy where placing more branches does not affect other branches or the backbone, it is desirable to place as many branches as possible on the backbone. To make this more clear, imagine a scenario where only two exit options are available: option A with 80% accuracy and 3B FLOPS, and option B with 90% accuracy and 6B FLOPS. If the computation budget at hand is 5B FLOPS, the only possible option to choose is A, resulting in 80% accuracy. However, with a finer-grained model that also includes option C with 85% accuracy and 4B FLOPS, choosing option C leads to 85% accuracy. Hence we examine all possible branch locations: if there exists a single practical branch at a location, that branch should be added at that location; if there are no practical branches at a location, then no branches should be added there, since more accurate and more lightweight exits are available; and if there are more than one practical branches at a location (for instance, with the DISCO dataset in Figure 5 (a), both CNN-Ignore-EE and CNN-Project-EE make practical branches at layer 2) it means that there is a trade-off between accuracy and computation at that location, and the proper branch should be selected based on the particular application. Alternatively, it is possible to deploy multiple branches at the same location simultaneously, and exit the one that fits the budget during inference. Note that with the classifier-wise strategy, there can be different branch types on the same backbone, for instance, there can be a CNN-Add-EE branch at location 1 and a ViT-EE branch at location 2.

Several observations can be made from these results. First, all of our proposed architectures create at least one practical branch. As expected, MLP-EE does not contain enough parameters and layers to extract useful features in early locations on its own, and thus performs poorly, while it catches up in the later locations where the features extracted by the intermediate layers can compensate. Furthermore, MLP-EE only processes the classification head, which contains only low-level features in very early layers. However, MLP-EE always creates the first practical branch as it is the most lightweight. Secondly, CNN-based branches outperform other types in the first few locations. This is likely because the fusion of convolutional layers that capture local interactions well, with the global attention of the backbone, combines the best of both worlds. However, this effect seems to fade in later locations, perhaps since several layers of the backbone are able to capture both local and global interactions fairly well. In addition, CNN-Ignore-EE outperforms other CNN-based early exits in most of these early cases, as the classification token in the very early layers contains only low-level features. Thirdly, aside from the very early locations where CNN-based branches dominate, ResMLP-EE performs better in classification problems, while ViT-EE performs better in crowd counting. Evident from the use of visual attention mechanisms and dilated convolutions in many high-performing models for crowd counting [5], global information such as perspective plays an important role in crowd counting, therefore, ViT-EE which can capture multiple types of attention through the use of the multi-head attention layer in Transformer encoder, outperforms ResMLP-EE which does not include a mechanism for handling multiple types of attention. Fourthly, observe that MLP-Mixer-EE outperforms ResMLP-EE in most locations in the crowd counting cases. This is because the affine transformation in ResMLP can be used instead of normalization when the training is stable [24], however, with crowd counting, the training process is not as stable as image classification.

Note that in the last six locations in CIFAR-10 and Fashion MNIST cases, the differences between the performance of different branch types are minuscule, and therefore less informative. Moreover, observe that unlike multi-exit architectures with CNN backbones, branches placed later on a ViT backbone do not necessarily provide a higher accuracy compared to previous branches. This is because in CNN backbones, the network has a very local receptive field in the early layers, and the receptive field gradually increases throughout the network, whereas ViTs have a global receptive field from the very first layer. This means that the accuracy of later branches of CNN backbones is expected to increase since the receptive field has increased, whereas in ViT backbones, later intermediate layers do not have any advantages in terms of the receptive field, thus their branches may obtain a lower accuracy. Finally, note that in the DISCO experiments, some of the very late early exit branches achieve a lower MAE compared to the final exit. This is because the MLP in our proposed architectures consists of three layers while the MLP in the ViT-B/16 backbone has one.

4.2 End-to-End and Layer-Wise

Unlike the classifier-wise training strategy, it is not possible to conduct a comprehensive study of the end-to-end and layer-wise strategies, since there are $2^{L-1} - 1$ possible branch placements. In addition, the end-to-end strategy can have infinitely many weight values for each of the placements. Therefore, for these training strategies, we only investigate two cases; one where a single early exit branch is placed after the sixth layer; the other where three branches are placed after the third, sixth and ninth layers. In both cases, the contribution of the final exit to the loss is double the contribution of the early exits.

Results are summarized in Tables 1 and 2. In all image classification cases, final accuracy is decreased compared to the backbones without early exits, which have an accuracy of 98.31% for CIFAR-10, 91.24% for CIFAR-100 and 95.00% for Fashion MNIST. However, in crowd counting, the final MAE is improved from the original 11.07 when a single MLP-EE, ViT-EE or MLP-Mixer-EE branch is used. Similar to the classifier-wise strategy, in both cases involving the DISCO dataset, MLP-Mixer-EE outperforms ResMLP-EE for the same reason explained above. Furthermore, ViT-EE outperforms other branch types in most cases, particularly when there are 3 exit branches, and performs very high in others. Since the end-to-end training strategy affects the parameters of the backbone, perhaps ViT-EE branches have the least negative impact on the backbone due to the similarity of their architecture with the layers of the backbone. This is further supported by the fact that CNN-based branches whose architectures differ the most from that of the backbone, typically perform much worse.

With the layer-wise strategy, we encounter a problem. Since Vision Transformers are data-hungry [12], they need to be pre-trained on very large datasets. For the first step of the layer-wise strategy where all layers up to and including the first early exit branch are trained, pre-trained weights exist, therefore, the training procedure achieves results with a high accuracy on par with their end-to-end counterpart. For instance, in the case of CIFAR-

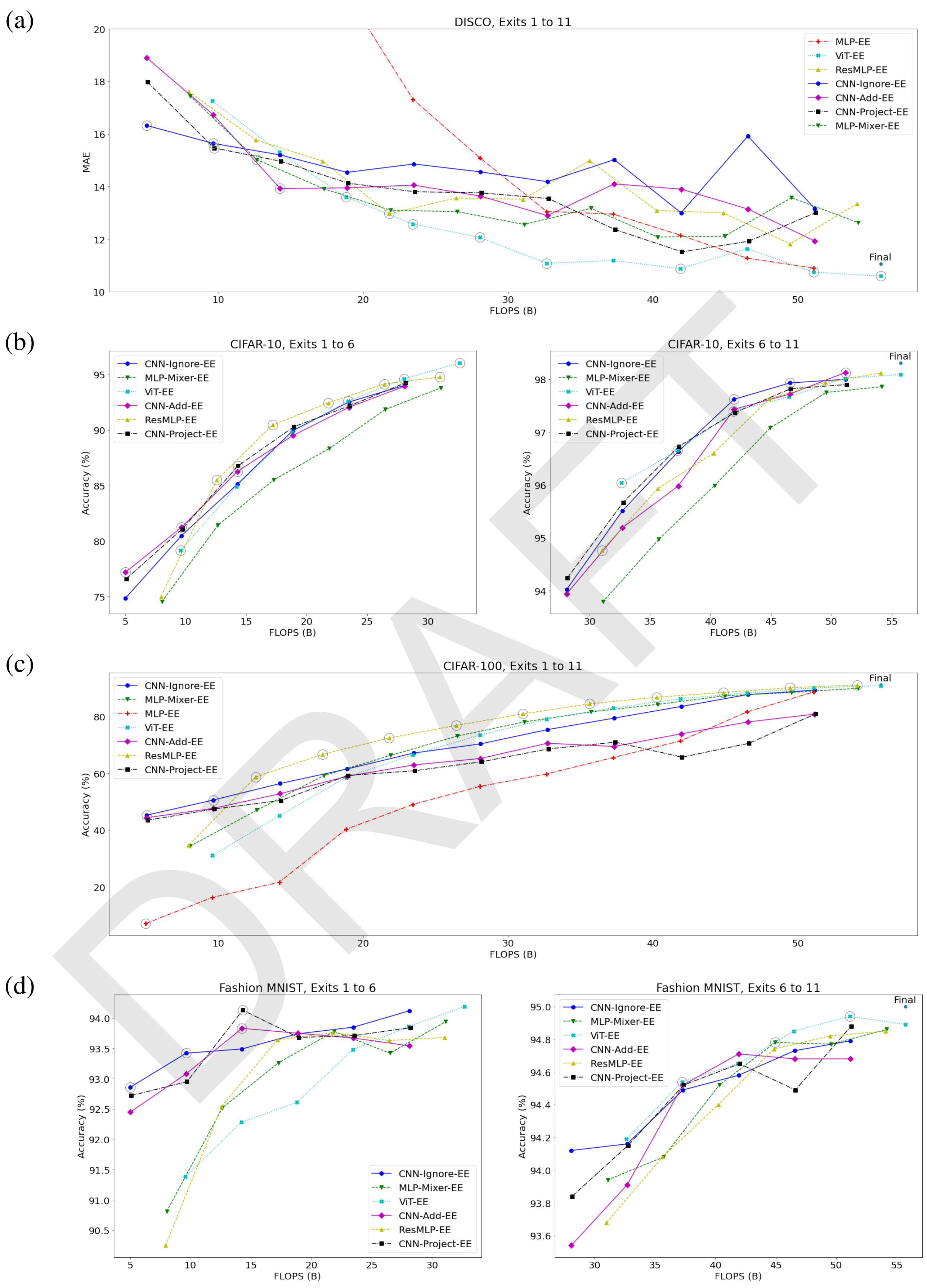


Figure 5: Performance of different multi-exit architectures on (a) DISCO; (b) CIFAR-10; (c) CIFAR-100 and (d) Fashion MNIST datasets trained with classifier-wise strategy. Practical early exit branches are circled. In order to highlight the differences, early exits with MLP-EE architecture are removed in (b) and (d) and branches 1 to 6 and 6 to 11 are separated.

Branch Arch.	CIFAR-10 Acc.		CIFAR-100 Acc.		Fasion MNIST Acc.		DISCO MAE		FLOPS
	Early@6	Final	Early@6	Final	Early@6	Final			
MLP-EE	94.90%	96.73%	81.12%	87.45%	**94.47%**	94.85%	**11.04**	**10.72**	28.04
CNN-Ignore-EE	**95.95%**	97.10%	77.97%	85.90%	94.43%	94.67%	21.88	11.09	28.10
CNN-Add-EE	94.94%	96.87%	75.75%	86.96%	94.22%	94.77%	18.46	11.24	28.10
CNN-Project-EE	94.66%	96.80%	77.95%	86.89%	94.33%	94.69%	18.23	11.29	28.16
ViT-EE	95.89%	96.99%	**85.23%**	**89.44%**	94.39%	94.84%	11.06	11.01	32.65
MLP-Mixer-EE	95.78%	97.07%	81.72%	87.53%	94.41%	94.88%	13.03	10.93	31.11
ResMLP-EE	95.44%	**97.35%**	82.41%	87.57%	94.38%	**94.95%**	16.99	11.36	31.02

Table 1: Performance of multi-exit architectures with one branch, trained with end-to-end strategy. The last column shows the FLOPS up to and including the branch.

Branch Arch.	CIFAR-10 Acc.				CIFAR-100 Acc.				DISCO MAE			
	Early@3	Early@6	Early@9	Final	Early@3	Early@6	Early@9	Final	Early@3	Early@6	Early@9	Final
MLP-EE	87.21%	94.48%	95.64%	96.19%	61.00%	79.83%	84.42%	86.46%	13.77	11.54	**11.55**	11.44
CNN-Ignore-EE	91.44%	95.68%	96.55%	96.56%	65.08%	79.35%	84.74%	86.32%	20.99	23.65	20.88	11.42
CNN-Add-EE	90.27%	95.63%	96.80%	96.94%	62.66%	78.86%	85.11%	87.01%	18.33	19.25	18.92	11.77
CNN-Project-EE	91.19%	95.77%	96.81%	96.99%	64.26%	78.63%	84.47%	86.19%	21.25	21.46	17.99	11.49
ViT-EE	92.35%	**96.01%**	**97.25%**	**97.33%**	**74.73%**	**84.31%**	**87.43%**	**87.88%**	12.76	**11.27**	11.59	11.18
MLP-Mixer-EE	91.16%	95.99%	96.96%	96.99%	66.24%	81.84%	86.68%	87.29%	**12.24**	13.95	15.29	11.46
ResMLP-EE	**92.53%**	95.87%	96.76%	96.86%	70.45%	82.61%	87.36%	87.85%	14.15	14.71	17.05	**11.09**

Table 2: Performance of multi-exit architectures with 3 branches, trained with end-to-end strategy.

10, CNN-Ignore-EE achieves 95.97% accuracy at the sixth layer. However, for subsequent steps, the original pre-trained weights can not be used since the weights of earlier layers have changed. We tested the training process with no pre-trained weights, with the original pre-trained weights as well as pre-trained weights from the end-to-end strategy, however, neither achieved a high accuracy.

5 Conclusion and Future Direction

We proposed seven architectures for early exiting Vision Transformer backbones, provided the motivations behind each of these architectures, and showed that depending on the branch locations, training strategy and the problem at hand, any of our proposed architectures can prove useful. Furthermore, we provided recommendations on selecting lightweight high-performing branches based on the results of our experiments. We discussed the role of architectural elements such as local and global interactions, receptive field, classification token, support for multiple types of attention, normalization and similarity of branch architecture with the backbone layers, on the performance of multi-exit ViT architectures.

A potential future research direction would be to check whether other recent architectures for computer vision that operate on a sequence of image patches such as Perceiver [11], gMLP [18] and FNet [16] produce suitable early exits for Vision Transformer backbones.

Acknowledgment

This work was funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, and by the Danish Council for Independent Research under Grant No. 9131-00119B. This publication reflects the authors views only. The European Commission and the Danish Council for Independent Research are not responsible for any use that may be made of the information it contains.

References

[1] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Improving the accuracy of early exits in multi-exit architectures via curriculum learning. *arXiv:2104.10461*, 2021.

[2] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Single-layer vision transformers for more accurate early exits with less overhead. *arXiv:2105.09121*, 2021.

[3] Feng Dai, Hao Liu, Yike Ma, Juan Cao, Qiang Zhao, and Yongdong Zhang. Dense scale network for crowd counting. *arXiv:1906.09707*, 2019.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[5] Guangshuai Gao, Junyu Gao, Qingjie Liu, Qi Wang, and Yunhong Wang. Cnn-based density estimation and crowd counting: A survey. *arXiv:2003.12783*, 2020.

[6] Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, Dun Liang, Ralph R. Martin, and Shi-Min Hu. Can attention enable mlps to catch up with cnns? *arXiv:2105.15078*, 2021.

[7] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *ICML*, 2019.

[8] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *arXiv:2102.04906*, 2021.

[9] Di Hu, Lichao Mou, Qingzhong Wang, Junyu Gao, Yuansheng Hua, Dejing Dou, and Xiao Xiang Zhu. Ambient sound helps: Audiovisual crowd counting in extreme conditions. *arXiv:2005.07097*, 2020.

[10] Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. In *ICLR*, 2020.

[11] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. *arXiv:2103.03206*, 2021.

[12] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv:2101.01169*, 2021.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, May 2015.

[16] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv:2105.03824*, 2021.

[17] Dingkang Liang, Xiwu Chen, Wei Xu, Yu Zhou, and Xiang Bai. Transcrowd: Weakly-supervised crowd counting with transformer. *arXiv:2104.09116*, 2021.

[18] Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. Pay attention to mlps. *arXiv:2105.08050*, 2021.

[19] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *arXiv:2103.13413*, 2021.

[20] Amin Sabet, Jonathon Hare, Bashir Al-Hashimi, and Geoff V. Merrett. Temporal early exits for efficient video object detection. *arXiv:2106.11208*, 2021.

[21] Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. Why should we add early exits to neural networks? *Cognitive Computation*, 12(5):954–966, Sep 2020.

[22] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, 2016.

[23] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. *arXiv:2105.01601*, 2021.

[24] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv:2105.03404*, 2021.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[26] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.

7.9 Automatic Social Distance Estimation for Photographic Studies: Performance Evaluation, Test Benchmark, and Algorithm

The appended preprint follows.

Automatic Social Distance Estimation For Photographic Studies: Performance Evaluation, Test Benchmark, and Algorithm[⋆]

Mert Seker[a,*], Anssi Männistö[b], Alexandros Iosifidis[c,**], Jenni Raitoharju[d]

[a] Unit of Computing Sciences, Tampere University, Tampere, Finland
[b] Unit of Communication Sciences, Tampere University, Tampere, Finland
[c] Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark
[d] Programme for Environmental Information, Finnish Environment Institute, Jyväskylä, Finland

Abstract

The COVID-19 pandemic has been ongoing since March 2020. While social distancing regulations can slow the spread of the virus, they also directly affect a basic form of non-verbal communication, and there may be longer term impacts on human behavior and culture that remain to be analyzed in proxemics studies. To obtain quantitative results for such studies, large numbers of personal and/or media photos must be analyzed. Several social distance monitoring methods have been proposed for safety purposes, but they are not directly applicable to general photo collections with large variations in the imaging setup. In such studies, the interest shifts from safety to analyzing subtle differences in social distances. Currently, there is no suitable benchmark for developing such algorithms. Collecting images with measured ground-truth pair-wise distances using different camera settings is cumbersome. Moreover, performance evaluation for these algorithms is not straightforward, and there is no widely accepted evaluation protocol. In this paper, we provide an image dataset with measured pair-wise social distances under different camera positions and settings. We suggest a performance evaluation protocol and provide a benchmark to easily evaluate such algorithms. We also propose an automatic social distance estimation method that can be applied on general photo collections. Our method builds on object detection and human pose estimation. It can be applied on uncalibrated single images with known focal length and sensor size. The results on our benchmark are encouraging with 91% human detection rate and only 38.24% average relative distance estimation error among the detected people.

Keywords: Social Distance Estimation, Person Detection, Human Pose Estimation, Performance Evaluation, Test Benchmark, Proxemics

1. Introduction

Social distances are a part of non-verbal human communications and, naturally, there are personal and cultural differences in how people feel about their personal space and interpret the interpersonal distance in different situations. The research field under social studies concerning these phenomena related to space is known as *proxemics* (Hall et al., 1968). Despite the long history of studies in the field (Hall, 1966; Cook, 1970; Harrigan, 2005), it remains difficult to carry out quantitative analysis on the actual social distances in the natural situations outside of monitored test conditions, e.g., when people are spending their free time with their families. One way to approach this problem is *visual social distancing* (VSD), where the interpersonal distances are automatically measured from the images or videos. A comprehensive overview of the VSD problem, including the main challenges and connections to social studies, is provided in (Cristani et al., 2020).

Social distancing has recently received a lot of attention due to the outbreak of SARS-CoV-2 virus (Gorbalenya et al., 2020) that was declared as a global pandemic by the World Health Organization (WHO) in

[⋆] M. Seker, A. Männistö, and J. Raitoharju would like to acknowledge the financial support from Helsingin Sanomat foundation, project "Machine learning based analysis of the photographs of the corona crisis". A. Iosifidis acknowledges funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337 (MARVEL).

[*] Corresponding author.
[**] EURASIP member.
Email addresses: mert.seker@tuni.fi (Mert Seker), anssi.mannisto@tuni.fi (Anssi Männistö), ai@ece.au.dk (Alexandros Iosifidis), jenni.raitoharju@syke.fi (Jenni Raitoharju)

March 2020. The pandemic, also known as the COVID-19 pandemic is still ongoing as of November 2021 and there has been a total of about 253 million confirmed cases and 5.1 million deaths worldwide within the period of December 2019-November 2021 (Organization, b). Social distancing plays an important role in slowing down the spread of the virus. WHO recommends to stay at least one meter apart from other people in order to reduce the risk of infection (Organization, a). Automatically monitoring the social distances is important for safety reasons, but it is also interesting as a phenomenon that has globally changed basic human behavior (Zhang et al., 2021; Di Corrado et al., 2020; Eden et al., 2020). After the pandemic eases, there are many interesting research questions in proxemics and other fields to look into: how the social distancing affected every-day life, what kind of significant differences were there between different countries, can the differences be linked to the spreading speed, will there be any long-term changes that will stay after the pandemic.

While there are methods and sensors available for automatic monitoring of social distances (Nguyen et al., 2020), the analysis of deeper and longer term social and cultural impacts of the social distancing regulations requires looking into different source data, such as personal photo collections and pictures published in newspapers and magazines. For monitoring purposes, it is possible to use fixed camera setup and location, take videos or simultaneous images from multiple viewpoints, and use additional sensors such as depth or thermal cameras. All these can make the social distance estimates more accurate but are not available for typical personal and media photos that are not taken with a fixed setup, but have varying parameters such as focal length, sensor size, lighting conditions, and pitch angle. An example of an image that could be found in a personal or media photo collection, but not in a monitoring or surveillance setup is shown in Fig. 1. At the same time, in social and proxemics studies the focus shifts from monitoring whether people are obeying the regulations to more subtle differences in the social distances and how they are represented in the media.

During the pandemic, most effort has been understandably on the monitoring side, and currently there is no suitable benchmark for developing and testing algorithms for accurate social distance analysis from single images having varying camera parameters. This can be due to the laboriousness of gathering varying images with measured pair-wise distances between humans. At the same time, there is no clear protocol for measuring the algorithm performance in this task. To address these lacks, we provide a social distance evaluation test benchmark including a protocol for mapping the detected pair-wise distances into the corresponding ground truth distances, a suggested overall performance metric, and 300 test images taken with varying setups: indoors-outdoors, sitting-standing, varying camera angles using 2 different cameras and 7 different focal lengths. The photos were taken by a professional photojournalist to follow the typical media photography style. We publish also easy-to-use codes for evaluating novel methods and make it easy to integrate additional test photos.

Figure 1. An example of an image that represents a style, which is common in personal and media photography, but not in monitoring.

We also propose a social distance estimation algorithm that can be applied on any uncalibrated single image taken by a regular camera as long as focal length and sensor size are known. It combines object detection and human pose estimation with projective geometry using image parameters (focal length, sensor size) and pixel locations. While the results are promising, we also point out some of the main remaining challenges for future development.

The rest of the paper is organized as follows. Section 2 introduces related work on social distancing and automatic distance evaluation. Section 3 describes the provided test benchmark and the proposed evaluation protocol. Our method for automatic social distance estimation is described in Section 4. Section 5 provides our experimental setup and results and, finally, Section 6 concludes the paper.

2. Related Work

Effectiveness of social distancing on slowing down the spread of the COVID-19 virus has been widely studied (Vokó and Pitter, 2020; Sun and Zhai, 2020; Prem et al., 2020; Courtemanche et al., 2020; Abouk and Heydari, 2021; Balasa, 2020), and these studies confirm that social distancing measures are successful in reducing the growth rate of the virus. Therefore, monitoring and regulating the social distancing behaviour between people has played a crucial part in dampening the effects of the virus. In addition to directly effecting the virus spread, social distancing has globally changed human behavior and interactions leading to different side-impacts, e.g., on mental health (Ford, 2020; Jacob et al., 2020), physical activity (Di Corrado et al., 2020; Jacob et al., 2020), mood and memory (Zhang et al., 2021), and media consumption (Eden et al., 2020). Such impacts and their cross-cultural (Al-Hasan et al., 2020a,b; Doogan et al., 2020) and cross-sectional (Jacob et al., 2020; Lee et al., 2021) differences continue to draw attention from researchers in many fields.

Social distance monitoring for safety reasons can be eased by automatic social distance estimation from images and videos. A comprehensive survey in (Nguyen et al., 2020) explores the wide array of current technologies that can be used to monitor and encourage social distancing. A commercial pedestrian tracking system was used in (Pouw et al., 2020) to detect passengers in crowded environments and estimate the distances between them by using a graph based approach. A study in (Ahmed et al., 2021) proposed using a deep learning based model with YOLOv3 (Redmon and Farhadi, 2018) as its backbone to monitor social distancing violations from overhead view cameras. In (Punn et al., 2020), the authors used YOLOv3 and DeepSort (Wojke et al., 2017; Wojke and Bewley, 2018) to detect bounding boxes of people in RGB images and by utilizing these bounding boxes, they detected the cases of social distance violations.

A work in (Aghaei et al., 2021) proposed to use skeleton keypoints generated from human body pose estimation algorithms (Cao et al., 2019; Simon et al., 2017; Cao et al., 2017; Wei et al., 2016) to estimate the distance between people from uncalibrated images. The authors used manual tuning to estimate the homography matrix (Young, 1982) of an image plane and then used leg, arm, and torso lengths of the people alongside with the homography matrix to draw a safe space circle underneath every detected person. Then, any collision between the estimated safe space circles was reported as a social distance violation. Similarly, the work in (Fabbri et al., 2020) takes advantage of manual homography matrix calibration to estimate social distances for fixed cameras. Separating the work from (Aghaei et al., 2021), bounding boxes obtained from the object detection model (Zhou et al., 2019) and the height of these boxes are used as reference points to estimate the locations of the people. Moreover, a small CNN is used to estimate the feet locations even when they are not visible. The output of this CNN is used to correct the height of the bounding boxes in cases of occlusions. Another similar study in (Yang et al., 2020) also used bounding boxes obtained from object detectors (Alexey Bochkovskiy, 2020; Ren et al., 2016) to estimate locations of the people from surveillance camera footage by using the homography matrix that is calculated from the known extrinsics.

The work in (Bertoni et al., 2021) used a feed forward neural network that was trained on the intrinsic parameters of the camera and the keypoints obtained from a pose estimation model. The model outputs the predicted 3D locations as well as the orientations of the detected people. While detecting safe distance violations, not only the proximity but also the orientation of the people with respect to one another is considered. Finally, the study in (Morerio et al., 2021) proposed a neural network architecture that takes a pair of 2D body keypoints as input and outputs the estimated pair-wise distance. The two sets of body keypoints are converted into feature vectors by an encoder block. The vectors are then concatenated and given as input to a regressor block, followed by a fully connected layer that was trained on the public datasets Epfl-Mpv-VSD (Fleuret et al., 2008), Epfl-Wildtrack-VSD (Chavdarova et al., 2018), OxTown-VSD (Benfold and Reid, 2011) and Kitti (Geiger et al., 2012) to estimate pair-wise distances. The output of the regressor block is also used as input to another branch with a gradient reversal layer (Ganin et al., 2016) to estimate the camera's tilt angle and height from the ground plane in order to make the estimations more robust to variations in camera viewpoints. The method works on any single uncalibrated image.

Most of the introduced works approach automatic social distance estimation as a monitoring or surveillance task, where the goal is to prevent social distance regulation violations. To this end, they apply additional sensors, use predefined camera settings, and/or manually define a homography matrix for a certain environment. While such approaches can improve the social distance estimation accuracy, they are not feasible when the purpose is to analyze the impacts of social distances from personal or media photo collections.

Moreover, the above-mentioned studies approach the automatic social distance estimation problem as a binary classification problem where they aim to classify the pair-wise distances between people either as safe or unsafe, depending on a given threshold. Classifying distances in a binary manner has a high tolerance for distance estimation errors. For example, if the threshold for safe distance is set to 2 meters, the actual distance between a pair of people is 1.9 meters, and a method estimates that distance as 0.1 meter, the percentual distance estimation error would be 94.7%, but a binary classification approach would still correctly label the situation as a social distance violation. Furthermore, the binary approach does not provide any additional information on the severity of the violations in different situations which may be relevant information for subsequent analysis.

A common pattern observed in most of the machine learning based social distance estimation methods (with the exception of at least (Aghaei et al., 2021; Morerio et al., 2021; Bertoni et al., 2021) that use keypoints of the human body) is that they rely on the bounding boxes drawn by object detectors to detect social distance violations. Although the current object detectors are accurate in detecting objects, the bounding boxes are generally loosely drawn around these objects. Thus, it is not reliable to use only the bounding box information for estimating exact distances between people as it is not possible to infer accurate 3D location estimates from the bounding boxes alone. Therefore, we aim to estimate exact 3D locations of all the people in uncalibrated RGB images with respect to the camera by using the information extracted from the human body skeleton detected by body estimation algorithms. Moreover, we also incorporate an object detection model for people detection. However, the purpose of the people detection in our approach is to only detect the false positives in skeleton keypoints, when they are drawn on non-human objects.

The method in (Aghaei et al., 2021) is the most similar to our method as it also uses body poses. In (Aghaei et al., 2021), manual input is used to estimate the homography matrix of the image plane to the ground plane. The method is evaluated on surveillance camera footage and the task is approached as a binary classification problem. It is feasible to manually set the homography matrix of surveillance cameras as these cameras are generally non-moving and stable. Contrary to this, we want our method to be fully automatic as we aim to estimate distances in images taken in different locations with different cameras. Instead of requiring manual input to estimate the homography as the study in (Aghaei et al., 2021), we assume that we can find keypoint pairs that are parallel to camera's sensor plane and we use the image parameters, i.e., focal length and sensor size in our distance estimation.

For the developing and testing social distance estimation methods, it is important to have image datasets that have a suitable setup and ground-truth for the task. The previous works have used datasets such as Epfl-Mpv-VSD, Epfl-Wildtrack-VSD and OxTown-VSD. These datasets include videos taken by surveillance cameras with fixed extrinsic and intrinsics and they do not include manually measured ground truth locations and distances. Instead, the locations of the people are estimated by making use of the annotation boxes that were drawn on the people. The pixel locations of these annotation boxes are used as a reference point to estimate the subjects' locations by taking the extrinsic parameters into account. This means that these locations are not exactly ground truth, but estimations based on the known extrinsics and the pixel locations of the manually annotated person bounding boxes. Furthermore, since exact body parts are not annotated and the annotations are only in bounding box format, it is not feasible nor possible to accurately match the detected people with the given ground truth people when there are multiple overlapping boxes. Moreover, only the people that are passing on a certain region of interest are annotated.

Due to the aforementioned reasons, the existing datasets are not suitable for evaluating methods that aim at estimating distances in general photo collections and are not manually tuned for a specific camera and environments. Furthermore, the approximate person annotations and location estimates do not allow accurately measuring the distance estimation performance, but are only suitable for detecting coarse violations in social distancing recommendations. While this may be sufficient for surveillance purposes in fixed environments, more accurate ground-truth and annotations are needed for evaluating methods aiming at detecting subtle changes in long-term social distancing behavior in varying environments. In the following section, we introduce our novel dataset that addresses the mentioned drawbacks of the existing datasets.

3. KORTE SOCIAL DISTANCE ESTIMATION BENCHMARK

We provide a test benchmark for facilitating research in automatic social distance evaluation. We propose a performance evaluation protocol and provide 300 test images with ground-truth pair-wise distances. While the number of images is too low for training fully learning-based systems, it provides a varied test setup. All the evaluation codes along with the test photos are publicly available at https://doi.org/10.23729/b2ea87e6-b845-46b8-abf3-cdbe299ce8b0. It is also easy to complement the benchmark with additional images by fol-

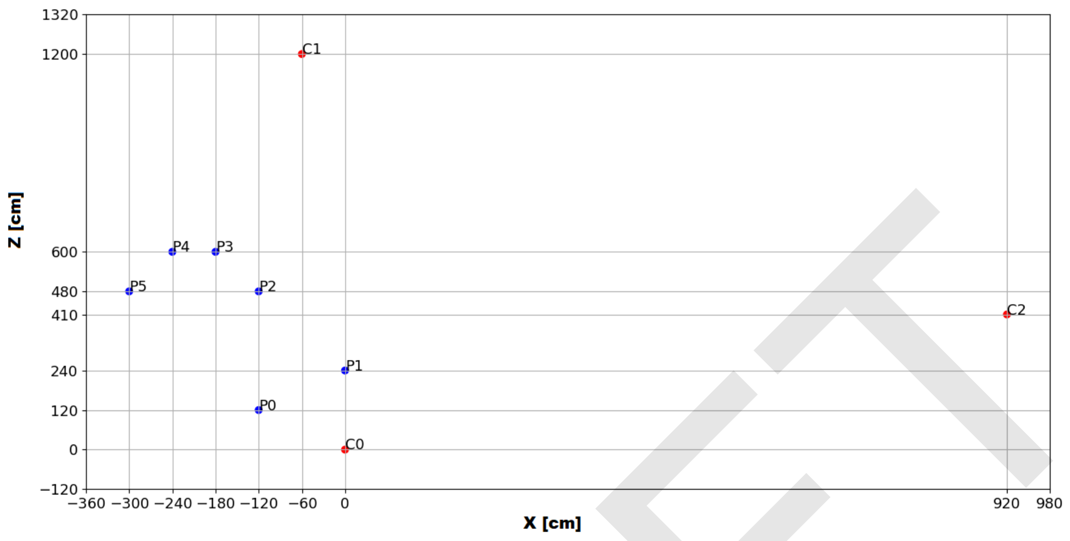


Figure 2. Birdseye view of the first photo shoot (outdoor). The ground truth locations of the people and cameras are given in blue and red dots, respectively.

lowing the proposed annotation format and using the provided evaluation protocol.

3.1. *Test Photo Collection*

We collected test photos in four separate photo shoots. The first and third photo shoots were organized outdoors at Tampere University campus in December 2020 and August 2021, respectively. Every person was standing. The second and fourth photo shoots were organized indoors at Tampere University campus in January 2021 and August 2021 with people sitting around tables and sofas. We had 6 volunteer test subjects in the first and second photo shoots and 7 volunteer test subjects in the third and fourth photo shoots. We followed the COVID-19 restrictions at the time: everyone was wearing a mask and we were less than 10 people gathering. As an additional safety measure, we placed to closest distances from each other only people who meet regularly anyway because they share working space or live together. Every test subject signed an agreement allowing to use their images for research purposes. Any bypassers in the images were censored out to respect their privacy and because their exact positions were unknown. The photos were taken by a professional photojournalist.

During the photo shoots, test subjects stayed on the same known positions, while the photographer changed his position and used multiple cameras and lenses at each spot. Fig. 2 shows as an example the birdseye view of the first photo shoot (outdoor). P0, P1, P2, P3, P4, P5 are the locations of the 6 test subjects and C0, C1, C2 are the camera locations. For the first photo shoot, P0, P1, P2, P3, P4, P5, C0 and C1 were all on the same ground plane, while C2 was at a balcony with a height of 230 cm relative to the ground plane that all of the other locations were at. Similar birdseye views of the other photo shoots are included in the Appendix (A.12-A.14). The unit of the x and z axis labels is centimeters. The ground truth locations of the cameras and the test subjects were measured and maintained exploiting tiles on the ground/floor that were equal in size. While test subjects' locations were fixed during each photo shoot, they were asked to vary their orientation and pose. The ground truth locations of all the cameras and test subjects for all the photo shoots are provided with the dataset.

We do not report the exact pitch angles, and they were not fixed in the photo shoots. Due to the camera positions, pitch angles are close to zero in most of the images except for the 54 photos taken from camera position C2 in the first and third photo shoot, where the camera was at an elevated position. We believe that our dataset represents a typical media or personal photo collection with respect to the pitch angles, but it should be noted that methods performing well on our dataset (especially if they rely on the zero pitch angle assumption) may not perform equally well on extreme pitch angles such as overhead images.

The used camera models were Canon EOS 5D Mark II and Canon EOS 6D Mark II. The used focal lengths were 16, 24, 35, 50, 105, 200, and 300 mm. The cameras were stabilized on a tripod. Fig. 3 shows example photos from the first and second photo shoots, one photo from each camera position.

Focal	Camera Model	Shooting Setting	
Length	Canon	Indoor	Outdoor
16	EOS 5D Mark II	-	6
16	EOS 6D Mark II	12	10
24	EOS 5D Mark II	5	8
24	EOS 6D Mark II	25	8
35	EOS 5D Mark II	-	-
35	EOS 6D Mark II	24	24
50	EOS 5D Mark II	-	31
50	EOS 6D Mark II	23	30
105	EOS 5D Mark II	15	-
105	EOS 6D Mark II	22	32
200	EOS 5D Mark II	-	7
200	EOS 6D Mark II	-	-
300	EOS 5D Mark II	-	8
300	EOS 6D Mark II	-	10
All		126	174

Table 1. Numbers of photos in the test dataset for different focal lengths (mm), camera models, and shooting settings (indoor/outdoor).

3.2. *Test Data Description*

The overall dataset contains 300 images including 174 outdoor images and 126 indoor images. All of the images are in JPG format. The resolutions of the images are 2400x1600, 4080x2720 and 4160x2768 with 139, 80 and 81 images in each resolution, respectively. Two different camera models were used and the sensor size for both of these cameras is 36 mm in width and 24 mm in height. The distribution of the pictures in terms of focal lengths, camera models, and shooting settings is given in Table 1.

Along with the images, we also provide different annotation data provided in three separate .csv files illustrated in Fig. 4. The first file (Fig. 4a) contains the pixel locations of four different body parts. These annotated body parts are the center of the eyes, the center of the shoulders, the center of the torso, and the center of the head. If a body part is not visible in the image, it is not annotated. The people in the images are labeled as P0, P1, P2, P3, P4, P5, P6, P7 and P8 in the annotation file. These person tags are consistent through all of the images. This means that a person tag always refers to the same person in all of the images that we provide. The second file (Fig. 4b) contains the 3D locations of people and different camera positions in all photo shoots. Photo shoot IDs 0, 1, 2, and 3 refer to the first (outdoor), second (indoor), third (outdoor), and fourth (indoor) photo shoots, respectively. The third file (Fig. 4b) links the image filenames with the corresponding photo shoot and camera location. The cameras' exterior orientation parameters are not included in the metadata of the images.

New images can be added to the dataset simply by following the described structure of the annotation data shown in Fig. 4. This does not require any changes in the provided evaluation codes. New photo shoots, i.e., new settings of people, must be identified with a unique integer identifier. For any photo shoot, the real world locations of the people should stay the same in all the photos. There may be pictures taken from different camera locations. The person and camera tags should start with a letter P and C, respectively, followed by a unique identifier integer. The person and camera location tags must be consistent within a given photo shoot, however repeated tags in different photo shoots are allowed. This means that two different people or camera tags could be the same as long as they belong to a different photo shoot. At least 1 of 4 body parts (center of the eyes, shoulders, torso, head) of the people in the images must be annotated in terms of pixel locations. They should be named "Eyes", "Shoulder", "Torso", and "Head" in the body part column of the body part pixel location file in Fig. 4a.

To be consistent with the annotations in the provided test images, the annotation can be done as follows. Using the keypoint numbering in Fig. 6, the center of the eyes refers to the middle point of the keypoint pair 15-16, the center of the shoulders refers to the middle point of the keypoint pair 2-5, the center of the torso refers to the middle point of the keypoint pair 1-8, and the head should be annotated as middle point of the head regardless of the head's angle with respect to the camera. If a head is sideways and only one of the eyes is visible, the visible eye can be annotated as the center of the eyes. If no eyes are visible, the center of the eyes should not be annotated. The center of the eyes should also not be annotated if at least one of the eyes is out of the picture due to the head being on the edge of the picture. The other body parts can be annotated as long as they are either completely visible in the picture or are partially occluded by another person or object. In the cases where they are partially occluded, the pixel loca-

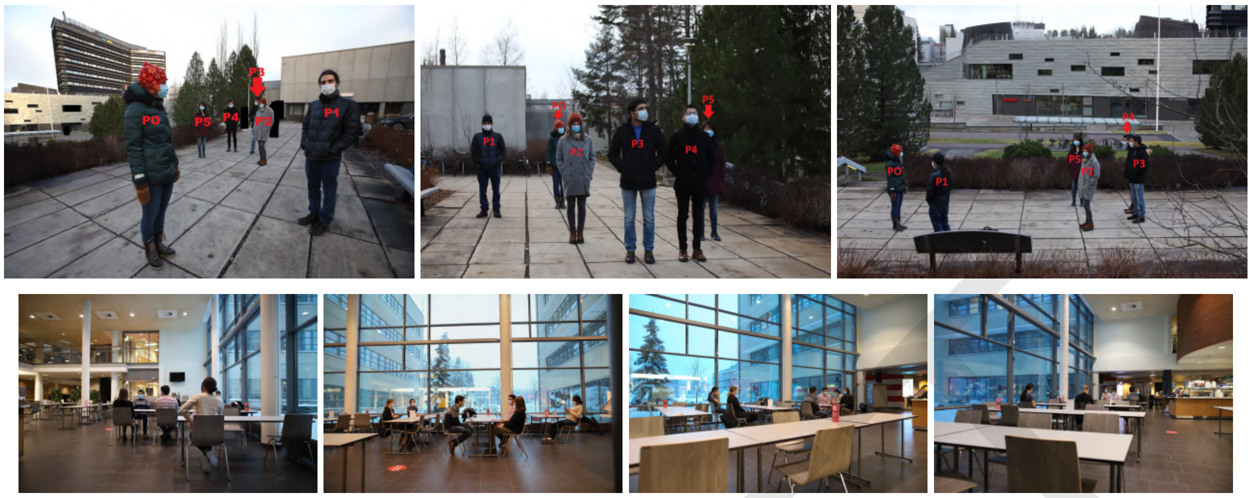


Figure 3. Example photos from the test dataset. The upper row has photos from the first photo shoot (outdoor) taken from all camera positions C0 (left) to C2 (right) and the lower row has photos from the second photo shoot (indoor) taken from all camera positions C0 (left) to C3 (right).

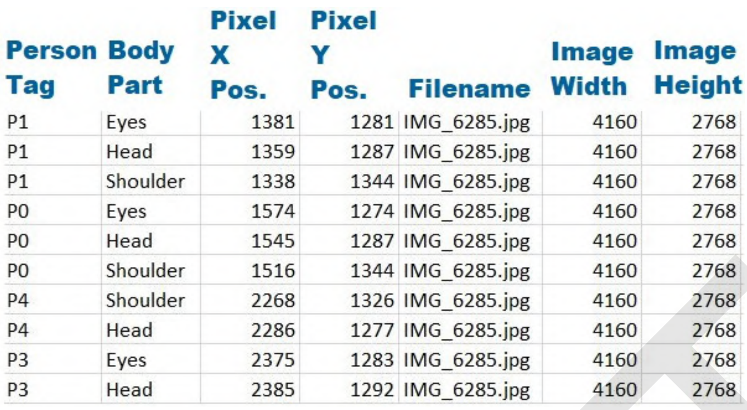
tion should be estimated as if the occluding person or object was not present in the picture. The center of the shoulders, torso, and head should not be annotated only in the cases where these body parts are either partially or completely out of the picture due to the person being on the edge of the picture. If a person is sideways and only one of the shoulders, i.e., keypoints 2 and 5, is visible, this point can be annotated as the center of the shoulders.

3.3. Evaluation Protocol

Any distance estimation method to be tested using the benchmark should give as output at least 1 of the 4 annotated pixel body locations along with either the estimated 3D location of the persons or the estimated distances between the people. The body part can be different for each person, or a method may choose to give only a single body part, such as the head, for all the persons. The test benchmark uses the pixel locations to automatically match each detected person with one of the ground truth locations and then computes average percentual pair-wise estimation errors between the estimated and ground truth distances.

We provide all the necessary functionalities for testing as long as the required output for each image is given. Internally, the matching is carried out by comparing the automatically detected body pixel locations with the points annotated in the files. The automatically detected body parts are compared to all of the respective annotated body parts. As an example, a detected torso point is compared to all of the annotated torso points for that image. For all of the detected body parts of a person, the closest respective annotated point in terms of pixel-wise distance is found. In case there are more than one detected persons matched with the same ground truth person, the matching is done in a greedy manner by selecting only the closest match and the rest of the detected persons for that ground truth person are regarded as false positives.

After matching the detections with the persons labeled in the photos, we calculate the distances between each person pair by using their estimated 3D locations. Then, the estimated pair-wise distances are compared to the corresponding ground truth pair-wise distances to obtain a percentual distance estimation error for each pair. The performance is evaluated by taking the average of all of the pair-wise percentual distance estimation errors for each image and then averaging over images. In addition to the pair-wise percentual distance estimation error, we evaluate also the person detection rate, i.e., the ratio of correctly detected person averaged over all the images, and the false discovery rate averaged over all the images. It should be noted here that we do not use any threshold for matching the detections with the actual people. As long as the number of detections is lower or equal to the actual number of people in an image, all the detections are matched. Thus, detections can be considered false positives only if there are more detections than actual people for an image. Therefore, a method producing many false positive detections is expected to get a high detection rate, but naturally the distance estimations would likely be poor and the false discovery rate would be higher. On the other hand, a method missing most the people could have a



Person Tag	Body Part	Pixel X Pos.	Pixel Y Pos.	Filename	Image Width	Image Height
P1	Eyes	1381	1281	IMG_6285.jpg	4160	2768
P1	Head	1359	1287	IMG_6285.jpg	4160	2768
P1	Shoulder	1338	1344	IMG_6285.jpg	4160	2768
P0	Eyes	1574	1274	IMG_6285.jpg	4160	2768
P0	Head	1545	1287	IMG_6285.jpg	4160	2768
P0	Shoulder	1516	1344	IMG_6285.jpg	4160	2768
P4	Shoulder	2268	1326	IMG_6285.jpg	4160	2768
P4	Head	2286	1277	IMG_6285.jpg	4160	2768
P3	Eyes	2375	1283	IMG_6285.jpg	4160	2768
P3	Head	2385	1292	IMG_6285.jpg	4160	2768

(a) Body part pixel locations

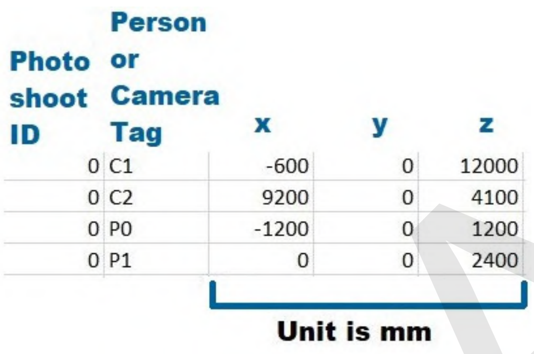


Photo shoot ID	Person or Camera Tag	x	y	z
0	C1	-600	0	12000
0	C2	9200	0	4100
0	P0	-1200	0	1200
0	P1	0	0	2400

Unit is mm

(b) Ground truth relative 3D location

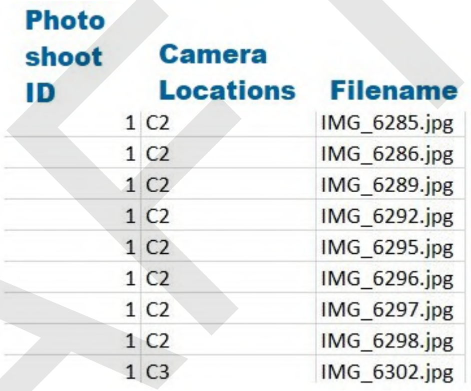


Photo shoot ID	Camera Locations	Filename
1	C2	IMG_6285.jpg
1	C2	IMG_6286.jpg
1	C2	IMG_6289.jpg
1	C2	IMG_6292.jpg
1	C2	IMG_6295.jpg
1	C2	IMG_6296.jpg
1	C2	IMG_6297.jpg
1	C2	IMG_6298.jpg
1	C3	IMG_6302.jpg

(c) Photo shoot identifiers and camera locations

Figure 4. Annotation file formats

low pair-wise percentual distance estimation error for the detected people, but still not be suitable for social distancing analysis. Therefore, it is important to consider all these metrics together, when evaluating a social distance estimation algorithm.

The pair-wise percentual distance estimation error D_e for the e^{th} single image is given by the following formula, where n is the number of detected people in the image, E_i is the estimated 3D location of the i^{th} person and G_i is the ground truth 3D location of the i^{th} person:

$$D_e = \frac{\sum_{k=1}^{n-1} \sum_{i=k+1}^{n} \frac{\left| \|E_k - E_i\| - \|G_k - G_i\| \right|}{\|G_k - G_i\|} * 100}{\binom{n}{2}}. \quad (1)$$

Here, the distances may be also directly given instead of the 3D locations.

In order to obtain an overall distance estimation error metric for a set of images, D_e of all of the images in the image set are averaged. The distance estimation error for a set of images D_E is given by the following formula where N is the number of images in the set:

$$D_E = \frac{\sum_{e=1}^{N} D_e}{N}. \quad (2)$$

The test benchmark gives D_E, the person detection rate, and the false discovery rate as an output for a given set of images as long as the input and annotated data are provided in the proper format. Currently, the test benchmark uses our provided test photos, but if new images are added to the dataset as explained in Section 3.2, these will be automatically considered in the evaluation.

4. Proposed Method for Social Distance Estimation

Our proposed method to estimate social distances takes advantage of object detection and human pose estimation methods. Firstly, the input image is given to YOLOv4 (Alexey Bochkovskiy, 2020) object detection model to obtain bounding boxes for people. After bounding boxes are obtained, overlapping boxes are grouped together. Then, these grouped boxes are

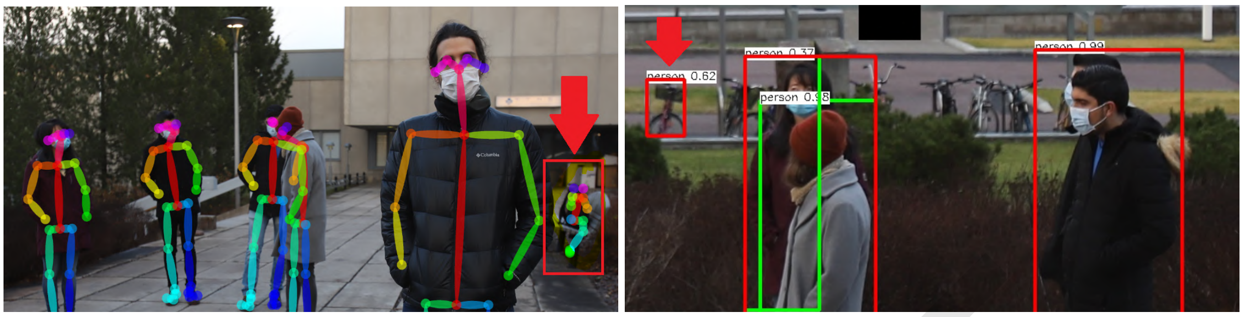


Figure 5. False positive examples for OpenPose (left) and YOLOv4 (right).

cropped from the full image and they are individually given to OpenPose (Cao et al., 2019; Simon et al., 2017; Cao et al., 2017; Wei et al., 2016) human pose estimation model. After the skeleton keypoints are extracted from OpenPose, the pixel locations of these keypoints are used in our distance estimation algorithm to obtain 3D location estimates for each person in the image.

When YOLOv4 and OpenPose models are used together, they eliminate each other's false positives. The left image in Fig. 5 shows a case where a backpack is falsely recognized as a human by OpenPose. However, YOLOv4 does not recognize it as a human. Therefore, the backpack would not be cropped and given to the OpenPose model. The right image in Fig. 5 shows a case where a bicycle is falsely recognized as a human by the YOLOv4 model. The bicycle is then cropped from the full image and given to the OpenPose model. However, the OpenPose model does not detect any human skeleton in the cropped bicycle image. Therefore, neither of these false positive cases is further processed by the distance estimation algorithm.

After the cropped images from YOLOv4 are processed by the OpenPose model, the skeleton keypoints for detected human bodies are extracted. We use the 25 keypoint output version of OpenPose illustrated in Fig. 6. Out of the extracted keypoints, we select pairs whose mutual distance is independent of the person's pose, whose average distance is available in the literature, whose angle towards the lens is as constant as possible, and which are visible in most of the photos. With these criteria, we select three key point pairs for our algorithm: 15-16 for pupillary distance, 2-5 for shoulder width, and 1-8 for torso length. In typical media or personal photos, the torso has the most constant angle towards the lens, but the eyes and shoulders are visible also in the close-up and portrait photos, where the torso is not seen. We assume average adult body proportions for the three keypoint pairs: 389 mm for shoulder width (Watson), 63 mm for pupillary distance (Evans),

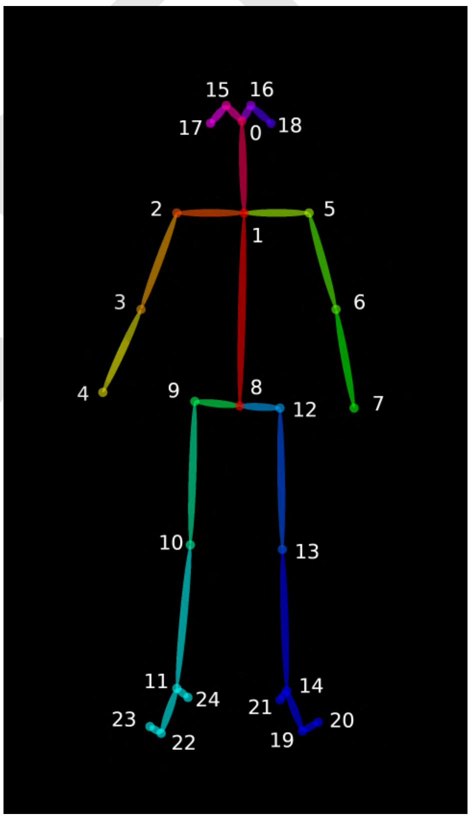


Figure 6. 25 skeleton keypoint output of OpenPose.

and 444 mm for torso length (White Mountain Backpacks). The extracted keypoint pairs are then processed by our distance estimation algorithm that estimates 3D positions with respect to the camera for each person.

We use the pinhole camera model (Sturm, 2014) shown in Fig. 7 for our calculations. We also make an assumption that every keypoint pair is parallel to the camera's sensor plane. We make these assumptions because the subjects' poses and camera's exterior orientation parameters (Ikeuchi, 2014) are not known. Estimat-

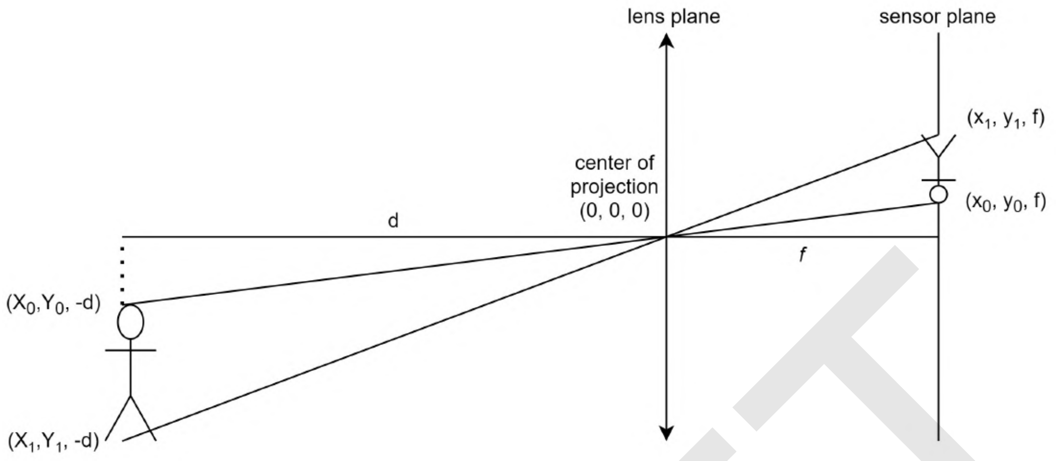


Figure 7. Pinhole camera model.

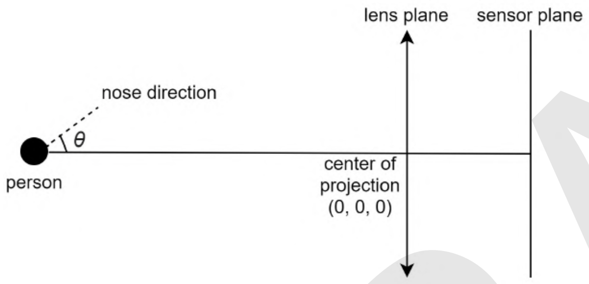


Figure 8. Birdseye view of orientation angle toward the lens.

ing the exterior orientation parameters (Ikeuchi, 2014) of the camera from single images is an ill-posed problem (Kabanikhin et al., 2008), but in most cases the angle between a person's torso and the camera's sensor plane is negligible for our calculations.

We denote 3D locations of the keypoints on the image coordinate system as

$$(x_a, y_a, f), \tag{3}$$

where f is the focal length, and 3D location estimates of the keypoints on the world coordinate system as

$$E_n = (X_a, Y_a, -d), \tag{4}$$

where d is the distance to the camera. The distance between a pair of keypoints on the image coordinate system is

$$D_i = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (f - f)^2} \tag{5}$$

and the distance between the keypoints on the world coordinate system is

$$D_w = \sqrt{(X_0 - X_1)^2 + (Y_0 - Y_1)^2 + (d - d)^2}. \tag{6}$$

Since the camera sensor's plane size is known, x_a and y_a in Eq. (3) can be derived from the x and y pixel locations of the keypoints in the image. The last coordinate, f, in Eq. (3) is obtained from the camera parameters. Thus, all the keypoints' 3D positions on the image coordinate system in Eq. (3) are known and D_i can be solved. By using triangle similarity, the following equations give 3D positions of the keypoints on the world coordinate system. Eq. (7), where D_w is one of the average body proportions, is used to derive d in Eq. (4). After d is derived, X_a and Y_a are obtained from Eqs. (8) and (9).

$$\frac{D_i}{f} = \frac{D_w}{d} \tag{7}$$

$$X_a = -\frac{d}{f} x_a \tag{8}$$

$$Y_a = -\frac{d}{f} y_a \tag{9}$$

After the 3D coordinates of the keypoints on the world coordinate system in Eq. (4) are estimated, the middle points of each detected keypoint pair are used to represent a 3D location for the person. Thus, we have at most 3 different estimated 3D locations for a person, one for each keypoint pair (shoulder, pupil, torso). While we assume that the keypoint pairs are parallel to the camera's sensor plane, this assumption may not be

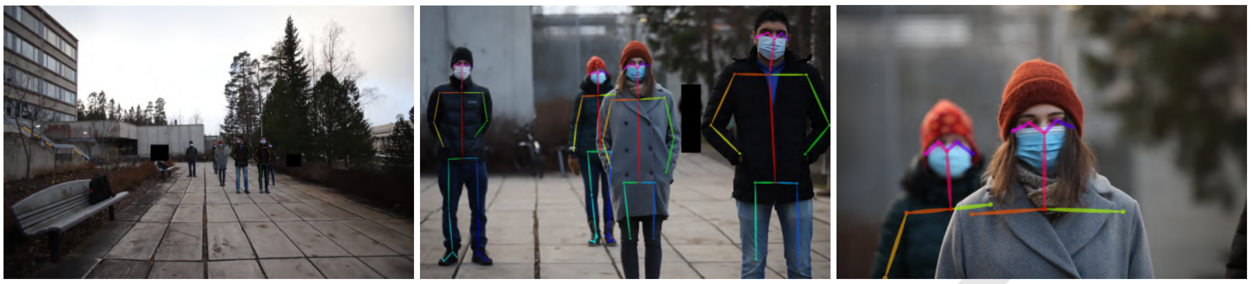


Figure 9. Examples of pictures from the dataset belonging to the first photo shoot, all of them taken from camera location C1. The used focal lengths for the pictures are 16mm, 105mm and 300mm from left to right.

valid, and the accuracy of the estimated locations is affected by the severity of the violations. Fig. 8 shows the birdseye view of a person's orientation angle θ toward the lens. If the angle is non-zero, the shoulder and pupil keypoint pairs are no longer parallel to the sensor plane and the estimates based on these keypoint pairs are prone to error. However, in a typical situation of upright torsos the estimates made from the torso length are unaffected by θ, because θ does not affect D_i computed using Eq. (5) for the torso. On the other hand, also a torso may not be parallel to the sensor plane either because the person is in a bent position or because the camera's pitch angle is non-zero. For an overhead image, shoulders might be parallel to the sensor plane, while torsos would be perpendicular. Whenever the assumption on a keypoint pair being parallel to the sensor plane is violated, D_i in Eq. (5) decreases. A smaller D_i leads to a larger estimate for d from Eq. (7). For this reason, we select the 3D location estimate with the smallest distance to the camera. For typical media or personal photos, where the pitch angle is small, this usually means using the estimate derived from the torso whenever it is available. However, for close-up and portrait pictures, the torso is often not visible. Fig. 9 shows three pictures taken from the same location but with increasing focal lengths. The rightmost image in Fig. 9 is an example of a close-up picture where the distance estimations have to be made from the shoulder and pupil distances since there are no visible torsos.

Finally, our method computes the distances between all the pairs of detected people and gives them as outputs. The pixel locations for the detected persons are given to be able to evaluate on our benchmark, while they are not needed if the method is used for analysing social distancing in novel images for photographic studies. The overall flowchart of the proposed social distance estimation method is illustrated in Fig. 10.

5. Experimental Results

5.1. Experimental Setup

All of the code was developed in Python programming language version 3.8 (Van Rossum and Drake Jr, 1995). OpenPose (Cao et al., 2019; Simon et al., 2017; Cao et al., 2017; Wei et al., 2016) and YOLOv4 (Alexey Bochkovskiy, 2020) models were used for human detection. The input size of YOLOv4 was set to 704x704. Input size was not set for OpenPose as OpenPose is able to adapt its input size for each image. The version of the OpenPose model we were using was originally trained by using the COCO keypoint challenge dataset (Lin et al., 2014), combined with OpenPose authors' own annotated dataset for foot keypoint estimation which consists of a small subset of the COCO dataset where the authors labelled foot keypoints. YOLOv4 uses CSPDarknet53 (Wang et al., 2019) as its backbone which was trained on the ImageNet dataset (Deng et al., 2009). The deep learning models were downloaded from their respective official source code pages [1] [2] and they were loaded and used by TensorFlow library version 2.3.1 (et al., 2015). For image processing purposes, OpenCV imaging library was used (Bradski, 2000). In addition to our final method that generates 3D position estimates using torso, shoulders, and eyes and selects the estimate closest to the camera as explained in Section 4, we also evaluate variants of the proposed method, where only one of these body parts is used at the time. We use our test benchmark to compute the results for all the images and for each photo shoot separately.

5.2. Results

Table 2 shows the person detection rates and pairwise percentual distance estimation errors for the overall dataset. Table 3 gives the results for the first photo

[1]https://github.com/CMU-Perceptual-Computing-Lab/openpose
[2]https://github.com/AlexeyAB/darknet

Focal Length (mm)	Number of Pictures	Shoulder Based Method		Pupil Based Method		Torso Based Method		Combined Method	
		Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error
16	28	0.75	333.42	0.55	39.79	0.82	36.30	**0.89**	**28.80**
24	46	0.81	346.05	0.55	39.52	0.91	33.22	**0.94**	**24.68**
35	48	0.81	450.49	0.58	65.63	0.91	48.52	**0.92**	**34.68**
50	84	0.80	306.56	0.44	72.37	0.91	39.29	**0.94**	**35.03**
105	69	0.72	332.72	0.57	110.50	0.79	73.29	**0.89**	**52.50**
200	7	0.69	105.28	0.73	52.28	0.69	93.53	**0.78**	**53.66**
300	18	0.70	1244.59	0.60	52.88	0.61	148.94	**0.78**	**52.51**
All	300	0.78	385.22	0.54	68.56	0.84	51.01	**0.91**	**38.24**

Table 2. Person detection rates and pair-wise percentual distance errors for each of the methods for all of the images (indoor and outdoor) combined.

Focal Length (mm)	Number of Pictures	Shoulder Based Method		Pupil Based Method		Torso Based Method		Combined Method	
		Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error
16	7	**0.85**	120.60	0.71	26.44	**0.85**	**18.33**	**0.85**	18.48
24	8	0.83	190.70	0.64	76.24	**0.91**	**16.99**	**0.91**	21.49
35	11	0.90	174.68	0.84	57.78	**0.96**	**20.17**	**0.96**	21.09
50	11	0.87	190.12	0.77	72.35	0.89	**24.34**	**0.91**	26.40
105	11	**1.00**	127.57	**1.00**	48.99	**1.00**	41.63	**1.00**	**33.08**
200	7	0.69	105.28	0.73	**52.28**	0.69	93.53	**0.78**	53.66
300	8	0.70	288.13	0.88	**34.48**	0.18	-	**0.89**	**34.48**
All	63	0.85	165.27	0.78	54.43	0.90	**28.76**	**0.91**	28.97

Table 3. Person detection rates and pair-wise percentual distance errors for each of the methods for the first photo shoot (outdoor) where every person is standing up.

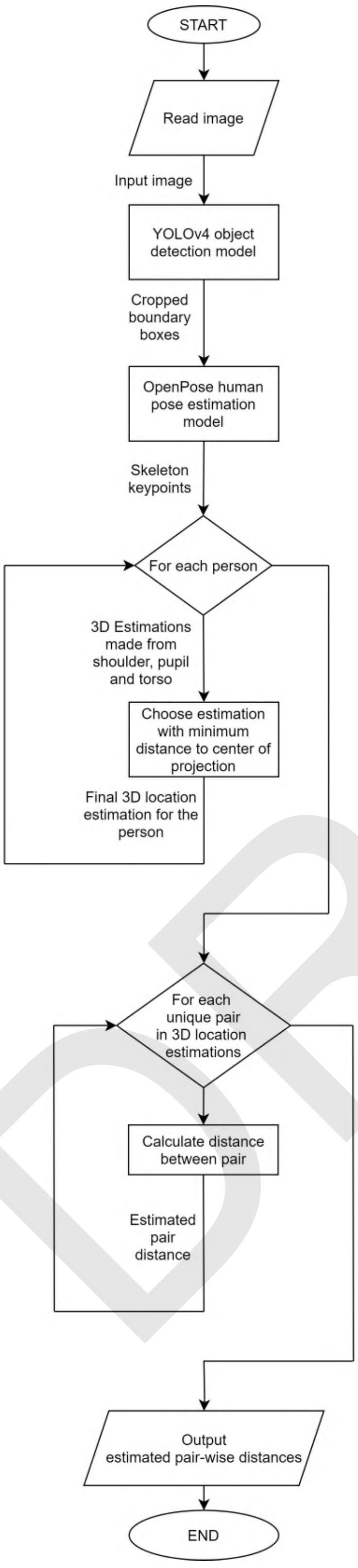

START
Read image
Input image
YOLOv4 object detection model
Cropped boundary boxes
OpenPose human pose estimation model
Skeleton keypoints
For each person
3D Estimations made from shoulder, pupil and torso
Choose estimation with minimum distance to center of projection
Final 3D location estimation for the person
For each unique pair in 3D location estimations
Calculate distance between pair
Estimated pair distance
Output estimated pair-wise distances
END

Figure 10. Flowchart of the method.

Number of Pictures	Combined Method	
	Person Detection Rate	Pair-wise Percent Distance Error
53	0.85	37.59

Table 4. Person detection rates and pair-wise percentual distance errors for the combined method for the photos taken from camera location C2, for which the zero pitch angle assumption is not valid.

shoot separately. For the other photo shoots, the separate results are provided in the Appendix (B.6-B.8). Since YOLOv4 is used in addition to OpenPose and they cancel each other's false positives, we have no cases with more detections than actual people in an image. This leads to almost zero false discovery rates as explained in Section 3.3. Therefore, false discovery rates are not reported in the tables.

It can be observed from Table 2 that the most reliable body part to estimate locations is the torso. However, estimations made from the torso alone fail for close-up pictures where the torso detection rate is low. When all three body parts (shoulder, pupil, and torso) are used together for the estimations, the obtained results shown in the last column are better than the results obtained from any single body part. The combined method mostly uses the torso whenever it is visible (overall shots) and uses the shoulder and pupil distances when the torso is not visible (close-up shots).

Looking at Tables 3, B.6, B.7 and B.8 it can be seen that there are no significant differences in terms of person detection rates when it comes to indoor and outdoor pictures. However, it should be noted that the pair-wise distance estimation errors for the indoor pictures are slightly higher than the outdoor pictures. This is primarily caused by the fact that many body parts of the people in the indoor pictures are obstructed by the chairs and sofas. There are also more cases of people facing away from the camera, people standing in front of other people, and people in poses where their torsos were non-upright in the indoor photo shoots.

5.3. Additional Results and Analysis

We separately show the results for the images that were taken from camera location C2 for the first and third photo shoot (outdoor) on Table 4. C2 location was at a height of 360 cm on the first and 220 cm on the third photo shoot relative to the ground plane where the subjects were standing on. Thus, the camera was pitched

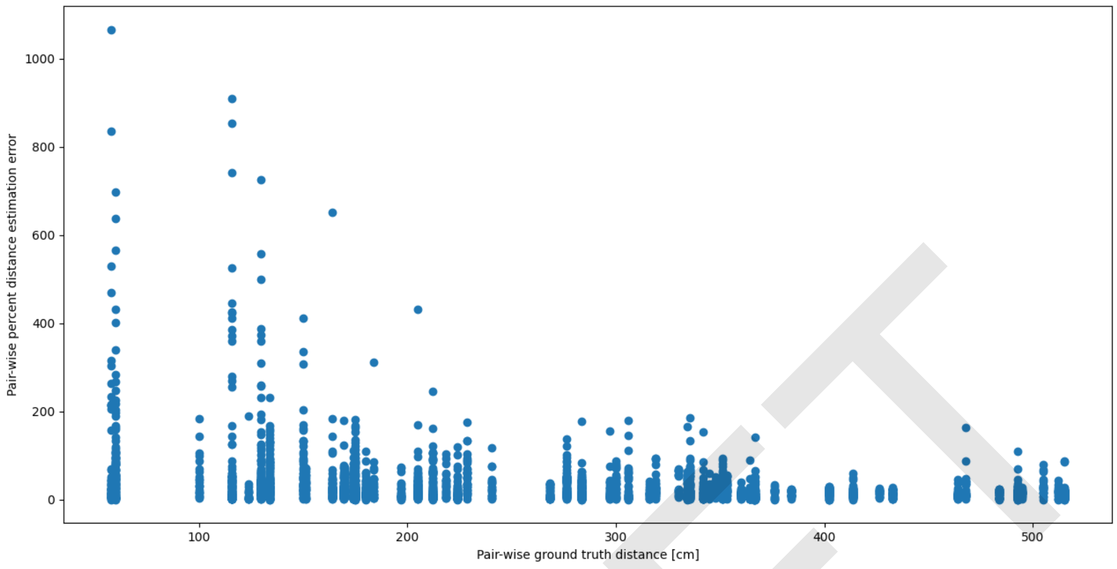


Figure 11. Pair-wise distance estimation errors for each of the ground truth pair-wise distances.

down to include the subjects within the field of view. For the other camera locations, the pitch angle was close to zero and people were mainly standing or sitting with their torsos upright. Therefore, the torsos are usually almost parallel to the camera's sensor plane and, thus, produce good distance estimates whenever they are visible. For camera location C2, this may no longer be the case. However, the results show that the relative pair-wise distance estimation errors for C2 locations are slightly lower than on the average despite the violation of the zero pitch angle assumption. We can conclude that this level of pitch angle does not cause significant problems.

A graph showing how the pair-wise distance estimation errors depend on the ground truth distances is given in Fig. 11. It can be observed from this graph that the pair-wise distance estimations errors are on average slightly lower for higher ground truth distances. This is reasonable as for the closest distances the variations in the poses also cause some error.

We also provide additional results by formulating the social distance estimation problem as a binary classification task similar to previous works. We set five different social distance thresholds as safe distances. If the distance between a pair is smaller than the threshold, we consider the distance to be unsafe and safe otherwise. We consider the unsafe case as the positive class. The standard evaluation metrics for binary classification problems are Precision, Recall, and F1-Score. The formulas for these metrics

Safe Distance (m)	F1-Score
1	0.46
1.5	0.62
2	0.75
3	0.83
4	0.90

Table 5. F1-scores of our proposed method for different safe distance thresholds

are $Precision = \frac{TruePositives}{TruePositives+FalsePositives}$, $Recall = \frac{TruePositives}{TruePositives+FalseNegatives}$, $F1-score = 2*(\frac{Precision*Recall}{Precision+Recall})$. F1-score is an overall measure of the binary classification performance and is always within the range of 0-1 with 1 indicating perfect performance. The F1-score results of our proposed method are given in Table 5.

As can be seen in Table 5, the choice of safe distance threshold changes the F1-scores drastically. For example, the low performance for 1m threshold follows from many ground-truth distances being just slightly above the threshold. As our methods tends to slightly underestimate the distances especially when the torsos are not visible as explained in Section 4, these cases lead to false positives. This supports our claim that formulating the problem of social distance estimation as a binary classification task is not an optimal way to evaluate the performance of the methods. As the results depend greatly on the threshold value, F1-scores do not reflect the true capacity and accuracy of the distance estimation performance of a method. Our proposed evaluation protocol, which gives the average pair-wise percentual distance estimation error offers greater insight on the method's performance.

6. Conclusion

To address the need for more accurate estimation of social distances from general images to analyze social and cultural impacts of the social distancing regulations introduced due to the COVID-19 pandemic, we proposed a new test benchmark for automatic social distance estimation algorithms. The benchmark includes an evaluation protocol for methods producing pair-wise social distances. The images follow a typical journalistic photographing style instead of a fixed monitoring setup, and they were taken with varying camera settings. Furthermore, we proposed a robust method that estimates 3D locations of persons in images and then uses these estimated locations to calculate the social distances between the people. Our method is able to estimate social distances in any single image without the need for knowing the extrinsic parameters or manually calibrating the homography matrix of the image plane to the ground plane, provided that the focal length and sensor size information of the camera are known, which enables our method to be used flexibly on all kinds of images. The proposed method was able to obtain 91% person detection rate along with 38.24% pair-wise distance error on the proposed test benchmark.

While our method gives satisfactory results for overall shots where the torsos of the people can be detected by OpenPose, the accuracy of the estimations gets weaker for close-up shots where the torsos are generally not visible in the image. This happens because our method assumes one of the keypoint pairs (eyes, shoulders, torso) to be parallel to the camera's sensor plane, and violations of this assumption lead to distance estimates that are longer than the ground-truth. In typical journalistic photos, where the camera's pitch angle is close to zero and the peoples' torsos are in upright positions, the assumption is typically most accurate for the torso keypoint pair whenever it is visible in the image. Thus, our method could be improved by estimating automatically also the pitch angle and persons' angles with respect to the camera. Our method also uses average adult human body proportions for the calculations. Therefore, the estimations made for children in the images would be less accurate. Our method can be improved by taking advantage of other methods that can estimate the gender and ages of the subjects and adaptively changing the assumed body dimensions for each individual subject depending on their gender and age. It should also be noted that our method requires the focal length and sensor plane size information of the camera. Therefore, our method cannot be applied on photos where these information are lacking. For our method to be applied on pictures where the focal length and sensor plane size are not known, these information would have to be estimated through other methods.

In our future research, we will use our benchmark to further enhance the proposed method and then use it in an interdisciplinary study, where we will analyze the impacts of the COVID-19 regulations on social interactions. While the COVID-19 makes the social distance analysis very topical, the benchmark and the developed methods are naturally not restricted on COVID-19 related analysis, but they can be beneficial in other image-based proxemics studies focusing on different historical, cultural, or journalistic phenomena.

References

Abouk, R., Heydari, B., 2021. The immediate effect of covid-19 policies on social-distancing behavior in the united states. Public Health Reports 136, 245–252. doi:10.1177/0033354920976575.

Aghaei, M., Bustreo, M., Wang, Y., Bailo, G.L., Morerio, P., Del Bue, A., 2021. Single image human proxemics estimation for visual social distancing, in: IEEE Winter Conference on Applications of Computer Vision (WACV).

Ahmed, I., Ahmad, M., Rodrigues, J.J., Jeon, G., Din, S., 2021. A deep learning-based social distance monitoring framework for covid-19. Sustainable Cities and Society 65, 102571. doi:10.1016/j.scs.2020.102571.

et al., M.A., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: http://tensorflow.org/. software available from tensorflow.org.

Al-Hasan, A., Khuntia, J., Yim, D., 2020a. Threat, coping, and social distance adherence during covid-19: Cross-continental comparison using an online cross-sectional survey. Journal of Medical Internet Research 22. doi:10.2196/23019.

Al-Hasan, A., Yim, D., Khuntia, J., 2020b. Citizens' adherence to covid-19 mitigation recommendations by the government: A 3-country comparative evaluation using web-based cross-sectional survey data. Journal of Medical Internet Research 22. doi:10.2196/20634.

Alexey Bochkovskiy, Chien-Yao Wang, H.Y.M.L., 2020. Yolov4: Yolov4: Optimal speed and accuracy of object detection. arXiv .

Balasa, A.P., 2020. Covid – 19 on lockdown, social distancing and flattening the curve – a review. European Journal of Business and Management Research 5. doi:10.24018/ejbmr.2020.5.3.316.

Benfold, B., Reid, I., 2011. Stable multi-target tracking in real-time surveillance video, in: CVPR 2011, pp. 3457–3464. doi:10.1109/CVPR.2011.5995667.

Bertoni, L., Kreiss, S., Alahi, A., 2021. Perceiving humans: From monocular 3d localization to social distancing. IEEE Transactions on Intelligent Transportation Systems , 1–18doi:10.1109/TITS.2021.3069376.

Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools .

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y.A., 2019. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence .

Cao, Z., Simon, T., Wei, S.E., Sheikh, Y., 2017. Realtime multi-person 2d pose estimation using part affinity fields, in: CVPR.

Chavdarova, T., Baqué, P., Bouquet, S., Maksai, A., Jose, C., Bagautdinov, T., Lettry, L., Fua, P., Van Gool, L., Fleuret, F., 2018. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Cook, M., 1970. Experiments on orientation and proxemics. Human Relations 23, 61–76. doi:10.1177/001872677002300107.

Courtemanche, C., Garuccio, J., Le, A., Pinkston, J., Yelowitz, A., 2020. Strong social distancing measures in the united states reduced the covid-19 growth rate. Health Affairs 39, 1237–1246. doi:10.1377/hlthaff.2020.00608.

Cristani, M., Bue, A.D., Murino, V., Setti, F., Vinciarelli, A., 2020. The visual social distancing problem. IEEE Access 8, 126876–126886. doi:10.1109/ACCESS.2020.3008370.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F., 2009. Imagenet: a large-scale hierarchical image database, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

Di Corrado, D., Magnano, P., Muzii, B., Coco, M., Guarnera, M., De Lucia, S., Maldonato, N.M., 2020. Effects of social distancing on psychological state and physical activity routines during the covid-19 pandemic. Sport Sciences for Health 16, 619–624. doi:10.1007/s11332-020-00697-5.

Doogan, C., Buntine, W., Linger, H., Brunt, S., 2020. Public perceptions and attitudes toward covid-19 nonpharmaceutical interventions across six countries: A topic modeling analysis of twitter data. Journal of Medical Internet Research 22. doi:10.2196/21419.

Eden, A.L., Johnson, B.K., Reinecke, L., Grady, S.M., 2020. Media for coping during covid-19 social distancing: Stress, anxiety, and psychological well-being. Frontiers in Psychology 11, 3388. doi:10.3389/fpsyg.2020.577639.

Evans, L., . What is pupillary distance and how do you measure it? URL: https://www.allaboutvision.com/eye-care/measure-pupillary-distance/. [Online]. Available: https://www.allaboutvision.com/eye-care/measure-pupillary-distance/ [Accessed: 03- Mar- 2021].

Fabbri, M., Lanzi, F., Gasparini, R., Calderara, S., Baraldi, L., Cucchiara, R., 2020. Inter-homines: Distance-based risk estimation for human safety. arXiv:2007.10243.

Fleuret, F., Berclaz, J., Lengagne, R., Fua, P., 2008. Multicamera people tracking with a probabilistic occupancy map. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 267–282. doi:10.1109/TPAMI.2007.1174.

Ford, M.B., 2020. Social distancing during the covid-19 pandemic as a predictor of daily psychological, social, and health-related outcomes. The Journal of General Psychology , 1–23doi:10.1080/00221309.2020.1860890.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., Lempitsky, V., 2016. Domain-adversarial training of neural networks. Journal of Machine Learning Research 17, 1–35. URL: http://jmlr.org/papers/v17/15-239.html.

Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354–3361. doi:10.1109/CVPR.2012.6248074.

Gorbalenya, A., Baker, S., Baric, R., de Groot, R., Drosten, C., Gulyaeva, A., Haagmans, B., Lauber, C., Leontovich, A., Neuman, B., Penzar, D., Perlman, S., Poon, L., Samborskiy, D., Sidorov, I., Sola, I., Ziebuhr, J., 2020. The species severe acute respiratory syndrome-related coronavirus: classifying 2019-ncov and naming it sars-cov-2. Nature Microbiology 5. doi:10.1038/s41564-020-0695-z.

Hall, E.T., 1966. The hidden dimension / Edward T. Hall. [1st ed.] ed., Doubleday Garden City, N.Y.

Hall, E.T., Birdwhistell, R.L., Bock, B., Bohannan, P., Diebold, A.R., Durbin, M., Edmonson, M.S., Fischer, J.L., Hymes, D., Kimball, S.T., La Barre, W., , McClellan, J.E., Marshall, D.S., Milner, G.B., Sarles, H.B., Trager, G.L., Vayda, A.P., 1968. Proxemics [and comments and replies]. Current Anthropology 9, 83–108. doi:10.1086/200975.

Harrigan, J.A., 2005. Proxemics, kinesics, and gaze.

Ikeuchi, K. (Ed.), 2014. Camera Parameters (Internal, External). Springer US, Boston, MA. pp. 81–81. URL: https://doi.org/10.1007/978-0-387-31439-6_100019, doi:10.1007/978-0-387-31439-6_100019.

Jacob, L., Tully, M.A., Barnett, Y., Lopez-Sanchez, G.F., Butler, L., Schuch, F., López-Bueno, R., McDermott, D., Firth, J., Grabovac, I., Yakkundi, A., Armstrong, N., Young, T., Smith, L., 2020. The relationship between physical activity and mental health in a sample of the uk public: A cross-sectional study during the implementation of covid-19 social distancing measures. Mental Health and Physical Activity 19, 100345. doi:https://doi.org/10.1016/

j.mhpa.2020.100345.

Kabanikhin, S., Tikhonov, N., Ivanov, V., Lavrentiev, M., 2008. Definitions and examples of inverse and ill-posed problems. Journal of Inverse and Ill-posed Problems 16, 317–357.

Lee, M., Kang, B., You, M., 2021. Knowledge, attitudes, and practices (kap) toward covid-19: a cross-sectional study in south korea. BMC Public Health 21. doi:https://doi.org/10.1186/s12889-021-10285-y.

Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft COCO: common objects in context. CoRR abs/1405.0312. URL: http://arxiv.org/abs/1405.0312, arXiv:1405.0312.

Morerio, P., Bustreo, M., Wang, Y., Bue, A.D., 2021. End-to-end pairwise human proxemics from uncalibrated single images, in: 2021 IEEE International Conference on Image Processing (ICIP), pp. 3058–3062. doi:10.1109/ICIP42928.2021.9506457.

Nguyen, C.T., Saputra, Y.M., Van Huynh, N., Nguyen, N.T., Khoa, T.V., Tuan, B.M., Nguyen, D.N., Hoang, D.T., Vu, T.X., Dutkiewicz, E.e.a., 2020. A comprehensive survey of enabling and emerging technologies for social distancing—part ii: Emerging technologies and open issues. IEEE Access 8, 154209–154236. doi:10.1109/access.2020.3018124.

Organization, W.H., a. Advice for the public on COVID-19. URL: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public [Accessed: 15- Jan- 2021].

Organization, W.H., b. WHO Coronavirus Disease (COVID-19) Dashboard. URL: https://covid19.who.int. [Online]. Available: https://covid19.who.int [Accessed: 17- Nov 2021].

Pouw, C.A.S., Toschi, F., van Schadewijk, F., Corbetta, A., 2020. Monitoring physical distancing for crowd management: real-time trajectory and group analysis doi:10.1371/journal.pone.0240963, arXiv:arXiv:2007.06962.

Prem, K., Liu, Y., Russell, T.W., Kucharski, A.J., Eggo, R.M., Davies, N., Group, C.f.t.M.M., Jit, M., Klepac, P., 2020. The effect of control strategies that reduce social mixing on outcomes of the covid-19 epidemic in wuhan, china. SSRN Electronic Journal doi:10.2139/ssrn.3552864.

Punn, N.S., Sonbhadra, S.K., Agarwal, S., 2020. Monitoring covid-19 social distancing with person detection and tracking via fine-tuned yolo v3 and deepsort techniques. arXiv:arXiv:2005.01385.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv .

Ren, S., He, K., Girshick, R., Sun, J., 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv:1506.01497.

Simon, T., Joo, H., Matthews, I., Sheikh, Y., 2017. Hand keypoint detection in single images using multiview bootstrapping, in: CVPR.

Sturm, P., 2014. Pinhole camera model, in: Computer Vision, A Reference Guide.

Sun, C., Zhai, Z., 2020. The efficacy of social distance and ventilation effectiveness in preventing covid-19 transmission. Sustainable Cities and Society 62, 102390. doi:10.1016/j.scs.2020.102390.

Van Rossum, G., Drake Jr, F.L., 1995. Python tutorial. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

Vokó, Z., Pitter, J.G., 2020. The effect of social distance measures on covid-19 epidemics in europe: an interrupted time series analysis. GeroScience 42, 1075–1082. doi:10.1007/s11357-020-00205-0.

Wang, C., Liao, H.M., Yeh, I., Wu, Y., Chen, P., Hsieh, J., 2019. Cspnet: A new backbone that can enhance learning capability of CNN. CoRR abs/1911.11929. URL: http://arxiv.org/abs/1911.11929, arXiv:1911.11929.

Watson, K., . What's an Average Shoulder Width? URL: https://www.healthline.com/health/average-shoulder-width. [Online]. Available: https://www.healthline.com/health/average-shoulder-width [Accessed: 03- Mar- 2021].

Wei, S.E., Ramakrishna, V., Kanade, T., Sheikh, Y., 2016. Convolutional pose machines, in: CVPR.

White Mountain Backpacks, . Backpack Fitting. URL: https://www.whitemountain.com.au/backpack-fitting/backpack-fitting-measure-torso-length.html. [Online]. Available: https://www.whitemountain.com.au/backpack-fitting/backpack-fitting-measure-torso-length.html [Accessed: 03- Mar- 2021].

Wojke, N., Bewley, A., 2018. Deep cosine metric learning for person re-identification, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE. pp. 748–756. doi:10.1109/WACV.2018.00087.

Wojke, N., Bewley, A., Paulus, D., 2017. Simple online and realtime tracking with a deep association metric, in: 2017 IEEE International Conference on Image Processing (ICIP), IEEE. pp. 3645–3649. doi:10.1109/ICIP.2017.8296962.

Yang, D., Yurtsever, E., Renganathan, V., Redmill, K.A., Ümit Özgüner, 2020. A vision-based social distancing and critical density detection system for covid-19. arXiv:2007.03578.

Young, J.W., 1982. Projective geometry. Pub. for the Mathematical Association of America by the Open Court Pub. Co.

Zhang, W., Gao, F., Gross, J., Shrum, L.J., Hayne, H., 2021. How does social distancing during covid-19 affect negative moods and memory? Memory 29, 90–97. doi:10.1080/09658211.2020.1857774.

Zhou, X., Wang, D., Krähenbühl, P., 2019. Objects as points. CoRR abs/1904.07850. URL: http://arxiv.org/abs/1904.07850, arXiv:1904.07850.

Appendix A. Birdseye Views of Photo Shoots 2-4

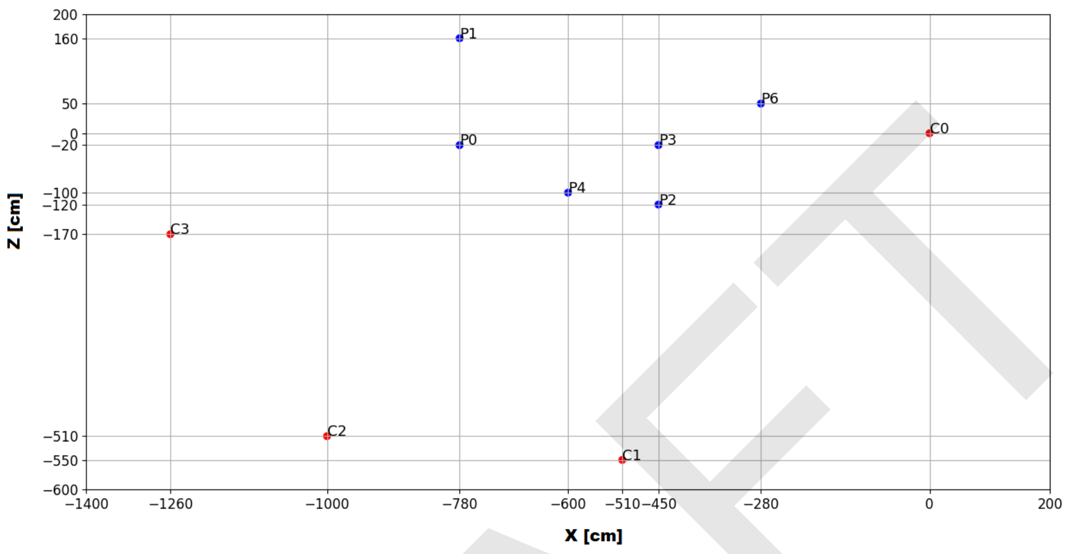


Figure A.12. Birdseye view of the second photo shoot (indoor). The ground truth locations of the people and cameras are given in blue and red dots, respectively.

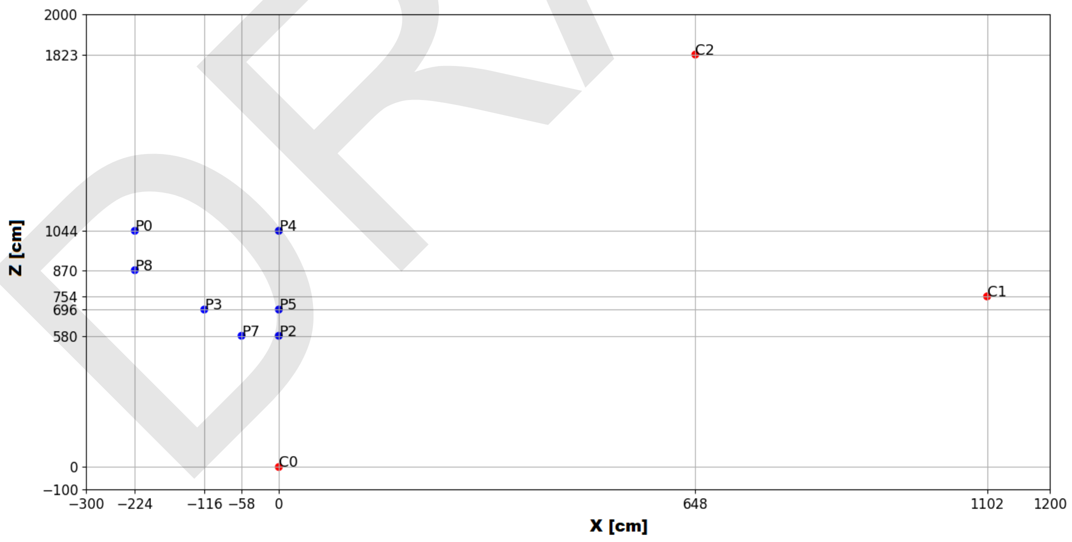


Figure A.13. Birdseye view of the third photo shoot (outdoor). The ground truth locations of the people and cameras are given in blue and red dots, respectively.

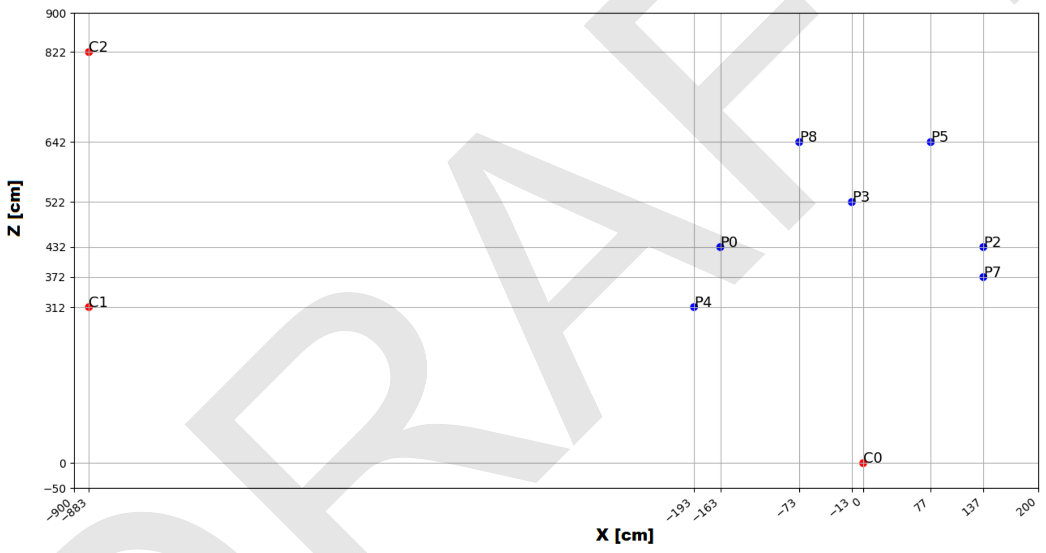


Figure A.14. Birdseye view of the fourth photo shoot (indoor). The ground truth locations of the people and cameras are given in blue and red dots, respectively.

Appendix B. Results for Photo Shoots 2-4

Focal Length (mm)	Number of Pictures	Shoulder Based Method		Pupil Based Method		Torso Based Method		Combined Method	
		Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error
16	4	0.83	178.58	0.62	29.70	0.62	**22.92**	**1.00**	23.98
24	4	0.83	354.18	0.54	24.94	0.70	**19.29**	**0.95**	23.40
35	4	0.66	49.27	0.54	**19.61**	**0.95**	25.79	**0.95**	20.40
50	7	0.76	189.61	0.51	29.79	0.76	29.57	**0.89**	**27.26**
105	14	0.68	102.84	0.62	55.40	0.56	**27.42**	**0.90**	35.07
All	33	0.74	163.61	0.57	37.76	0.70	**26.03**	**0.93**	28.88

Table B.6. Person detection rates and pair-wise percentual distance errors for each of the methods for the second photo shoot (indoor) where every person is sitting down.

Focal Length (mm)	Number of Pictures	Shoulder Based Method		Pupil Based Method		Torso Based Method		Combined Method	
		Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error
16	9	0.62	760.07	0.42	51.13	0.81	**18.66**	**0.84**	28.78
24	8	0.84	778.06	0.33	33.50	0.91	**17.06**	**0.93**	22.37
35	13	0.78	880.81	0.40	68.43	0.84	**16.50**	**0.86**	19.09
50	50	0.83	333.23	0.33	82.12	0.96	**25.18**	**0.97**	32.54
105	21	0.70	771.80	0.37	149.17	0.84	**36.14**	**0.88**	67.45
300	10	0.70	1669.68	0.41	117.25	0.63	148.94	**0.73**	**66.52**
All	111	0.78	658.29	0.34	81.44	0.88	**34.21**	**0.91**	39.35

Table B.7. Person detection rates and pair-wise percentual distance errors for each of the methods for the third photo shoot (outdoor) where every person is standing up.

Focal Length (mm)	Number of Pictures	Shoulder Based Method		Pupil Based Method		Torso Based Method		Combined Method	
		Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error	Person Detection Rate	Pair-wise Percent Distance Error
16	8	0.78	117.07	0.54	42.86	0.9	82.69	**0.92**	**40.27**
24	26	0.79	259.67	0.60	31.07	0.95	45.33	**0.96**	**26.57**
35	20	0.82	402.73	0.61	76.93	**0.94**	89.47	**0.94**	**55.15**
50	16	0.74	429.24	0.52	74.61	0.8	98.24	**0.86**	**54.40**
105	23	0.68	102.05	0.57	155.77	0.80	137.55	**0.85**	**57.06**
All	93	0.76	266.61	0.58	79.10	0.89	90.03	**0.91**	**46.22**

Table B.8. Person detection rates and pair-wise percentual distance errors for each of the methods for the fourth photo shoot (indoor) where some people are sitting down and some are standing up.

7.10 Improving the Accuracy of Early Exits in Multi-Exit Architectures via Curriculum Learning

The appended paper follows.

Improving the Accuracy of Early Exits in Multi-Exit Architectures via Curriculum Learning

Arian Bakhtiarnia, Qi Zhang and Alexandros Iosifidis
DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark
{arianbakh,qz,ai}@ece.au.dk

***Abstract*—Deploying deep learning services for time-sensitive and resource-constrained settings such as IoT using edge computing systems is a challenging task that requires dynamic adjustment of inference time. Multi-exit architectures allow deep neural networks to terminate their execution early in order to adhere to tight deadlines at the cost of accuracy. To mitigate this cost, in this paper we introduce a novel method called *Multi-Exit Curriculum Learning* that utilizes curriculum learning, a training strategy for neural networks that imitates human learning by sorting the training samples based on their difficulty and gradually introducing them to the network. Experiments on CIFAR-10 and CIFAR-100 datasets and various configurations of multi-exit architectures show that our method consistently improves the accuracy of early exits compared to the standard training approach.**

I. Introduction

Deep learning models have been successful in solving many problems in various domains of science and technology, ranging from autonomous vehicles to drug discovery [1]. However, a general drawback of deep neural networks is that, by definition, they are built from many layers of interconnected neurons. This results in models containing millions of parameters that need to be deployed on powerful processors due to their high computational cost. This restriction has sparked a great deal of research targeting neural network compression in recent years, thus many methods have been developed for the purpose of making deep learning models more lightweight; including pruning [2], quantization [3], regularization [4] and knowledge distillation [5] to name a few.

The high computational cost of deep learning models becomes even more problematic in computationally restricted environments, such as mobile and IoT devices. Yet, deep learning has many use cases in such settings, including but not limited to video surveillance, voice assistants, network intrusion detection and augmented reality [6]. Many of these use cases are time-sensitive and require applications to run with respect to strict time limits, for instance, in the cases of cooperative autonomous driving and augmented reality [7].

To enable time-sensitive Internet of Things applications, computationally expensive tasks, such as deep learning services, are sometimes offloaded from end-devices to edge servers using edge computing systems in order to decrease the overall execution time [8]. However, these systems often have a distributed and multi-tiered network architecture where the time required for the transmission of data between various devices is variable and depends on the communication channel state and the data size. This calls for novel neural network designs that can dynamically adapt their inference time to account for these variations in transmission time. Among lightweight deep learning methods, the concept of *early exits* [9] is a promising solution that particularly fits these settings, which is sometimes also referred to as *multi-exit architectures* or *auxiliary classifiers* in the literature.

In multi-exit architectures, branches composed of just a few layers of neurons are added at intermediate layers of a deep network called the *backbone* network. Such branches are trained to perform the same task as the backbone network and produce an output similar to that of the final layer of the network, albeit they are inevitably less accurate. These branches can then be used to make inference time more dynamic at the cost of accuracy. For instance, when there is a strict time budget and it is suspected that the deadline will be missed if the entire network is traversed, the output of these early exit branches can be used instead. Another way of utilizing early exits for dynamic inference is to use the output of early exit branches for "easier samples" and only compute the output of later branches or the final output of the backbone network when the input sample is difficult. There are various methods for detecting where to exit, one of the easiest and most intuitive ones being to determine the confidence of the output of a branch. For instance, a strategy that is used for classification problems is to set a threshold on the entropy of the classification result [10].

As previously mentioned, early exit branches are typically less accurate compared to the final output of the corresponding backbone network, therefore it is vital for them to be as accurate as possible to maintain the reliability of the output. Since the architecture of early exit branches is often very shallow in order to avoid introducing high additional overhead, increasing their accuracy is generally a challenging task. Phuong et al. [11] recently showed that knowledge distillation-based training can be used to improve the accuracy of early exits.

In this paper, we propose a new approach for improving the accuracy of early exit branches based on curriculum learning. Curriculum learning [12] is a training strategy for neural networks that has been shown to improve the final accuracy of a network in certain cases. The idea behind curriculum learning is similar to how humans learn new tasks: a well-informed teacher can be used to initiate the training with the simplest material and gradually introduce more difficult subjects to the student. For neural networks however, sometimes the opposite approach of introducing the hardest subjects first, called *anti-curriculum*, can be beneficial as well. To the best of our knowledge, curriculum learning has not yet been explored in the context of multi-exit architectures. We tested our proposed approach in 16 different scenarios involving multi-exit architectures for the problem of image classification, and found that it consistently increases the accuracy of early exits in every case. These scenarios involve two different datasets, namely CIFAR-10 and CIFAR-100 [13], four different backbone networks and two different branch locations for each backbone. We also show that the proposed approach works regardless of the optimization algorithm used during training[1].

The remainder of the paper is structured as follows. Section II provides an overview of relevant literature. The proposed approach, called *Multi-Exit Curriculum Learning*, is described in Section III. Experimental results are provided in Section IV. Finally, Section V concludes the paper and briefly discusses future research directions.

II. Related Work

In this section, we provide more detailed explanations regarding multi-exit architectures as well as curriculum learning, which are the foundations of our method. We start by describing the mathematical model for multi-exit architectures and listing popular training strategies proposed for such architectures. Subsequently, we elaborate on the curriculum learning strategy, including the concepts of sorting and pacing functions, and recount various approaches to these functions that exist in the literature.

A. *Multi-Exit Architectures*

Following the notation of Scardapane et al. [9], basic neural networks are formulated as function $f(x) = f_L(f_{L-1}(...f_1(x)))$ where L is the number of layers and f_i denotes the operator at layer i, which can be a convolution layer, a dense layer, batch normalization or any other differentiable operator. The output of the i-th layer is denoted by $h_i = f_i(h_{i-1})$ where $h_0 = x$, and θ_i signifies all trainable parameters of layer i.

In order to extend this framework to multi-exit architectures, first, a set of branch locations $B \subseteq \{1, .., L\}$ are selected. For each branch location b, a classifier or regressor $c_b(h_b) = y_b$ is defined, where y_b is the hypothesis of the early exit branch at location b. The schematic illustration of a multi-exit architecture is depicted in Figure 1.

The training of a neural network can be formulated as tuning its parameters by applying an optimization algorithm on a loss landscape:

$$f^* = \arg\min_{\theta} \sum_{n=1}^{N} l(y_n, f(x_n)), \tag{1}$$

[1]Our code is made available at https://gitlab.au.dk/maleci/MultiExitCurriculumLearning.

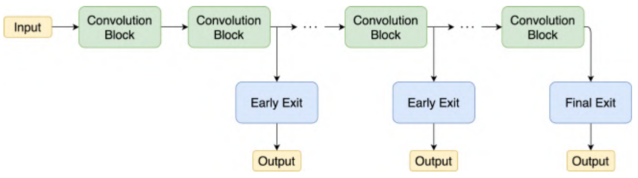


Fig. 1. Schematic illustration of a multi-exit architecture.

where $\theta = \bigcup_{i=1}^{L} \theta_i$ is the set of all parameters of the neural network, $\{(x_n, y_n)\}_{n=1}^{N}$ is the set of training samples, and $l(\cdot)$ is a loss function.

However, due to the attached early exit branches, the training of multi-exit architectures is not as straightforward. Three main approaches were proposed for training a multi-exit architecture [9], [14]:

- *End-to-End Training:* Training is formulated as a single optimization problem where the total loss is defined as a combination of the losses of early exit branches and the final layer. In this case, the contribution of each of the early exit branches to the total loss is expressed with a weight value (a hyper-parameter) that causes trade-offs and can have a significant impact on the accuracy of the early exit branches as well as the final layer. For instance, a certain weighting scheme for the contribution of branches may result in an increase in the accuracy of early exit branches but a decrease in the accuracy of the final layer.
- *Layer-Wise Training:* Initially, the entire network up to and including the first early exit branch is trained. Subsequently, the trained weights are frozen, meaning that they are not allowed to be modified anymore, and the rest of the network up to and including the second early exit branch is trained. This operation is repeated until the entire network has been trained. Note that with this strategy, there is no guarantee that the accuracy of the final layer will be similar to the case where the network does not have any early exit branches.
- *Classifier-Wise Training:* The entire backbone network is initially trained. Then, the parameters of the backbone network are frozen and each branch is trained separately since it does not affect the training of other early exit branches. Note that no trade-offs are introduced in this strategy, and since the parameters of the backbone network are not modified, its accuracy remains unchanged. However, the early exit branches have less parameters available for training compared to the other two strategies.

In this work we follow the classifier-wise training strategy for training the multi-exit architectures because of its practical importance. This is due to the fact that it can be easily added on top of existing networks (as a "plug-and-play" solution) without the need for re-training a high-performing backbone network, or computationally expensive and tedious experimentation for determining the optimal hyper-parameters that lower the effect of trade-offs introduced by combined training of the parameters of the early exit branches with those of the backbone network. Furthermore, one of the issues with multi-exit architecture is choosing the right number of early exit branches and their placement. With end-to-end and layer-wise training strategies, the choice of the total number of branches as well as their placement in the backbone network becomes important and can cause further trade-offs. On the other hand, with the classifier-wise training strategy, since the branches are independent of each other and the backbone network, early exit branches can be placed at any intermediate layer. However, we need to keep in mind that early exit branches placed later in the network do not necessarily achieve a higher accuracy, therefore some branch placements may be irrational and unnecessary since there are earlier branches which can potentially achieve higher accuracy.

Another concern with multi-exit architectures is devising a method that decides which exit should be used for each input example. As previously mentioned, a simple solution is to use the confidence of the network on its own prediction, although many other methods have been proposed for this purpose [9]. However, since our goal is to develop a method in order to increase the accuracy of all early exit branches regardless of their placement, this issue is outside the scope of this paper.

B. Curriculum Learning

As previously stated, curriculum learning draws inspiration from the way humans learn new subjects throughout their formal education. For each topic of study, a knowledgeable teacher often starts with explaining the simplest notions to the students and gradually introduces more difficult aspects of the topic during the course of the study. Curriculum learning treats the problem of training neural networks in the same manner by starting the training from a subset of training samples it deems to be simple, and progressively adding more difficult samples to the training process. Thus, curriculum learning is composed of two main components: a *sorting function* that takes training samples as input, assigns a difficulty value to each of them based on some metric and sorts them based on their difficulty values; and a *pacing function* that determines the pace at which new training samples are introduced to the network during the training process.

Scoring functions can either be predefined, meaning that the difficulty for each training sample is determined based on some prior knowledge given by an expert, or automatic, meaning that the difficulty of each sample is determined based on an algorithm. Examples of predefined sorting functions include sorting based on the length of the input text in natural language processing problems, or based on the number of objects in an image in object detection problems. A comprehensive list of predefined sorting functions for various types of data can be found in [16].

Most automatic sorting functions can be categorized into the following three groups [16]:

- *Self-Paced Learning:* In this approach, the student network itself determines the difficulty of each sample based on its current loss. It is important to note that Hacohen et al. [17] found that self-paced learning can lead to a decrease in the final test accuracy.
- *Transfer Teacher:* In this strategy, the loss of a pre-trained network called *teacher* is used to measure the difficulty of training samples. A variant of transfer teacher where the teacher network is the same as the backbone network is called *self-taught* (not to be confused with *self-paced learning*). The main difference between self-taught and other teacher transfer methods is that the self-taught method can be applied repeatedly, meaning that initially the network is trained normally and its losses are used to sort the examples and train the same network with curriculum learning. Afterwards, the losses of the new and improved network are used to re-sort the training samples and train the same network yet another time, and this process can be repeated until there are no further improvements.
- *Reinforcement Learning Teacher:* Curriculum learning can also be formulated as a reinforcement learning problem where the action is to decide which samples should be used for training, the state is the loss of the student for each sample, and the reward is the performance of the student.

Several other less common automatic sorting functions can be found in [16]. In this work, we use the *transfer teacher* method with two different teacher networks as scoring function. As previously mentioned, unlike human learning, the opposite approach of training the network starting from the most difficult samples to the easiest samples, called *anti-curriculum* or *harder-first*, has been shown to be more effective than curriculum learning in some cases [16].

Typically, a pacing function $\lambda(t) : \mathbb{N} \to (0, 1]$ takes the index of the current iteration as an input and outputs the fraction of the sorted training samples that should be used for training. Pacing functions can be categorized into two groups: discrete pacing functions and continuous pacing functions. The most popular discrete pacing function, called *baby step*, partitions sorted training samples into several buckets and gradually adds buckets of harder samples to the pool of training samples introduced to the network. A less common discrete pacing function called *one-pass* partitions the sorted training samples into several buckets, but discards the the samples of the previously introduced easier bucket from the training pool after adding the samples of a new harder bucket.

Popular examples of continuous pacing functions include *linear*, *root*, *root-p* and *geometric progression*, which are described by Equations (2)-(5) respectively:

$$\lambda_{\text{linear}}(t) = \min\left(1, \lambda_0 + \frac{1-\lambda_0}{T_f} \cdot t\right), \qquad (2)$$

$$\lambda_{\text{root}}(t) = \min\left(1, \sqrt{\lambda_0^2 + \frac{1-\lambda_0^2}{T_f} \cdot t}\right), \qquad (3)$$

$$\lambda_{\text{root-p}}(t) = \min\left(1, \sqrt{\lambda_0^p + \frac{1-\lambda_0^p}{T_f} \cdot t}\right), \qquad (4)$$

$$\lambda_{\text{geom}}(t) = \min\left(1, 2^{\left(\log_2 \lambda_0 - \frac{\log_2 \lambda_0}{T_f} \cdot t\right)}\right). \tag{5}$$

In the above equations, t is the index of current iteration, λ_0 denotes the initial fraction of training samples introduced to the network and T_f is the iteration at which the entire dataset is used for the first time.

Putting it all together, Figure 2 shows the random mini-batch process in curriculum learning. Each epoch is composed of $\frac{N}{N_b}$ batches where N is the total number of training samples and N_b is the batch size. Batch number t is sampled uniformly at random only from the first $\lambda(t)$ portion of the sorted data.

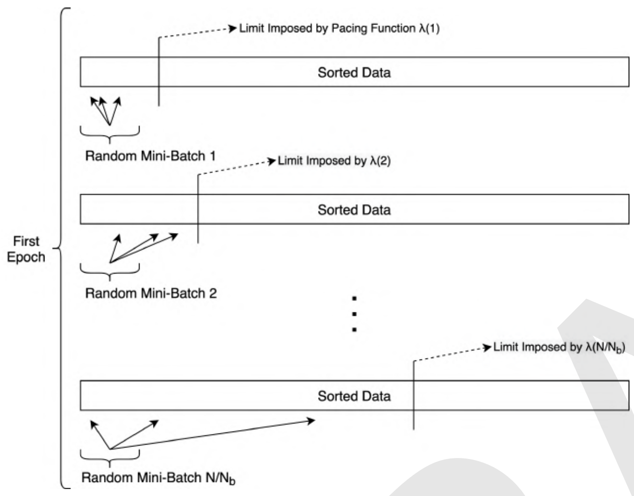


Fig. 2. Random Mini-Batch Process in Curriculum Learning.

As a final note, there are several theoretical analyses in the literature explaining why curriculum learning can improve the training procedure. Bengio et al. [12] point out that curriculum learning can be viewed as a *continuation method*. Continuation methods [15] are optimization strategies for non-convex problems that start with a smooth objective and gradually introduce less smooth versions in the hopes of revealing the global picture in the process [16]. Additionally, Hacohen et al. [17] reached the conclusion that curriculum learning modifies the optimization landscape to amplify the difference between the optimal parameter vector and all other vectors that have a small covariance with the optimal solution, including uncorrelated or negatively correlated parameter vectors.

III. Multi-Exit Curriculum Learning

In this section, we will explain the details of our method. We assume that an already trained high-performing deep neural network is given in the beginning. Due to time restrictions, this neural network must be converted to a multi-exit architecture, as it is preferable to provide an output within the strict time budget, even though it can be less accurate, rather than not providing an output within this time limit at all. Thus we augment this backbone network with a set of early exits. As previously stated, the parameters of the backbone network will not be fine-tuned, that is, if the backbone network represents function $f(x) = f_L(...f_1(x))$ with a set of parameters $\theta = \bigcup_{i=1}^{L} \theta_i$, θ will remain unchanged throughout the training process and only the parameters of early exit branch functions $c_i(h_i) : i \in B$ will be tuned. As the entire backbone network is frozen during classifier-wise training of the added early exit branches, and thus is not allowed to "help" the early exit branches by tuning its parameters, it is more difficult to increase the accuracy of the early exit branches compared to other training strategies listed in Section II. We use curriculum learning to train the early exit branches, in order to improve their accuracy.

For the purpose of sorting the training samples based on their difficulty, we use the categorical cross-entropy loss of a pre-trained teacher network. We use two different teachers, InceptionV3 [18] which is the same teacher used in Hacohen et al. [17], and the more recent EfficientNetB7 [19]. We take versions of these networks pre-trained on the ImageNet dataset [20] and use transfer learning to train them for the CIFAR-10 and CIFAR-100 datasets by removing the top layer, adding two dense layers with a Dropout layer [21] in between and retraining the network for the intended dataset. By using two dense layers, we are taking the output of pre-trained networks as feature vectors and training a multilayer perceptron classifier based on these features. In addition, since we freeze the first five blocks of the EfficientNetB7 backbone to overcome the limitations of our hardware resources, utilizing two dense layers instead of just one provides additional flexibility.

Figures 3 and 4 illustrate the easiest and most difficult training samples, respectively, in the CIFAR-10 dataset based on the loss values of InceptionV3 teacher. It is not difficult to interpret why the network finds some of these images particularly hard. For instance, a close-up from the front of the airplane might be very different from the usual perspective of other images with the same label, or it may be difficult to distinguish between dogs, cats and deer with certain colors and patterns of fur.

We use two variants of the *baby step* pacing function, the *fixed exponential pacing* function shown in Fig. 5 and the *single step pacing* function depicted in Fig. 6, both introduced by Hacohen et al. [17]. Similar to our work, Hacohen et al. [17] also investigate the effectiveness of curriculum learning on the problem of image classification (although not in multi-exit architectures) and document the pacing functions that lead to improvements in the final accuracy. These pacing functions introduce the entire dataset fairly quickly, meaning that curriculum learning effectively takes place only in the first few epochs. We found that such pacing functions are effective in our case as well. *Fixed exponential pacing* starts with only a small percentage of the training data and exponentially increases the amount of data after every fixed number of batches, whereas *single step pacing* starts with a higher percentage of data and introduces the entire dataset after a certain number of batches have been processed. The details of *fixed exponential pacing* and *single step pacing* functions are shown in Equations (6) and (7) respectively,

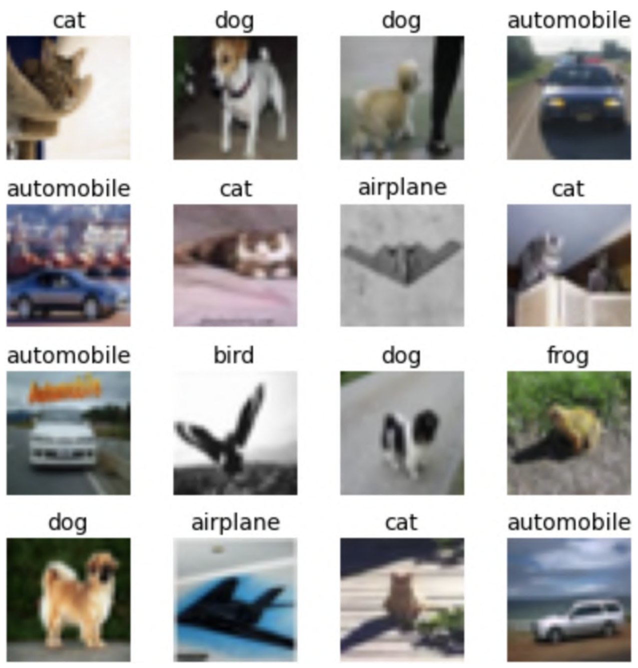


Fig. 3. Easiest training samples in the CIFAR-10 dataset based on the loss values of InceptionV3 teacher network. The labels are ground truth, not the predictions of the network.

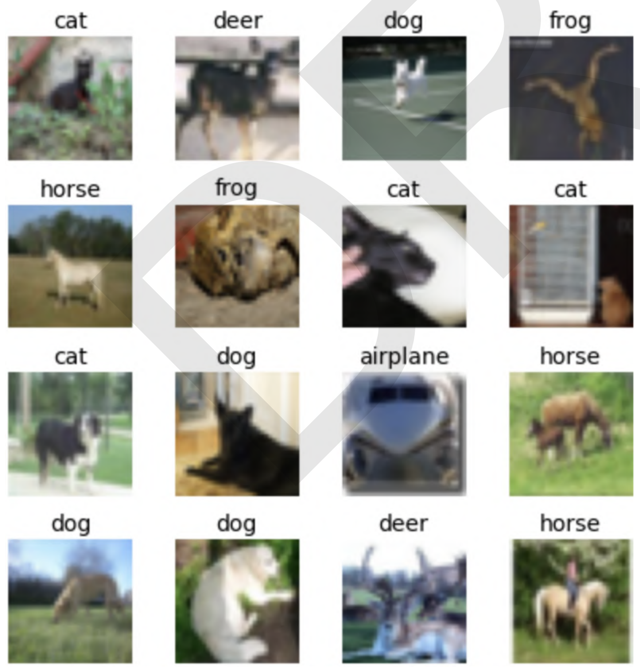


Fig. 4. Hardest training samples in the CIFAR-10 dataset based on the loss values of InceptionV3 teacher network. The labels are ground truth, not the predictions of the network.

where t is the index of the current batch, s indicates the initial fraction of data used, r denotes the increase in data and δ is the fixed number of batches after which the data is increased. It is important to note that *fixed exponential pacing* has three hyper-parameters, namely s, r and δ, while *single step pacing* has only two.

$$\lambda(t) = \min\left(s \cdot r^{\lfloor \frac{t}{\delta} \rfloor}, 1\right) \tag{6}$$

$$\lambda(t) = \begin{cases} s, & t < \delta \\ 1, & t \geq \delta \end{cases} \tag{7}$$

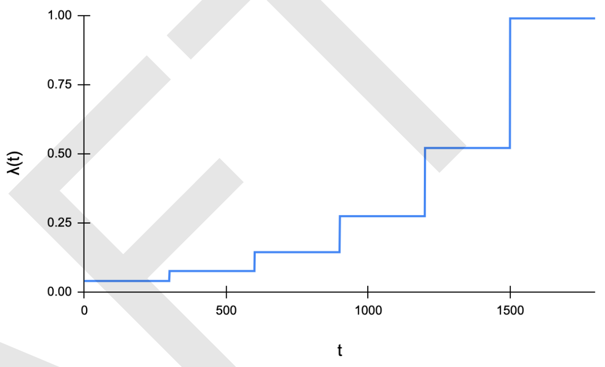


Fig. 5. *Fixed exponential pacing* function with $s = 0.04$, $r = 1.9$ and $\delta = 300$.

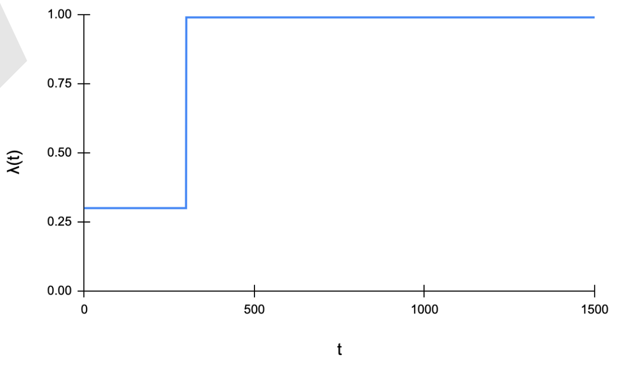


Fig. 6. *Single step pacing* function with $s = 0.30$ and $\delta = 300$.

We use four different backbone networks in our experiments, namely DenseNet201 [22], MobileNetV1 [23], ResNet152 [24] and InceptionV3 [18]. We train these networks on the CIFAR-10 and CIFAR-100 datasets using transfer learning in the exact same way as the aforementioned teacher networks.

In the training of teacher and backbone networks, in order to overcome the limitations of our available resources, the size of the batches are adjusted and some of the layers in the networks are frozen, that is, their weights are not modified during the training process. Keep in mind that since these networks are all pre-trained on the ImageNet dataset, the frozen layers are still capable of providing useful features. Table I summarizes the details of the training process for each of these networks.

TABLE I
TRAINING DETAILS FOR TEACHER AND BACKBONE NETWORKS

Network	Batch Size	Frozen Layers	Test Accuracy	
			CIFAR-10	CIFAR-100
DenseNet201	32	All except batch normalization	96.48%	82.53%
MobileNetV1	64	None	94.28%	76.91%
ResNet152	32	All except batch normalization	95.36%	82.25&
InceptionV3	64	None	96.56%	83.80%
EfficientNetB7	32	First five blocks	96.50%	83.76%

We place two early exit branches at two different intermediate layers on each backbone network. All early exit branches have the same architecture, which is a convolution layer, followed by a maximum pooling layer, and three dense layers with a Dropout layer between each pair, as shown in Figure 7. Note that since the dimention of features in different branch locations might be different, the size of the flattened vector varies for each branch location. This is the same branch architecture used by by Hu et al. [25]. The location of each branch depends on the architecture of the backbone network. We found that placing an early exit branch later in the backbone network does not necessarily improve the overall accuracy of the branch, and generally speaking branches located immediately after the "natural blocks" - for instance, concatenation layers, residual connections or dense blocks - in the architecture performed better than several other layers immediately before or after them. Early exit branches are placed at the earlier sections of the backbone network as they correspond to the locations where dynamic inference would be desired in practical scenarios. The exact placement of branches for each backbone network can be found in Tab. II.

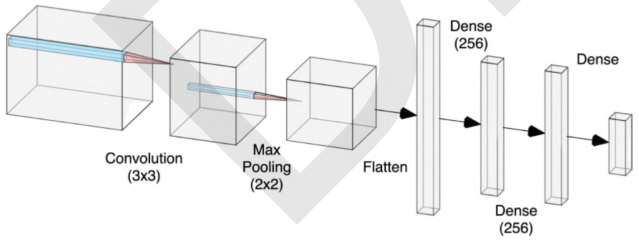


Fig. 7. Architecture of Early Exit Branches.[2]

As previously mentioned, we use the classifier-wise training strategy for training the multi-exit architecture. During the training of each branch, first we test both stochastic gradient descent and Adam [27] optimizers with different learning rates of $\{10^{-1}, 0.12, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ to obtain the highest accuracy for the normal training method without any

[2]Image created using the NN-SVG tool [26].

TABLE II
PLACEMENT OF BRANCHES FOR EACH BACKBONE NETWORK

Backbone	Dataset	BN*	Branch Placed After
DenseNet201	CIFAR-10	1	Layer 15 of 201
		2	Layer 40 of 201
	CIFAR-100	1	Layer 40 of 201
		2	Layer 137 of 201
MobileNet	CIFAR-10	1	Layer 8 of 28
		2	Layer 14 of 28
	CIFAR-100	1	Layer 8 of 28
		2	Layer 14 of 28
ResNet152	CIFAR-10	1	Layer 13 of 152
		2	Layer 38 of 152
	CIFAR-100	1	Layer 13 of 152
		2	Layer 38 of 152
InceptionV3	CIFAR-10	1	1st Filter Concat
		2	2nd Filter Concat
	CIFAR-100	1	1st Filter Concat
		2	2nd Filter Concat

*Branch Number

curriculum, which we call *vanilla*. We chose to test the 0.12 learning rate in addition to 10^{-1} since it was the best case discovered for the experiments in Hacohen et al. [17]. With both optimizers, the learning rate is automatically reduced when the validation accuracy plateaus. Subsequently, using the same optimizer, we train the branch using the *curriculum* and *anti-curriculum* training methods. Similar to Hacohen et al. [17], we also compare the results with *random curriculum* which uses the pacing function on randomly-ordered training data without any sorting. This comparison is in order to show that the benefit does not solely come from the pacing, and that the sorting from easiest to hardest or vice-versa contributes to the increase in accuracy as well.

As with many machine learning paradigms, curriculum learning is sensitive to hyper-parameters, therefore we perform a grid search in order to find the suitable teacher network and pacing function in each case. We use a dataset separate from training, validation and test datasets for this task. Table III summarizes the different choices of pacing functions tested during the hyper-parameter optimization step.

TABLE III
HYPER-PARAMETER OPTIMIZATION FOR PACING FUNCTION

Function Type	Parameters			Abbreviation
	s	r	δ	
Fixed Exponential Pacing	0.04	1.9	100	FEP(100)
Fixed Exponential Pacing	0.04	1.9	200	FEP(200)
Fixed Exponential Pacing	0.04	1.9	300	FEP(300)
Single Step Pacing	0.30	-	300	SSP(300)

We repeat each of the experiments five times and record the average accuracy along with the standard deviation. In order to make the comparisons fair, in each repetition, in all four cases of *vanilla*, *curriculum*, *anti-curriculum* and *random curriculum*, the early exit branch starts with the same

TABLE IV
COMPARISON OF THE FINAL TEST ACCURACY OF EARLY EXIT BRANCHES USING DIFFERENT TRAINING METHODS

Backbone	Dataset	BN*	Vanilla	Curriculum	AC†	RC§	Opt.	LR¶	Teacher	Pacing
DenseNet	CIFAR-10	1	71.71% ± 0.47	71.59% ± 0.57	**71.75% ± 0.75**	71.58% ± 0.57	Adam	10^{-4}	EfficientNet	FEP(100)
		2	77.43% ± 0.64	**77.91% ± 0.03**	77.21% ± 0.80	77.25% ± 0.35	SGD	0.12	EfficientNet	FEP(100)
	CIFAR-100	1	38.36% ± 0.31	**39.56% ± 0.57**	35.32% ± 2.06	38.74% ± 1.37	SGD	0.12	EfficientNet	FEP(200)
		2	61.72% ± 1.26	**64.05% ± 1.18**	58.78% ± 1.68	62.95% ± 0.78	Adam	10^{-4}	EfficientNet	FEP(300)
MobileNet	CIFAR-10	1	67.30% ± 0.25	**67.33% ± 0.31**	67.04% ± 0.45	67.02% ± 0.48	Adam	10^{-4}	EfficientNet	SSP(300)
		2	79.06% ± 0.65	**79.47% ± 0.05**	79.04% ± 0.43	78.55% ± 0.41	Adam	10^{-4}	Inception	FEP(100)
	CIFAR-100	1	44.26% ± 0.69	44.83% ± 0.19	**44.89% ± 0.26**	44.84% ± 0.45	Adam	10^{-4}	Inception	FEP(300)
		2	47.48% ± 0.99	**48.39% ± 0.66**	47.54% ± 0.45	48.34% ± 0.74	Adam	10^{-4}	EfficientNet	FEP(200)
ResNet	CIFAR-10	1	67.87% ± 0.76	**68.75% ± 0.16**	67.78% ± 0.26	67.44% ± 0.74	Adam	10^{-4}	EfficientNet	FEP(100)
		2	76.25% ± 0.25	76.24% ± 0.28	**76.32% ± 0.25**	76.29% ± 0.11	Adam	10^{-4}	EfficientNet	FEP(100)
	CIFAR-100	1	35.53% ± 0.74	**36.57% ± 1.07**	36.46% ± 0.82	36.08% ± 0.70	Adam	10^{-4}	EfficientNet	SSP(300)
		2	41.26% ± 0.56	41.30% ± 1.02	**41.45% ± 0.73**	40.89% ± 0.36	Adam	10^{-4}	EfficientNet	FEP(100)
Inception	CIFAR-10	1	76.91% ± 0.58	**77.34% ± 0.27**	77.13% ± 0.07	77.19% ± 0.17	Adam	10^{-4}	EfficientNet	FEP(300)
		2	79.06% ± 0.37	79.18% ± 0.12	**79.47% ± 0.52**	79.42% ± 0.15	Adam	10^{-4}	Inception	FEP(100)
	CIFAR-100	1	44.24% ± 0.70	**44.56% ± 0.44**	44.53% ± 0.42	44.07% ± 0.75	Adam	10^{-4}	Inception	FEP(200)
		2	45.86% ± 0.21	**46.50% ± 0.21**	45.13% ± 0.92	46.11% ± 1.28	Adam	10^{-4}	Inception	FEP(300)

*Branch Number
†Anti-Curriculum
§Random Curriculum
¶Learning Rate

weight initialization.

IV. RESULTS

Our results are summarized in Table IV. The first three columns of the table determine the case under study; the next four columns compare the final accuracy of different training approaches for each case; and the last four columns summarize the optimal hyper-parameters discovered for each case. We can observe that in all 16 cases, the accuracy of our method (curriculum and anti-curriculum) is higher than the accuracy of the models trained following the vanilla approach. Notice that in five of the cases the accuracy of the anti-curriculum strategy is better than that of curriculum. The fact that anti-curriculum can achieve superior results is hardly surprising, since there are many documented cases in the literature where the anti-curriculum approach yields higher performance than the curriculum approach [16]. One possible explanation is that anti-curriculum forces the network to focus on the boundary cases and ambiguous examples early on and thus performs better when separating the classes. We can also observe that for three of the cases involving the Inception backbone network, the selected teacher is the Inception network as well. Thus these cases are examples of self-taught teacher transfer. Finally, we note that the best optimizer found for two of the DenseNet cases is SGD while in all other cases the Adam optimizer is selected.

V. DISCUSSION AND FUTURE DIRECTIONS

In this paper, we proposed a robust way of increasing the accuracy of early exits branches in multi-exit architectures and showed that it works across different datasets, backbone networks, branch locations and optimizers. We have also shown that even though curriculum learning provides the highest accuracy in most of the experiments, in some cases, anti-curriculum achieves the best performance, therefore it is better to test both of these approaches rather than relying exclusively on the former.

As future research directions, it would be worth investigating how our method performs with other training strategies for multi-exit architectures. It is unlikely that this method increases the accuracy for every weighting scheme of the end-to-end training strategy or for every layer in the layer-wise training strategy, however, it is an interesting problem to discover the conditions under which it provides a benefit. Moreover, the combination of our method with other methods for increasing the accuracy of early exits such as the distillation-based training introduced in Phuong et al. [11] can be explored. Indeed, curriculum learning in combination with an ensemble approach has been shown to improve knowledge distillation [28].

ACKNOWLEDGMENT

This work was partly funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, and by the Danish Council for Independent Research under Grant No. 9131-00119B. This publication reflects the authors views only. The European Commission and the Danish Council for Independent Research are not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," Nature 521, 436–444 (2015), doi: 10.1038/nature14539.
[2] H. Li, A. Kadav, I. Durdanovic, H. Samet and H. P. Graf, "Pruning filters for efficient ConvNets," in proceedings, ICLR 2017.
[3] M. Rastegari, V. Ordonez, J. Redmon and A. Farhadi "XNOR-Net: ImageNet classification using binary convolutional neural networks," in proceedings, ECCV 2016.
[4] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," arXiv:1705.08922 [cs.LG], Jun 2017.
[5] G. Hinton, O. Vinyals and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531 [stat.ML], Mar 2015.

[6] J. Chen and X. Ran, “Deep learning With edge computing: a review,” in Proceedings of the IEEE, vol. 107, no. 8, pp. 1655-1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.

[7] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan and X. Chen, “Convergence of edge computing and deep learning: a comprehensive survey,” in IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 869-904, Secondquarter 2020, doi: 10.1109/COMST.2020.2970550.

[8] J. Liu and Q. Zhang, “To improve service reliability for AI-powered time-critical services using imperfect transmission in MEC: an experimental study” in IEEE Internet of Things Journal, vol. 7, no. 10, pp. 9357-9371, Oct. 2020, doi: 10.1109/JIOT.2020.2984333.

[9] S. Scardapane, M. Scarpiniti, E. Baccarelli and A. Uncini, “Why should we add early exits to neural networks?,” Cogn Comput 12, 954–966 (2020), doi: 10.1007/s12559-020-09734-4.

[10] S. Teerapittayanon, B. McDanel and H. T. Kung, ”BranchyNet: Fast inference via early exiting from deep neural networks,” 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, 2016, pp. 2464-2469, doi: 10.1109/ICPR.2016.7900006.

[11] M. Phuong and C. Lampert, ”Distillation-Based training for multi-exit architectures,” 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 1355-1364, doi: 10.1109/ICCV.2019.00144.

[12] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in Proceedings of the 26th Annual International Conference on Machine Learning (ICML ’09), doi: 10.1145/1553374.1553380.

[13] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Technical report, Citeseer, 2009.

[14] E. Baccarelli, S. Scardapane, M. Scarpiniti, A. Momenzadeh and A. Uncini, “Optimized training and scalable implementation of conditional deep neural betworks with early exits for fog-supported IoT applications,” Information Sciences, Volume 521, 2020, Pages 107-143, ISSN 0020-0255, doi: 10.1016/j.ins.2020.02.041.

[15] E. L. Allgower and K. Georg, “Numerical continuation methods: an introduction,” volume 13. Springer Science & Business Media, 2012. doi: 10.1007/978-3-642-61257-2.

[16] X. Wang, Y. Chen and W. Zhu, “A comprehensive survey on curriculum learning,” arXiv:2010.13166 [cs.LG], Oct 2020.

[17] G. Hacohen and D. Weinshall, “On the power of curriculum learning in training deep networks,” in proceedings, ICML 2019.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, “Rethinking the Inception architecture for computer vision,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.

[19] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in proceedings, ICML 2019.

[20] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

[22] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, “Densely connected convolutional networks,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, “MobileNets: efficient convolutional neural networks for mobile vision applications,” arXiv:1704.04861 [cs.CV], Apr 2017.

[24] K. He, X. Zhang, S. Ren and J. Sun, ”Deep residual learning for image recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[25] T.-K. Hu, T. Chen, H. Wang and Z. Wang, “Triple wins: boosting accuracy, robustness and efficiency together by enabling input-adaptive inference,” in proceedings, ICLR 2020.

[26] A. LeNail, “NN-SVG: Publication-ready neural network architecture schematics,” Journal of Open Source Software, 4(33), 747, doi: 10.21105/joss.00747.

[27] D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization,” in proceedings, ICLR 2015.

[28] G. Panagiotatos, N. Passalis, A. Iosifidis, M. Gabbouj and A. Tefas, “Curriculum-based teacher ensemble for robust neural network distillation,” 2019 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2019, pp. 1-5, doi: 10.23919/EUSIPCO.2019.8903112.

7.11 Single-Layer Vision Transformers for More Accurate Early Exits with Less Overhead

The appended preprint follows.

Single-Layer Vision Transformers for More Accurate Early Exits with Less Overhead

Arian Bakhtiarnia[a,*], Qi Zhang[a], Alexandros Iosifidis[a]

[a] *DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Finlandsgade 22, Aarhus, 8200, Midtjylland, Denmark*

Abstract

Deploying deep learning models in time-critical applications with limited computational resources, for instance in edge computing systems and IoT networks, is a challenging task that often relies on dynamic inference methods such as early exiting. In this paper, we introduce a novel architecture for early exiting based on the vision transformer architecture, as well as a fine-tuning strategy that significantly increase the accuracy of early exit branches compared to conventional approaches while introducing less overhead. Through extensive experiments on image and audio classification as well as audiovisual crowd counting, we show that our method works for both classification and regression problems, and in both single- and multi-modal settings. Additionally, we introduce a novel method for integrating audio and visual modalities within early exits in audiovisual data analysis, that can lead to a more fine-grained dynamic inference.

Keywords: dynamic inference, early exiting, multi-exit architecture, vision transformer, multi-modal, deep learning

*Corresponding author

Email addresses: `arianbakh@ece.au.dk` (Arian Bakhtiarnia), `qz@ece.au.dk` (Qi Zhang), `ai@ece.au.dk` (Alexandros Iosifidis)

1. Introduction

Over the past decade, deep learning has shown tremendous success across various fields, such as computer vision and natural language processing [1]. However, deep learning models are by definition composed of many layers of interconnected neurons, even reaching billions of parameters, which makes them computationally expensive. This has sparked a great deal of research in order to make deep learning models more lightweight, for which many approaches have been proposed, for instance, *model compression* methods [2] such as *quantization* [3], *pruning* [4], *low-rank approximation* [5] and *knowledge distillation* [6].

More and more emerging internet of things (IoT) applications are integrating deep learning models, such as video surveillance, voice assistants, augmented reality and cooperative autonomous driving, which are often time-sensitive and require inputs to be processed within specific deadlines [7, 8]. The heavy computational burden of deep learning becomes problematic for these time-critical IoT applications, due to resource-constrained IoT devices. *Edge computing* is a promising computing paradigm for addressing this issue, in which the deep learning task is offloaded to edge servers in the proximity of IoT devices.

Since edge computing systems introduce computation offloading over a communication network and involve multiple nodes working collaboratively in order to complete the task in a timely manner, transmission time has to be taken into account in addition to the deep learning computation time. However, transmission time may vary greatly over time and across different channels. Consequently, deep learning models running on edge computing systems and IoT networks should be capable of *anytime prediction*, meaning they should be able to provide a valid response even if they are interrupted before traversing the entire neural network, although the model is expected to provide a better answer if it is allowed to run for longer time.

Dynamic inference approaches [9] modify the computation graph based on each input during the inference phase in order to fit the time constraints. A dynamic inference approach that particularly suits anytime prediction is *early exiting* [10], also referred to as *multi-exit architectures* or *auxiliary classifiers* in the literature. In multi-exit architectures, one or more early exit *branches* are placed after some of the intermediate hidden layers of the *backbone* network. The goal of each of these branches is to provide an early result similar to the final result of the neural network using only the features extracted up to that particular branch location. These early results are inevitably less accurate than the final result of the network. In order to achieve anytime prediction using early exiting, the latest early result can be used whenever the execution is interrupted, for instance, whenever a hard deadline is reached. Computation time can be further decreased by applying model compression techniques on the backbone of multi-exit architectures. Besides anytime prediction, early exiting can also be used in *budgeted batch classification* where a fixed amount of time is available in order to classify a set of input samples. In such a setting, the result of earlier branches can be used for "easier" samples whereas the result of later branches or the final result can be used for "harder" ones. The difficulty of each sample can be determined based on the confidence of the network about its output [11], although other approaches exist in the literature [10].

Early exit branches are expected to have a low overhead in terms of the extra computation they introduce, since a high overhead would defeat the purpose. Therefore, they often contain only a handful of layers. Ideally, we want the accuracy of the early results to be close to that of the final result, since a higher accuracy for early exit branches means that the overall reliability of the system increases. However, the low-overhead constraint makes it quite challenging to achieve a high accuracy since the early exit branches have significantly less trainable parameters compared to the rest of the network. Several approaches for increasing the accuracy of early exits such as knowledge distillation [12], curriculum learning [13] and architectures designed specifically for early exit branches [14] have been suggested. In this paper, we propose a novel architecture in order to obtain more accurate early exits for convolutional neural network (CNN) backbones.

A neural architecture called *vision transformer* (*ViT*) [15] has been recently introduced for image classification which is radically different from convolutional neural networks. The building blocks of Vision Transformer have been used for early exits placed on Vision Transformer backbones [14], however, using Transformer-based early exit branches on CNN backbones is not intuitive and requires additional steps and architectural modifications. We use a modified version of this architecture instead of the usual convolution and pooling layers in early exit branches and show that our method can significantly increase the accuracy of early exits compared to conventional architectures by fusing local and global receptive fields[1]. The contributions of this paper can be summarized as follows:

- We propose a novel architecture for early exit branches in multi-exit architectures based on vision transformers, called *single-layer vision transformer* (*SL-ViT*). We compare our method with conventional CNN-based early exit architectures across 27 scenarios involving different datasets, branch locations and backbone networks and show that our method is significantly more accurate in 26 of these scenarios, while having less overhead in terms of number of parameters and floating point operators (FLOPS). To the best of our knowledge the fusion of global and local scope in early exits has never been used in multi-exit architectures before.
- We show that our method is a general purpose approach that works across different modalities as well as multi-modal settings by investigating image classification, audio classification and audiovisual crowd counting scenarios. We also show that our method works for both classification and regression problems.
- We introduce a novel way of integrating audio and visual features in early exits using vision transformers. To the best of our knowledge, this is the first time early exits have been studied in multi-modal settings.
- We provide insight into why our method achieves better results compared to conventional CNN-based architectures by investigating the

[1]Our code will be available at `https://gitlab.au.dk/maleci/sl_vit`.

role of attention and receptive field.

- We introduce a fine-tuning strategy for SL-ViT called *copycat single-layer vision transformer* (*CC-SL-ViT*) which is based on the copycat strategy developed for CNNs [16] and show that this method can further increase the accuracy of SL-ViT early exits. To the best of our knowledge this is the first time the copycat strategy is used for vision transformers or early exits.

The rest of this paper is organized as follows: Section 2 provides an overview of the relevant literature; Section 3 describes our proposed method in detail; Section 4 explains the details of our experiments; Section 5 showcases the experiment results; and, finally, Section 6 briefly discusses the results and concludes the paper.

2. Related Work

This section provides the necessary prerequisites for understanding our method and experiments. We start by describing the particulars of multi-exit architectures. Subsequently, we provide the details of the vision transformer architecture, which is the foundation of the proposed method. Then, we briefly touch on how audio classification is normally carried out, which is included in several scenarios in our experiments. Finally, we explain another scenario investigated in our experiments, i.e. crowd counting, and how it can be approached in a multi-modal manner.

2.1. Multi-Exit Architectures

In order to describe multi-exit architectures, we use the same notation as Scardapane et al. [10] where a neural network is formulated as a function $f(X) = f_L(f_{L-1}(...f_1(X)))$. In this formulation L signifies the total number of layers in the network and f_i is the operator corresponding to layer i, which can be a convolutional layer, a fully-connected layer, a normalization layer, or any other differentiable operator. $h_i = f_i(h_{i-1})$ denotes the output of layer i, where h_0 is the input X. Finally, θ_i symbolizes the trainable parameters of layer i.

Equation (1) formulates the training process for the neural network which is achieved by tuning its parameters using an optimization algorithm on the landscape defined by a loss function. In this equation, the parameters of the neural network are denoted by $\theta = \bigcup_{i=1}^{L} \theta_i$, the training samples are signified by $\{(X_n, y_n)\}_{n=1}^{N}$, and $l(\cdot,\cdot)$ is the loss function.

$$f^* = \arg\min_{\theta} \sum_{n=1}^{N} l(y_n, f(X_n)) \tag{1}$$

Extending this notation to multi-exit architectures, $B \subseteq \{1, .., L\}$ signifies the set of selected branch locations after which early exit branches will be placed. $c_b(h_b) = y_b$ is the classifier or regressor representing the early exit branch at each branch location b, where y_b denotes the early result at that location. The schematic illustration of a multi-exit architecture is presented in Figure 1. However, since there are multiple outputs, and thus multiple loss signals in a multi-exit architecture, its training is not as straightforward.

Three different approaches for training multi-exit architectures exist in the literature [10, 17, 13]. In the first approach, called *end-to-end* training,

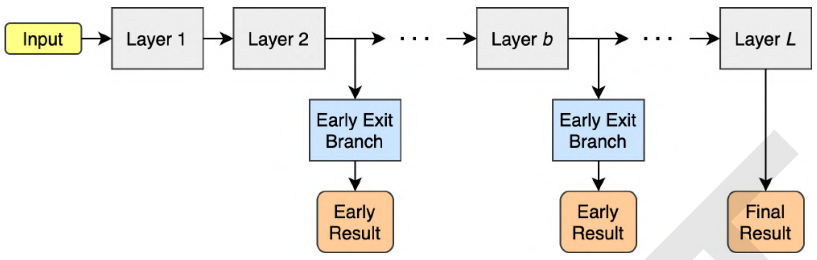


Figure 1: Schematic illustration of a multi-exit architecture with two early exits.

the loss signals of all exits are combined and backpropagated through the network at the same time. With end-to-end training, the contribution of each loss signal to the total loss is expressed with weight values, which are therefore hyper-parameters of the model.

The second approach, called *layer-wise* training, first trains the network up to and including the first exit branch. Subsequently, the part of the network that has been trained so far is frozen, meaning its parameters are not modified any further, and the remainder of the network up to and including the second exit branch is trained. This process continues until the entire network is trained. Note that with this approach, there is no guarantee that the accuracy of the final exit remains unchanged.

In the final approach, called *classifier-wise* training, the backbone network is completely frozen and each branch is trained independent of the rest of the network and other branches, meaning the parameters θ are not modified and only the parameters of the classifers/regressors $\{c_b\}, b \in B$ are trained. With this approach, no new hyper-parameters are introduced and the backbone remains unchanged. However, the early exit branches affect a lower number of trainable parameters compared to the other approaches.

In this paper, we choose to follow the classifier-wise training approach due to its practical importance. This is because with classifier-wise training, early exit branches can be easily added on top of existing backbone networks without the need for re-training and hyper-parameter optimization, which can be computationally expensive and time consuming. Furthermore, with end-to-end and layer-wise training strategies, the number of branches and their placement can lead to further trade-offs and affect the overall performance of the model. Since branches are independently trained in the classifier-wise strategy, any number of branches can exist and a branch can be placed at any location without affecting the performance of other branches or the backbone.

It is important to mention that branches placed later in the backbone network do not necessarily result in a higher accuracy compared to branches placed earlier. The usage of such branches would therefore not be sensible since earlier branches exist that require less computation and provide more accurate results. We hereby use the term *impractical* to refer to such branches.

As previously mentioned, there are several methods that try to improve the accuracy of early exits. The method in [12] uses the combination of the distillation loss from the final exit and the loss signal from ground truth labels to train more accurate early exits using in the end-to-end training setting. The method in [18] expands on this idea by adding a third loss signal based on the difference between features of the latest early exit with earlier exits. The method in [19] proposes a technique called *gradient equilibrium* to combat the problem of gradient imbalance that surfaces when using the end-to-end strategy, which is when the variance of the gradients becomes very large when loss signals from multiple exits are combined, leading to unstable training. Moreover, this paper introduces forward and backward knowledge transfer that aims to encourage collaboration among different exits. The method in [20] improves the accuracy of later exits by reusing predictions from earlier exits. The method in [21] circumvents the problem of impractical branches by adaptively selecting the exit location based on time budget and the specific input. The method in [22] simplifies the design of multi-exit architectures by removing the hyper-parameters of the end-to-end training strategy that specify the contribution of each loss signal.

Besides efficient inference, early exits can prove useful in several other applications, for instance, the method in [23] allows for parallel training of the segments of the DNN that exist between early exits, by training each segment based on the loss signal of the next segment obtained in the previous training stage. Moreover, early exits can be added to the network during the training in order to increase the accuracy of the backbone network and discarded after the training phase, for instance, the widely used Inception model [24] was trained in this way.

Besides early exiting, several other approaches exist for dynamic inference, for instance, layer skipping [25, 26, 27, 28] where the execution of some of the layers of the DNN are skipped, and channel skipping [29] where less impactful channels of convolutional neural networks are ignored and their computation is skipped during the inference phase. However, unlike early exits, these approaches cannot provide an output if the execution is interrupted due to a strict deadline, as these methods need to perform the computations until the very last layer.

2.2. Vision Transformer

The transformer architecture was first introduced by Vaswani et al. [30] for natural language processing, and it has recently been adapted for solving computer vision problems by Dosovitskiy et al. [15]. Vision transformer was originally developed for the problem of image classification, however, variations of vision transformer have since been applied to many computer vision problems, such as object detection, depth estimation, semantic segmentation, image generation and action recognition, as well as multi-modal data analysis tasks such as text-to-image synthesis and visual question answering [31, 32, 33].

In order to describe the vision transformer architecture, we first explain the *self-attention* layer. The input of this layer is in the form of a sequence $X = (x_1, \ldots, x_n)$ where $X \in \mathbb{R}^{n \times d}$ and d is the embedding dimension to represent each entity. Its output is in the form of $Z = (z_1, \ldots, z_n)$ where $Z \in \mathbb{R}^{n \times d_v}$. The goal of self-attention is to capture the interaction between the entities in the sequence. For this purpose, each vector x_i in the sequence is transformed into three separate vectors: the *query* vector $q_i \in \mathbb{R}^{d_q}$, the *key* vector $k_i \in \mathbb{R}^{d_k}$ and the *value* vector $v_i \in \mathbb{R}^{d_v}$, where $d_q = d_k$. To construct the output vector z_i that corresponds to the input x_i, for each vector x_j in X (including x_i itself), the scalar a_{ij} is calculated by the inner product of q_i and k_j. Output vector z_i is then calculated by summing the value vectors $v_1, \ldots, v_n$ weighted by their corresponding scalars, that is, $z_i = \sum_{j=1}^{n} a_{ij} v_j$. The scalar a_{ij} basically specifies how much attention the i-th entity should pay to the j-th entity, since a_{ij} determines the contribution of v_j to the combined output z_i. In practice, the scalars are normalized by $\sqrt{d_k}$ and converted into probabilities using the softmax function.

If the key, query and value vectors are packed into matrices $Q = XW^Q$, $K = XW^K$ and $V = XW^V$, where W^Q, W^K and W^V are learnable weight matrices, the above operation can be rephrased as follows:

$$Z = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

In order to enable the model to capture more than one type of relationship between the entities in the sequence, self-attention is extended to *multi-head attention* by concatenating the output of h different self-attention blocks $Z_1, \ldots, Z_h$ each with its own set of learnable weight matrices, into a single matrix $Z' = [Z_0, \ldots, Z_h] \in \mathbb{R}^{n \times h.d_v}$, which is then projected using a weight matrix $W' \in \mathbb{R}^{h.d_v \times d}$.

A *transformer encoder* is constructed by passing the input sequence into a normalization layer, a multi-head attention layer, a second normalization layer and a multi-layer perceptron (MLP), respectively. Two residual connections are added, one by adding the input sequence to the output of the multi-head attention, and the other by adding the output of the multi-head attention to the output of the MLP.

Putting it all together, a vision transformer is created by first splitting the input image into patches. Subsequently, the sequence of patches is projected into a sequence of vectors and a positional embedding is added to the corresponding vector of each patch. An additional learnable embedding called *classification token* is added to the beginning of the sequence. The sequence then passes through L transformer encoders. Finally, the first vector in the output of the last transformer encoder, which corresponds to the classification token, is passed to a MLP which outputs the final classification result. The architecture of vision transformer is depicted in Figure 2.

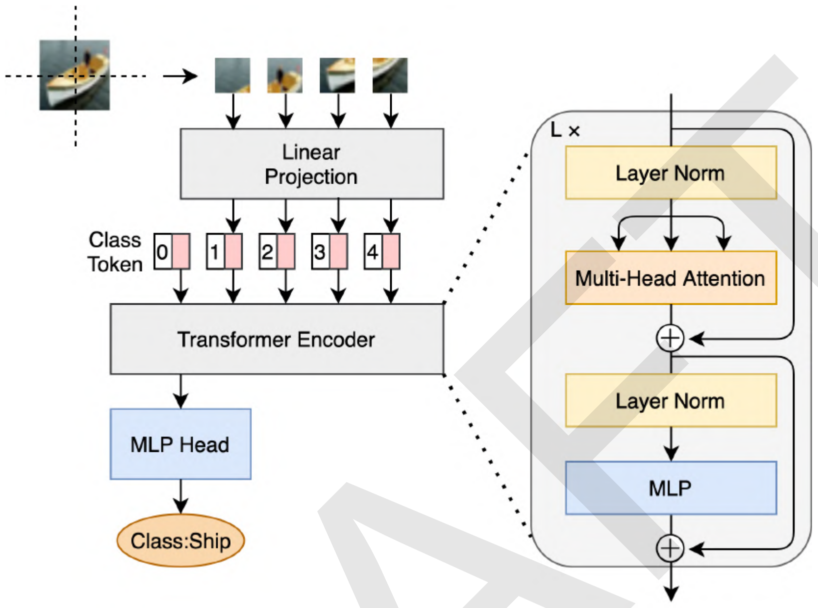


Figure 2: The vision transformer (ViT) architecture for image classification.

ViT-EE is a method which uses transformer encoders for early exits placed on ViT backbones [14]. ViT-EE uses the exact same layer as the ViT backbone. Using the building blocks of the backbone network for early exit branches is simple and intuitive, and it is the reason why so far, mostly convolutional layers have been used for early exiting CNN backbones. However, as we show in this work, carefully designing the architecture of early exit branches can lead to significant improvements. Using Transformer-based early exit branches on CNN backbones is not intuitive, and requires additional steps such as converting tensors to patches, dealing with the classification token and fine-tuning the architecture parameters including patch size, attention heads, embedding representation, the size and number of layers for MLP, and dropout. Moreover, we show that removing the last residual connection in the transformer encoder can improve the performance in some cases.

Furthermore, ViT backbones have a global receptive field in every layer, this means that ViT-EE is not necessarily ideal for early exits at all layers, as it adds too much overhead without providing improvements in terms of receptive field. On the other hand, CNN backbones have a limited receptive field particularly in earlier layers, therefore fusing this receptive field with a global one leads to improvements.

2.3. Audio Classification

Similar to image classification, audio classification is the problem of categorizing a given audio waveform into one of several predetermined classes. For instance, the given audio waveform could be a musical recording, and the goal could be to specify which genre of music it belongs to. To represent the input features, *spectrograms* obtained by applying short-time Fourier transform (STFT) and *Mel spectrograms* are commonly used [34], although raw audio waveforms can been used as well [35]. Mel spectrograms are spectrograms that are constructed using the *Mel scale* which is a nonlinear transformation of the frequency scale designed based on domain knowledge about the human auditory system. Various deep learning models for audio classification exist in the literature, including models that are commonly used for image classification, namely ResNet [36], DenseNet [37] and Inception [38], which have been shown to be quite effective for audio classification as well [39]. Conveniently, the same three networks have previously been used as backbone networks when investigating early exiting for image classification

[13]. Therefore we use these backbone networks for both image and audio classification in our experiments.

2.4. Audiovisual Crowd Counting

Crowd counting refers to the problem of identifying the total number of people present in a given image. Crowd counting has many applications such as safety monitoring, disaster management, design of public spaces, intelligence gathering and analysis, creation of virtual environments and forensic search [40]. With many of these applications, it is vital for the model to perform in near real-time. However, the input images in these scenarios often have high resolutions, such as HD or Full HD. Moreover, many of the available methods contain an immense number of parameters [41]. This means that crowd counting models are often very computationally expensive, therefore, dynamic inference methods such as early exiting and other lightweight deep learning methods become essential in real world applications.

Although the main objective of this task is to obtain a single count from an image, many methods treat this problem as dense prediction where the output is a *density map* depicting the density of the crowd across the input image, and the total count is calculated by the sum of all values in the density map. Therefore, in most crowd counting datasets, such as Shanghai Tech [42] and World Expo '10 [43], the locations of the heads of individuals in the image are annotated and provided as targets. A ground truth density map can then be obtained from these *head annotations* using Gaussian kernels or more complicated and specialized methods [41]. Figure 3 shows an image from the Shanghai Tech dataset and the ground truth density map that was generated from the provided head annotations using the method presented in

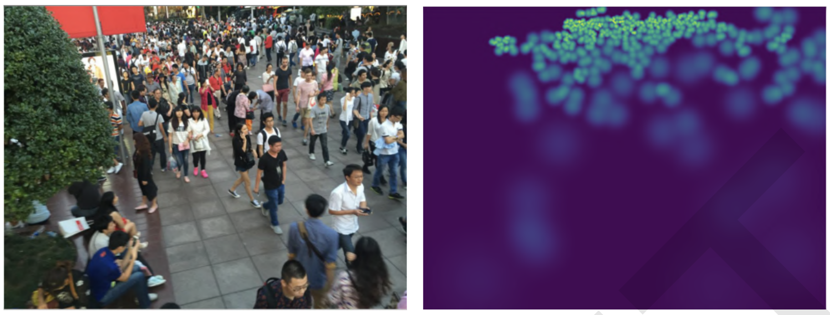


Figure 3: An example image from the Shanghai Tech dataset and its corresponding ground truth density map.

Zhang et al [42]. In crowd counting, *Mean Absolute Error* (*MAE*) is usually used as a measure of accuracy whereas *Mean Squared Error* (*MSE*) is used as a measure of robustness [44].

Many crowd counting methods exist in the literature [41], however, most of these methods are applied in a single-modal fashion where the input is an image or a video frame. In contrast, AudioCSRNet [45], a multi-modal extension of the widely-used CSRNet model for crowd counting [46], takes as input the ambient audio of a scene in addition to its image. The authors show that the ambient audio improves the result in situations where the image quality is not ideal, for instance, low image resolution, presence of noise, occlusion and low illumination.

In CSRNet, the features extracted from the input image by the first 10 layers of a VGG-16 [47] network pre-trained on the ImageNet dataset [48] are passed through 6 dilated convolution layers and a 1×1 convolution layer in order to obtain the density map. AudioCSRNet extends this architecture by converting each of the dilated convolution layers into a fusion block. The architecture of AudioCSRNet is depicted in Figure 4. First, a Mel spectrogram is obtained from the raw audio waveform. Subsequently, in each fusion block, the features extracted from the input Mel spectrogram by the first 6 layers of a VGGish [49] network pre-trained on the AudioSet dataset [49] are projected to two vectors called γ and β which represent the multiplicative and additive aspects of the audio features. The γ and β vectors are then tiled in order to match the size of the visual features. Finally, the output of the dilated convolution is element-wise multiplied by γ and added to β.

The fusion operation can be summarized as

$$v_{l+1} = \mathcal{F}_l(\gamma_l \odot D_l(v_l) + \beta_l), \tag{3}$$

where $v_l \in \mathbb{R}^{C_l \times W_l \times H_l}$ is the output of the l-th fusion block, F_l denotes an activation function, γ_l and β_l are the tiled vectors and D_l represents the l-th dilated convolution.

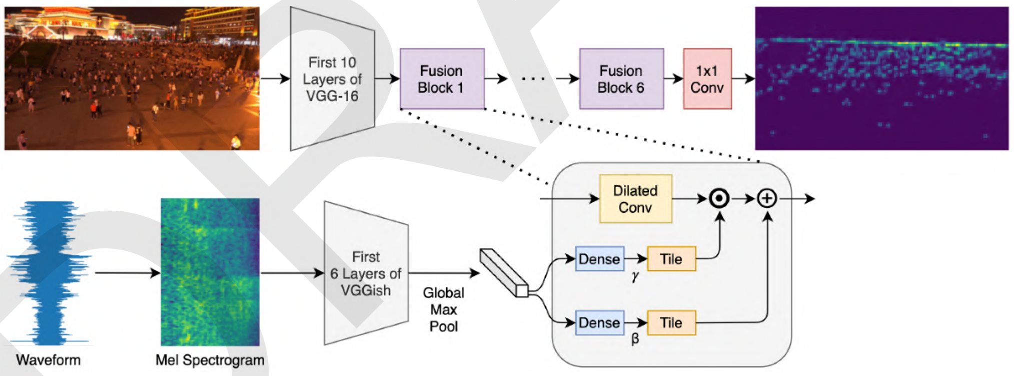


Figure 4: Architecture of AudioCSRNet.

In practice, a batch normalization layer [50] is added immediately after each dilated convolution. Furthermore, the height and width of the intermediate features remain unchanged by using padding in the convolution operations, meaning $H_l = H_{l+1}$ and $W_l = W_{l+1}$. Additionally, since the first 10 layers of VGG-16 decrease both height and width by a factor of 8 via several pooling operations, the final result of the network needs to be upsampled by a factor of 8 in order to match the resolution of the input image. It is important to preserve the total sum of the density map during this upsampling operation, since it represents the total count.

3. Single-Layer Vision Transformers for Early Exits

We assume a pre-trained and high performing backbone network is already available. Due to time constraints arising from the particular application, it is desirable that the network provides a result within the specific deadline rather than not providing a result at all, even though this result may be less accurate than it would be if time constraints did not exist. Therefore, the backbone needs to be augmented with early exit branches to allow for dynamic inference and anytime prediction. As previously mentioned, we use the classifier-wise approach for training the early exit branches since it results in "plug-and-play" branches that can easily be added to the backbone network without any re-training or hyper-parameter tuning.

3.1. SL-ViT

Typically, the architecture of early exit branches starts with one or more convolution layers, although some may have no convolutions at all. Afterwards, they may have a pooling layer, which may be global pooling, and one MLP [51, 11]. Here, as a baseline, we choose to utilize the architecture depicted in Figure 5 with one 3×3 convolution, followed by a 2×2 max pooling layer and finally a MLP. The size of the max pooling layer is increased to 4×4 for crowd counting since the input images have a very high resolution.

Additionally, we use dropout [52] inside the MLP to avoid overfitting. We use a single convolution since early exits with two or more convolution layers have a high overhead and may even lead to lower accuracy [11]. Early exits without convolutions are sometimes used very late in the network, however, since they are straightforward and leave no room for modifications, we do not apply our method for such cases. The resulting architecture is a common setup within the literature, and is effectively the same architecture used for earlier exits by Hu et al. [51].

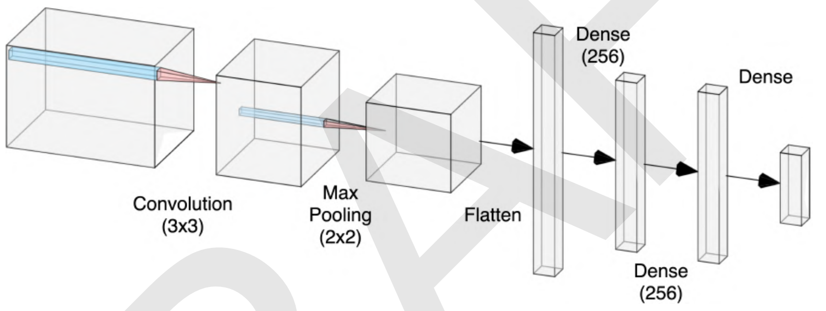


Figure 5: Architecture of CNN early exit branches. Size of the flattened feature vector depends on the dimensions of the features at the specific branch location. For branches placed on the AudioCSRNet backbone, max pooling size is increased to 4x4 since the input images have a high resolution. Figure created using the NN-SVG tool [53].

Our method called *single-layer vision transformer* or *SL-ViT* for short, is an alternative architecture for early exit branches that can achieve a higher accuracy compared to the aforementioned baseline, while having less overhead in terms of the number of parameters as well as floating point operations per second (FLOPS). Our proposed architecture is based on the vision transformer architecture introduced in section 2.2, where instead of the input image, we split the intermediate features at the branch location into patches (sub-tensors) and pass them to a vision transformer.

The choice of vision transformer architecture is primarily due to its global receptive field. Receptive field is crucial in many deep learning problems, including ones studied in this work. The receptive field of state-of-the-art CNNs developed for image classification has steadily increased over time and is correlated with increased classification accuracy [54]. Additionally, in audio classification using spectrograms, each location relates to a different frequency band in a different window of time. It is reasonable to assume that processing combinations of frequencies and windows that are not necessarily adjacent could be of importance. Moreover, many crowd counting methods have made use of global information through visual attention mechanisms and dilated convolutions [41]. Since the receptive field is particularly limited in early layers of CNN backbones, choosing an architecture for early exit branches with a global receptive field could be beneficial.

Many other designs strive to increase the receptive field in their building blocks, for instance, the *pyramid pooling module (PPM)* in PSPNet [55] or *atrous spatial pyramid pooling (ASPP)* in DeepLab [56]. However, they all fall short in comparison with the global receptive field of transformers. PPM increases the receptive field through aggregating different levels of pooling, which means far locations have only access to coarse representations of each other, and ASPP has holes in its receptive field.

It is important to mention that the local receptive field of convolutional layers is not fundamentally bad. On the contrary, it plays a key role in representation learning and extracting local information, especially in the early layers of the network where the receptive field of the convolutional filters is small. Filters in successive convolutional layers have increasingly larger receptive fields, therefore, final layers in a CNN architecture have filters of large enough receptive fields that can effectively aggregate information from the entire input image to provide a proper response. However, this process of cascading local receptive fields of increasing size requires the number of layers in the CNN to be large, or at least all the layers in the network to be traversed in order to provide the network's response. When an early exit is added at an early layer, this chain of increasingly larger receptive fields is broken, and an early exit that has a local receptive field may not be able to effectively aggregate all required information in the image to provide a suitable response. This situation is the motivation behind the proposed branch architecture, which fuses the local receptive field of the layer in the network where the early exit branch is attached, with the global receptive field of the early exit, in order to effectively aggregate information from the entire input and provide a more accurate response. Indeed, the original Vision Transformer paper [15] attributes the success of their model to the combination of local and global receptive fields and shows that even in very early layers, this ability to integrate information globally is indeed used by the model.

There are some crucial differences between the original vision transformer and the architecture in our method. First, in order to introduce a low overhead for early exit branches, we only use a single transformer encoder layer instead of the original 12 to 36 layers, meaning that $L = 1$ in our case. Secondly, we do not utilize a separate classification token and instead pass the entire output of the transformer encoder layer to the MLP head. This is possible because the width and height of tensors are generally reduced throughout CNN backbones by pooling operations, and thus the number of patches in our architecture is lower than that of the original vision transformer. In addition to the number of patches, the size of the embedding dimension (d) is also reduced in our proposed architecture, introducing far less parameters when passing the entire output of the last transformer encoder layer to the MLP head, even with high-resolution inputs such as in our crowd counting experiments. Variations of our architecture have 5×5, 7×7 or 16×9 patches and embedding dimensions of 32 or 36, whereas different versions of the original vision transformer have 14×14 or 16×16 patches and embedding dimensions of 768, 1024 or 1280. We empirically found that using the entire transformer encoder output instead of just one classification token can increase the accuracy, perhaps because in a single-layer version, there are not enough layers for the classification token to learn to properly summarize other patches. Our proposed architecture is shown in Figure 6. It is also important to note that the MLP head used in our architecture is exactly the same as the MLP in the CNN early exit architecture.

Our model has several hyper-parameters, namely the size of each patch, the embedding dimension d and the number of attention heads h in multi-head attention. The patch size creates a trade-off where smaller patches result in a more fine-grained attention mechanism while increasing the total number of parameters in a bi-quadratic fashion. Therefore, similar to the original vision transformer, we choose the size of the patch to be close to the square root of the height and width of the input features. We also make sure that the size of the patch can divide the size of the input features to avoid

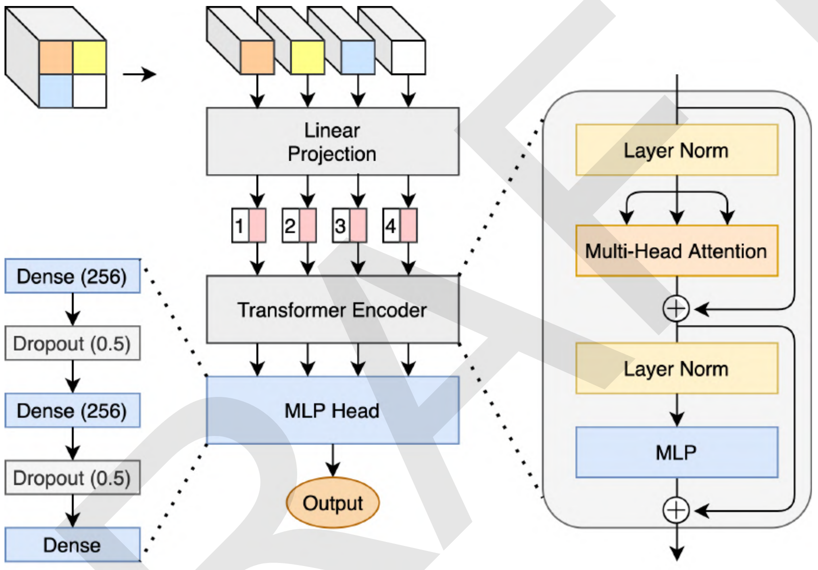


Figure 6: Architecture of SL-ViT early exit branches. Unlike typical vision transformers, only a single transformer encoder layer is used, extra learnable classification token is not added to the sequence and the entire output of the transformer encoder is passed on to the MLP head. The MLP head is the same as CNN early exit branches.

padding, for instance, a patch size of 4×4 for input features of size 28×28. We perform a grid search to find the values of d and h that result in the highest accuracy, while keeping the total number of parameters less than or equal to that of the CNN early exit counterpart.

At a first glance, it might seem like the SL-ViT architecture introduces more hyper-parameters than the conventional CNN architecture, however, the CNN architecture includes many design choices as well, such as the number of filters, filter size, padding, dilation, stride, pooling type and pooling size. The design choices for CNN architectures might seem simpler since they have been studied more extensively compared to vision transformers which were introduced more recently.

3.2. Audiovisual SL-ViT

With audiovisual backbones such as the AudioCSRNet model for audiovisual crowd counting, described in section 2.4, it is desirable to have audiovisual early exits that use both visual and audio features in order to achieve a higher accuracy. The simplest way to have such branches is to add the branches after the blocks where the fusion of visual and audio features take place. However, with our proposed SL-ViT architecture, it is also possible to include audio features as one or more patches alongside other patches, and directly fuse the features in the early exit. The advantage of this approach is that since in vision transformers, any of the patches can pay attention to any other patch, the visual features can be fused with the audio features without being directly impacted and modified. In contrast, since convolutional filters only take the immediate vicinity into account, the audio features must be present in every location. One option is to concatenate the visual features and the tiled audio features along the depth. However, that would greatly increase the amount of computation for each fusion operation, therefore intrusive operations such as element-wise multiplication and addition are used instead.

3.3. Copycat SL-ViT

Finally, we introduce a fine-tuning strategy for SL-ViT branches that can further increase their accuracy. Correia-Silva et al. [16] developed a method called *copycat CNN* where they create a "fake" dataset by taking images from another domain, giving them as input to a network trained on the target domain, and recording the output of the network as labels for these images. For instance, images from the ImageNet dataset [48] can be given to a network trained on the CIFAR-10 dataset [57], where the image of a camel may be labelled as a "dog" since there are no labels for "camel" in CIFAR-10. This fake dataset is then combined with a dataset for the target domain and used to train a new network. We use this strategy to fine-tune an already trained SL-ViT branch and obtain a *copycat single-layer vision transformer* (*CC-SL-ViT*). Note that the ratio of the fake data mixed with the available dataset is a hyper-parameter of this fine-tuning strategy.

4. Experimental Setup

In this section, we provide the details of our experiments. We begin by giving a short summary of the datasets as well as the training details for the backbone networks. We then lay out the details of the branch architectures, their training procedure and their placement on the backbone networks, and finally explain how the copycat strategy was used to fine-tune the branches.

A total of 27 different scenarios were tested in our experiments. For both image and audio classification, two datasets, three backbone networks and two different branch locations on each backbone were tested. In addition, three different branch locations for the audiovisual crowd counting backbone network were covered. All experiments were repeated 5 times and the average accuracy as well as the standard deviation were recorded. 4 $\times$ Nvidia 2080Ti GPUs were used for the training of our models.

4.1. Datasets

4.1.1. CIFAR-10 and CIFAR-100

These are widely-used datasets for image classification [57]. Both datasets consist of 60,000 color images of size 32 $\times$ 32 pixels and their corresponding class labels. The images in CIFAR-10 and CIFAR-100 are categorized into 10 and 100 different classes, respectively. We use 40,000 examples for training, 10,000 for validation and another 10,000 for testing. Since our backbone networks are pre-trained on ImageNet which consists of 224 $\times$ 224 pixel images, we resize each image to these dimensions before passing them into the network.

4.1.2. Speech Commands (SC)

A well-known audio dataset of spoken words [58]. It consists of 100,503 1-second audio clips with a sampling rate of 16kHz, each labelled as one of 12 classes: 10 different spoken words such as “Yes”, “No”, “Down” and “Stop” as well as one class for background noise and another for unknown words. We use 85,511 examples for training, 10,102 for validation and 4,890 for testing. We convert the raw audio waveforms into spectrograms using short-time Fourier transform (STFT) with a window size of 255 samples and step size of 128 samples, and resize the resulting spectrograms to 224×224 before passing them into the network.

4.1.3. GTZAN

It is the most widely-used dataset for music genre recognition [59]. The original dataset consists of 10 genres such as "Pop" and "Rock" and 100 30-second audio clips per genre with a sampling rate of 22,050Hz. We follow the common approach to split each audio clip into 10 separate 3-second clips in order to increase the size of the dataset to 10,000. We use 8,000 examples for training, 1,000 for validation and another 1,000 for testing. Following the approach of Palanisamy et al. [39] where different spectrograms with different parameters are placed in each channel of the input image, we use one spectrogram obtained from STFT with window size of 512 samples and step size of 256 samples as well as two Mel spectrograms with 128 bins and window sizes of 100 and 50 milliseconds, and step sizes of 50 and 25 milliseconds, respectively.

4.1.4. DISCO

An audiovisual dataset for crowd counting which contains 1,935 images of Full HD resolution (1920×1080) [45]. For each image, a corresponding 1-second audio clip of ambient sounds with a sampling rate of 48kHz, starting 0.5 seconds before the image was taken and ending 0.5 seconds afterwards, exists as well. The labels are provided in the form of head annotations in the image. At the time of this writing, DISCO is the only publicly available dataset for audiovisual crowd counting. We use 1435 examples for training, 200 for validation and 300 for testing. The input image is resized to 1024×576 pixels to reduce memory and computation requirements. Similar to Hershey et al. [49], the input audio waveform is transformed into a Mel spectrogram with 64 bins, window size of 25 milliseconds and step size of 10 milliseconds. Following Hu et al. [45] the ground truth density maps are obtained by convolving the head annotations with a 15×15 Gaussian kernel $\mathcal{K} \sim \mathcal{N}(0, 4.0)$.

4.2. Backbone networks

Transfer learning is used to train the ResNet152, DenseNet201 and InceptionV3 backbone networks for both image and audio classification. The backbone networks are all pre-trained on the ImageNet dataset and the top layer is replaced. We found that instead of adding just one dense layer at the top, as is common in transfer learning, using two dense layers and a dropout layer in between leads to a higher accuracy in our case. The resulting network is then trained using the Adam optimizer [60] with a learning rate of 10^{-4} and categorical cross-entropy loss function. The learning rate is reduced by a factor of 0.6 on plateau with a tolerance of 2 epochs, and an early stopping mechanism with a tolerance of 5 epochs is used.

The audiovisual crowd counting backbone is trained in two stages. We first train a network with the AudioCSRNet architecture described in Section 2.4 for 100 epochs. L_2 norm is used as loss function and AdamW [61] with a learning rate of 10^{-5} and weight decay of 10^{-4} is used as optimizer, where the learning rate is multiplied by a factor of 0.99 each epoch. This is the same training procedure used in the original paper [45]. Subsequently, in order to convert the problem from dense prediction to regression, a dense layer with an output size of one is added after the last layer of the trained AudioCSRNet. This layer is initialized as a sum, meaning the initial weights are all equal to one and no bias is used. Then the entire network is re-trained for another 100 epochs using MAE as loss function instead of the previous L_2 loss, a learning rate of 10^{-6} and weight decay of 10^{-5}. The learning rate is similarly multiplied by a factor of 0.99 every epoch. The resulting model achieves a MAE of 13.63 which is even lower than the MAE of 14.27 reported in the original paper. However, the output of the network is just a single number representing the total count instead of a density map. The final accuracy of all trained backbones can be seen in Table 1.

When training the backbone networks, in order to fit the limitations of our available computational resources, the batch sizes are adjusted and some layers of the backbone networks are frozen. All backbone networks were trained with a batch size of 32 except AudioCSRNet which has a batch size of 4 as well as InceptionV3 when trained on CIFAR-10 and CIFAR-100 which has a batch size of 64. All layers of the backbone networks were trained, except in the case of ResNet152 and DenseNet201 when trained on CIFAR-10 and CIFAR-100 where only the batch normalization layers were trained. We found that training only the batch normalization layers is sufficient to achieve a high-performing backbone network in these cases [62].

4.3. Branches

All branches were trained from scratch using the He initialization method [63] and the Adam optimizer with a learning rate of 10^{-4} where the learning rate is reduced by a factor of 0.6 on plateau with a tolerance of 2 epochs, and an early stopping mechanism with a tolerance of 5 epochs is utilized.

Table 1: Performance of backbone networks on each dataset

Backbone	CIFAR-10 Acc.	CIFAR-100 Acc.	SC Acc.	GTZAN Acc.	DISCO MAE
ResNet152	95.36%	82.25%	95.85%	91.29%	-
DenseNet201	96.48%	82.53%	96.36%	92.09%	-
InceptionV3	96.56%	83.80%	94.93%	87.79%	-
AudioCSRNet	-	-	-	-	13.63

The branches on classification backbones use a categorical cross-entropy loss function whereas the branches on the audiovisual crowd counting backbone use mean absolute error loss. The training batch size for branches were 64 in scenarios involving CIFAR-10, CIFAR-100 and Speech Commands, 32 in scenarios involving GTZAN and 4 in scenarios involving DISCO.

Table 2 shows the location of the branches placed on each backbone network. For the AudioCSRNet backbone network, branch V1 uses only the output of the VGG-16 layers, therefore, it only has access to the visual features. Branch AV1 uses the outputs of both VGG-16 and VGGish, therefore it has access to both audio and visual features. In this branch location, the fusion of audio and visual features is performed as described in Section 3 for the SL-ViT architecture, and similar to the fusion blocks in AudioCSRNet for the CNN architecture, however, without dilation. Finally, branch AV2 is placed after the first fusion block in AudioCSRNet, therefore audio and visual features have already been fused and thus fusion operation is not required within the branches. Adding branches after the second fusion block or later would not be reasonable since more than 85% of the computation of the backbone is carried out before that point, and thus the acceleration resulting from early exits would be negligible.

Table 2: Placement of branches for each backbone betwork

Backbone	BN*	Branch Placed After
DenseNet201	1	Transition Layer 1
	2	Transition Layer 2
ResNet152	1	12th Convolution
	2	36th Convolution
InceptionV3	1	First Filter Concat
	2	Second Filter Concat
AudioCSRNet	V1	Last Layer of VGG
	AV1	Last Layers of VGG and VGGish
	AV2	First Fusion Block

*Branch Number

4.4. SL-ViT and CC-SL-ViT Parameters

Table 3 summarizes the hyper-parameters used for the SL-ViT branches in each scenario. “Patch Size” shows the width and height of each image patch, “Patches” denotes the resulting number of patches across width and height of the input image, d is the size of embedding dimension and h is the number of heads in multi-head attention.

For copycat SL-ViT, images from the Tiny ImageNet dataset, which are the images from ImageNet down-sampled to 32×32, were given to the InceptionV3 backbone trained on CIFAR-10, and the outputs were used to create the fake dataset. Then the fake dataset was mixed with CIFAR-10 with a 2-to-1 ratio and used for re-training.

Table 3: Hyper-parameters of SL-ViT for different backbone networks and branches

Backbone	Dataset	BN*	Patch Size	Patches	d	h
DenseNet201	all	all	4x4	7x7	32	12
ResNet152	SC	2	4x4	7x7	32	24
	GTZAN	2	4x4	7x7	32	24
	Other		4x4	7x7	32	12
InceptionV3	CIFAR-100	all	5x5	5x5	36	8
	Other		5x5	5x5	32	12
AudioCSRNet	DISCO	all	8x8	16x9	32	12

*Branch Number

5. Results

The results of our experiments are presented in Tables 4 to 8. In these Tables, the final accuracy, the total FLOPS of the model up to and including the branch and the number of parameters of just the early exit branch are compared between the CNN architecture and the SL-ViT architecture. Higher accuracies, lower errors, lower number of parameters and lower total FLOPS are highlighted in these tables. Furthermore, the acceleration caused by SL-ViT early exits, defined as the total FLOPS of the backbone network divided by the total FLOPS of the model up to and including the SL-ViT branch, is also provided.

Several observations can be made about these results. First, in all scenarios except one, SL-ViT early exits achieve a significantly higher accuracy. Even in the one exceptional scenario, namely branch 2 of ResNet152 in Table 6, the accuracy of SL-ViT is very close to its CNN counterpart. Secondly, while in some cases SL-ViT branches have an equal number of parameters compared to CNN branches, in all scenarios, the total FLOPS of SL-ViT branches is lower, therefore SL-ViT branches are always more lightweight. Thirdly, in one scenario, namely branch 2 of ResNet152 in Table 7, removing the last residual connection in the SL-ViT architecture significantly improved the accuracy of the SL-ViT branch. Finally, in the AV2 branch location in Table 8, both CNN and SL-ViT are impractical branches since earlier branches with higher accuracies exist. This is perhaps due to the intrusive fusion operation in the first fusion block which might initially make the intermediate features more obscure. Nonetheless, even in this case, SL-ViT is more accurate.

Table 4: Comparison of different early exit architectures on the CIFAR-10 dataset

Backbone	Branch	Accuracy		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
ResNet152	1	66.74 ± 0.57%	**70.79 ± 0.72%**	0.78M	**0.59M**	1.66B	**1.64B**	13.77
	2	79.31 ± 0.81%	**81.18 ± 0.52%**	0.83M	**0.79M**	5.33B	**5.26B**	4.29
DenseNet201	1	71.27 ± 0.36%	**76.38 ± 0.33%**	0.78M	**0.59M**	2.55B	**2.53B**	3.39
	2	80.64 ± 0.29%	**83.53 ± 0.37%**	0.80M	**0.66M**	4.21B	**4.17B**	2.06
InceptionV3	1	77.27 ± 0.58%	**79.99 ± 0.20%**	0.61M	**0.56M**	2.17B	**2.14B**	2.65
	2	79.55 ± 0.24%	**81.72 ± 0.53%**	0.61M	**0.56M**	2.53B	**2.49B**	2.28

Table 9 shows the result of applying the copycat fine-tuning strategy to SL-ViT branches for the CIFAR-10 dataset. Observe than in all cases, the accuracy is significantly increased compared to SL-ViT, which itself was more accurate than CNN based on Table 4. We also tested this strategy for the CIFAR-100 dataset with 10-to-1, 2-to-1 and 1-to-1 ratios of fake and real data, however, neither improved the overall accuracy. Perhaps another

Table 5: Comparison of different early exit architectures on the CIFAR-100 dataset

Backbone	Branch	Accuracy		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
ResNet152	1	34.93 ± 0.52%	**38.59 ± 1.40%**	0.80M	**0.61M**	1.66B	**1.64B**	13.77
	2	47.39 ± 0.65%	**53.93 ± 0.68%**	0.86M	**0.81M**	5.33B	**5.26B**	4.29
DenseNet201	1	33.91 ± 1.00%	**42.50 ± 0.69%**	0.80M	**0.61M**	2.55B	**2.53B**	3.39
	2	47.22 ± 0.45%	**50.76 ± 1.01%**	0.82M	**0.68M**	4.21B	**4.17B**	2.06
InceptionV3	1	43.18 ± 0.69%	**46.86 ± 0.57%**	0.63M	0.63M	2.17B	**2.14B**	2.66
	2	44.87 ± 0.83%	**49.07 ± 0.55%**	0.63M	0.63M	2.53B	**2.50B**	2.28

Table 6: Comparison of different early exit architectures on the Speech Commands dataset

Backbone	Branch	Accuracy		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
ResNet152	1	75.80 ± 0.73%	**84.05 ± 0.31%**	0.78M	**0.59M**	1.66B	**1.64B**	13.77
	2	**89.78 ± 0.24%**	89.63 ± 0.52%	0.84M	0.84M	5.33B	**5.26B**	4.29
DenseNet201	1	72.78 ± 0.64%	**87.94 ± 0.85%**	0.78M	**0.59M**	2.55B	**2.53B**	3.39
	2	86.56 ± 0.61%	**90.93 ± 0.52%**	0.80M	**0.66M**	4.21B	**4.17B**	2.06
InceptionV3	1	84.64 ± 0.88%	**87.62 ± 0.65%**	0.61M	**0.56M**	2.17B	**2.14B**	2.65
	2	87.08 ± 1.11%	**88.33 ± 0.92%**	0.61M	**0.56M**	2.53B	**2.49B**	2.28

Table 7: Comparison of different early exit architectures on the GTZAN dataset

Backbone	Branch	Accuracy		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
ResNet152	1	67.01 ± 1.11%	**73.27 ± 0.91%**	0.78M	**0.59M**	1.66B	**1.64B**	13.77
	2*	80.26 ± 2.07%	**81.56 ± 1.57%**	0.83M	0.83M	5.33B	**5.26B**	4.29
DenseNet201	1	70.65 ± 1.23%	**76.38 ± 1.94%**	0.78M	**0.59M**	2.55B	**2.53B**	3.39
	2	81.72 ± 0.62%	**84.00 ± 1.67%**	0.80M	**0.66M**	4.21B	**4.17B**	2.06
InceptionV3	1	77.86 ± 0.90%	**79.42 ± 0.99%**	0.61M	**0.56M**	2.17B	**2.14B**	2.65
	2	78.90 ± 0.90%	**79.90 ± 0.79%**	0.61M	**0.56M**	2.53B	**2.49B**	2.28

*The last residual connection in the SL-ViT architecture was removed in this case

Table 8: Comparison of Different Early Exit Architectures on the DISCO Dataset

Backbone	Branch	MAE		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
AudioCSRNet	V1	16.99 ± 0.28	**15.04 ± 0.71**	2.50M	**2.35M**	329.77B	**328.72B**	1.49
	AV1	17.00 ± 0.23	**14.58 ± 0.64**	2.52M	**2.36M**	331.37B	**330.31B**	1.48
	AV2	17.90 ± 0.25	**17.03 ± 1.04**	2.50M	**2.35M**	374.86B	**373.81B**	1.31

mixing ratio, choice of dataset and network to generate the fake dataset, optimizer or hyper-parameters such as learning rate may result in improvements for CIFAR-100.

Table 9: Effect of Copycat strategy demonstrated on the CIFAR-10 dataset

Backbone	Branch	Accuracy	
		SL-ViT	CC-SL-ViT
ResNet152	1	70.79 ± 0.72%	**71.61 ± 0.45%**
	2	81.18 ± 0.52%	**83.41 ± 0.15%**
DenseNet201	1	76.38 ± 0.33%	**78.34 ± 0.31%**
	2	83.53 ± 0.37%	**84.89 ± 0.43%**
InceptionV3	1	79.99 ± 0.20%	**80.78 ± 0.23%**
	2	81.72 ± 0.53%	**82.20 ± 0.40%**

Table 10: Comparison of improvements gained by SL-ViT with gains from knowledge distillation for the CIFAR-10 dataset.

Backbone	Branch	CNN (Baseline)	CNN with KD	SL-ViT (Ours)
ResNet152	1	66.74 ± 0.57%	69.31 ± 0.28%	**70.79 ± 0.72%**
	2	79.31 ± 0.81%	78.79 ± 0.61%	**81.18 ± 0.52%**
DenseNet201	1	71.27 ± 0.36%	73.93 ± 0.15%	**76.38 ± 0.33%**
	2	80.64 ± 0.29%	81.56 ± 0.12%	**83.53 ± 0.37%**
InceptionV3	1	77.27 ± 0.58%	78.37 ± 0.34%	**79.99 ± 0.20%**
	2	79.55 ± 0.24%	80.41 ± 0.43%	**81.72 ± 0.53%**

Even though other early exit methods focus on improving the training procedure and can be used in combination with our proposed architecture, comparing the improvements gained by utilizing such methods with improvements gained from our approach can still provide insights into the significance of architecture design for early exits. Table 10 contains comparisons with knowledge distillation-based training similar to the method in. [12] for the CIFAR-10 dataset. Observe that in all cases, SL-ViT obtains a significantly higher accuracy compared to knowledge distillation.

5.1. Ablation Studies

Table 11 showcases the effect of using different architectural parameters on the accuracy of both SL-ViT and CNN branches. Where not specified, the CNN early exits have a 3×3 kernel size with no dilation, and the SL-ViT early exits have 12 attention heads, which are the baselines presented in previous tables. Other parameters such as the number of convolutional filters and padding size are adjusted accordingly in order to keep the number of parameters close to the baselines.

These results support our hypothesis that the improvements of SL-ViT are due to the fusion of local and global receptive fields. First, by increasing the number of attention heads in SL-ViT, the accuracy increases significantly while the parameters only slightly increase, hinting that learning multiple types of attention plays a major role in SL-ViT. Secondly, by increasing the CNN kernel size from 3×3 to 15×15 the accuracy is improved, yet it is still lower than that of SL-ViT. This is because even a large filter size does not provide a global receptive field. On the other hand, adding dilation to CNN decreases its accuracy compared to the CNN baseline. This is because dilated convolutions create holes in the receptive field, which increases the receptive field yet loses important local information. Thirdly, using two CNN layers also improves the accuracy compared to the CNN baseline, however, a higher gain in accuracy was achieved using a larger kernel size. Moreover, two SL-ViT layers still obtain a higher accuracy compared to two CNN layers while having a lower overhead in terms of parameters. Finally, we show that even if the backbone is not pre-trained on ImageNet and is trained completely from scratch, SL-ViT still obtains a higher accuracy compared to CNN.

Table 11: Ablation studies: the effect of the number of attention heads, number of layers, dilation, kernel size and backbone pre-training on the accuracy of early exits placed on the first branch location of a ResNet152 backbone trained on the CIFAR-10 dataset

Architecture Params	Accuracy	Branch Params	FLOPS
SL-ViT (1 head)	67.92 ± 0.86%	0.55M	1.64B
SL-ViT (2 heads)	68.65 ± 0.90%	0.55M	1.64B
SL-ViT (4 heads)	69.08 ± 1.07%	0.56M	1.64B
SL-ViT (8 heads)	69.85 ± 1.12%	0.58M	1.64B
SL-ViT (12 heads)	**70.79 ± 0.72%**	0.59M	1.64B
SL-ViT (16 heads)	70.76 ± 0.40%	0.61M	1.64B
CNN (3×3 kernel)	66.74 ± 0.57%	0.78M	1.66B
CNN (11×11 kernel)	69.71 ± 1.06 %	0.78M	1.88B
CNN (15×15 kernel)	69.90 ± 0.68%	0.79M	2.02B
CNN (dilation 2)	66.61 ± 0.47%	0.78M	1.66B
CNN (dilation 3)	65.43 ± 0.32%	0.78M	1.66B
SL-ViT (2 layers)	**71.89 ± 0.75%**	0.65M	1.64B
CNN (2 layers)	67.68 ± 1.06%	0.78M	1.66B
SL-ViT (no backbone pre-training)	**63.14 ± 0.57%**	0.59M	1.64B
CNN (no backbone pre-training)	62.86 ± 0.99%	0.78M	1.66B

Finally, we discovered that removing the second residual connection in the transformer encoder may lead to an increase in the overall accuracy of our method. This effect was moderate in most cases, yet quite significant in others. An example of this effect is shown in Table 12 for the Speech Commands dataset. We chose to keep the residual connection whenever the effect was moderate and only remove it if it leads to a significantly higher accuracy. Such cases are highlighted in our experiments (Table 7).

Table 12: Ablation studies: the effect of removing the last residual connection in the transformer encoder for the Speech Commands dataset

Backbone	Branch Number	Accuracy of SL-ViT	Accuracy of SL-ViT without the Last Residual
ResNet152	1	**84.05 ± 0.31%**	83.67 ± 0.85%
	2	**89.63 ± 0.52%**	85.79 ± 0.58%
DenseNet201	1	87.94 ± 0.85%	**88.35 ± 0.24%**
	2	90.93 ± 0.52%	**91.08 ± 0.52%**
InceptionV3	1	**87.62 ± 0.65%**	86.10 ± 0.32%
	2	**88.33 ± 0.92%**	88.21 ± 0.45%

5.2. Early Exit Procedure

Since our method improves the accuracy in all early exit locations, it provides improvements regardless of which early exit procedure is used. For instance, suppose a confidence-based method is used where the result of an early exit branch is selected as the final answer if it is confident enough. In this setting, our method will lead to faster inference on average, since more accurate branches lead to higher confidence.

Another example would be the anytime prediction setting explained in the introduction, for instance, an edge server which receives inputs from many IoT devices and needs to provide a response for each input within a strict deadline. The transmission time from the IoT devices to the server changes over time due to network congestion. Moreover, the computational workload of the server varies over time, therefore, the time budget available for each input is not known beforehand, and the inference can be interrupted at any moment. In this case, the output of the latest exit is used as the final answer. In such a setting, our method will lead to more accurate results and faster inference, since SL-ViT exits are more accurate and have less overhead.

To make this more clear, we have conducted experiments within the anytime prediction setting, where a random time budget is assigned to each image in the CIFAR-10 test set. We use the DenseNet backbone and the two branch locations specified in Table 2. We compare the average accuracy and FLOPS between the case where SL-ViT branches are used and the case where CNN branches are utilized. The results of these experiments are shown in Table 13. It can be observed that the multi-exit network with SL-ViT branches achieves a significantly higher average accuracy while having lower average FLOPS.

Table 13: Comparison of the average accuracy and FLOPS in the anytime prediction setting between a multi-exit DenseNet with SL-ViT early exits and one with CNN early exits.

Model	Average Accuracy	Average FLOPS
Multi-Exit DenseNet with CNN Branches	82.79 ± 0.17%	5.11B
Multi-Exit DenseNet with SL-ViT Branches	**85.65 ± 0.21%**	**5.09B**

6. Discussion and Conclusion

We showed that the proposed architecture for early exit branches, namely single-layer vision transformer (SL-ViT) can consistently obtain a significantly higher accuracy compared to conventional methods while introducing a lower overhead in terms of FLOPS. We showed that our method works for both classification and regression problems, in both single and multi-modal scenarios, and across different backbone networks and branch locations.

As previously mentioned, one possible explanation for why SL-ViT performs better, is the fact that even a single layer of transformer encoder has a global receptive field since each patch can attend to any other patch, while a convolutional layer has a limited receptive field and can only access the immediate vicinity based on its filter size. There are several clues that point to this explanation. First, Table 11 suggests that the attention mechanism plays a major role in the accuracy improvements. Secondly, based on Tables 4 to 8, the accuracy improvements are generally higher in earlier branches, where the receptive field of the backbone network up to the branch location is lower compared to later branches. Finally, the incorporation of global scale and global information such as perspective is known to be of great importance in crowd counting, and many crowd counting methods utilize visual attention mechanisms and dilated convolution layers to this end [41], which can explain why our method performs well for this problem.

Moreover, we showed that our fine-tuning strategy, namely Copycat SL-ViT, has the potential to further increase the accuracy of SL-ViT branches. It is well-known that with deep learning, more data almost always improves the final outcome, and this is especially true for vision transformers which are known to be data-hungry [32]. The copycat strategy can at times artificially increase the size of the dataset without introducing too much noise and thus improve the final result.

Furthermore, we introduced a novel approach for fusing audio and visual features within early exits using vision transformers. The importance of fusion inside early exits is that it creates much more options for branch locations, since a combination of any layer in the visual channel of the backbone network with any layer in the audio channel of the backbone can be selected. This allows for a more fine-grained dynamic inference, meaning a more recent result is available whenever the inference is interrupted in an anytime prediction setting, which is likely to be more accurate than earlier results.

Acknowledgment

This work was partly funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, and by the Danish Council for Independent Research under Grant No. 9131-00119B. This publication reflects the authors views only. The European Commission and the Danish Council for Independent Research are not responsible for any use that may be made of the information it contains.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. `doi:10.1038/nature14539`.
URL `https://doi.org/10.1038/nature14539`

[2] Y. Cheng, D. Wang, P. Zhou, T. Zhang, Model compression and acceleration for deep neural networks: The principles, progress, and challenges, IEEE Signal Processing Magazine 35 (1) (2018) 126–136.

doi:10.1109/msp.2017.2765695.
URL https://doi.org/10.1109/msp.2017.2765695

[3] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-net: ImageNet classification using binary convolutional neural networks, in: Computer Vision – ECCV 2016, Springer International Publishing, 2016, pp. 525–542. doi:10.1007/978-3-319-46493-0_32.
URL https://doi.org/10.1007/978-3-319-46493-0_32

[4] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, arXiv:1608.08710 (Unpublished results).

[5] D. T. Tran, A. Iosifidis, M. Gabbouj, Improving efficiency in convolutional neural network with multilinear filters, Neural Networks 105 (2018) 328–339.

[6] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv:1503.02531 (Unpublished results).

[7] J. Chen, X. Ran, Deep learning with edge computing: A review, Proceedings of the IEEE 107 (8) (2019) 1655–1674. doi:10.1109/jproc.2019.2921977.
URL https://doi.org/10.1109/jproc.2019.2921977

[8] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, X. Chen, Convergence of edge computing and deep learning: A comprehensive survey, IEEE Communications Surveys & Tutorials 22 (2) (2020) 869–904. doi:10.1109/comst.2020.2970550.
URL https://doi.org/10.1109/comst.2020.2970550

[9] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, Y. Wang, Dynamic neural networks: A survey, arXiv:2102.04906 (Unpublished results).

[10] S. Scardapane, M. Scarpiniti, E. Baccarelli, A. Uncini, Why should we add early exits to neural networks?, Cognitive Computation 12 (5) (2020) 954–966. `doi:10.1007/s12559-020-09734-4`.
URL `https://doi.org/10.1007/s12559-020-09734-4`

[11] S. Teerapittayanon, B. McDanel, H. Kung, BranchyNet: Fast inference via early exiting from deep neural networks, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016. `doi:10.1109/icpr.2016.7900006`.
URL `https://doi.org/10.1109/icpr.2016.7900006`

[12] M. Phuong, C. Lampert, Distillation-based training for multi-exit architectures, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, 2019. `doi:10.1109/iccv.2019.00144`.
URL `https://doi.org/10.1109/iccv.2019.00144`

[13] A. Bakhtiarnia, Q. Zhang, A. Iosifidis, Improving the accuracy of early exits in multi-exit architectures via curriculum learning, in: 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–8. `doi:10.1109/IJCNN52387.2021.9533875`.

[14] A. Bakhtiarnia, Q. Zhang, A. Iosifidis, Multi-exit vision transformer for dynamic inference, The 32nd British Machine Vision Conference (BMVC 2021) (2021).

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations, 2021.
URL `https://openreview.net/forum?id=YicbFdNTTy`

[16] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, T. Oliveira-Santos, Copycat CNN: Stealing knowledge by persuading confession with random non-labeled data, in: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, 2018. `doi:10.1109/ijcnn.2018.8489592`.
URL `https://doi.org/10.1109/ijcnn.2018.8489592`

[17] E. Baccarelli, S. Scardapane, M. Scarpiniti, A. Momenzadeh, A. Uncini, Optimized training and scalable implementation of conditional deep neural networks with early exits for fog-supported IoT applications, Information Sciences 521 (2020) 107–143. `doi:10.1016/j.ins.2020.02.041`.
URL `https://doi.org/10.1016/j.ins.2020.02.041`

[18] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, K. Ma, Be your own teacher: Improve the performance of convolutional neural networks via self distillation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3713–3722.

[19] H. Li, H. Zhang, X. Qi, R. Yang, G. Huang, Improved techniques for

training adaptive deep networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1891–1900.

[20] M. Wołczyk, B. Wójcik, K. Bałazy, I. Podolak, J. Tabor, M. Śmieja, T. Trzcinski, Zero time waste: Recycling predictions in early exit neural networks, Advances in Neural Information Processing Systems 34 (2021).

[21] Z. Jie, P. Sun, X. Li, J. Feng, W. Liu, Anytime recognition with routing convolutional networks, IEEE transactions on pattern analysis and machine intelligence 43 (6) (2019) 1875–1886.

[22] J. Pomponi, S. Scardapane, A. Uncini, A probabilistic re-intepretation of confidence scores in multi-exit models, Entropy 24 (1) (2021) 1.

[23] H. Lee, C.-J. Hsieh, J.-S. Lee, Local critic training for model-parallel learning of deep neural networks, IEEE Transactions on Neural Networks and Learning Systems (2021).

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[25] M. Elbayad, J. Gu, E. Grave, M. Auli, Depth-adaptive transformer, in: International Conference on Learning Representations, 2020.
URL `https://openreview.net/forum?id=SJg7KhVKPH`

[26] A. Graves, Adaptive computation time for recurrent neural networks, arXiv:1603.08983 (2016).

[27] A. Banino, J. Balaguer, C. Blundell, Pondernet: Learning to ponder, arXiv:2107.05407 (2021).

[28] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, J. E. Gonzalez, Skipnet: Learning dynamic routing in convolutional networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[29] X. Gao, Y. Zhao, L. Dudziak, R. D. Mullins, C. Xu, Dynamic channel pruning: Feature boosting and suppression, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.
URL `https://openreview.net/forum?id=BJxh2j0qYm`

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.
URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`

[31] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, D. Tao, A survey on visual transformer, arXiv:2012.12556 (Unpublished results).

[32] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, M. Shah, Transformers in vision: A survey, arXiv:2101.01169 (Unpublished results).

[33] S. F. Bhat, I. Alhashim, P. Wonka, Adabins: Depth estimation using adaptive bins, arXiv:2011.14141 (Unpublished results).

[34] K. Choi, G. Fazekas, M. Sandler, Automatic tagging using deep convolutional neural networks, arXiv:1606.00298 (Unpublished results).

[35] J. Lee, T. Kim, J. Park, J. Nam, Raw waveform-based audio classification using sample-level cnn architectures, arXiv:1712.00866 (Unpublished results).

[36] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016. doi:10.1109/cvpr.2016.90.
URL https://doi.org/10.1109/cvpr.2016.90

[37] G. Huang, Z. Liu, L. V. D. Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017. doi:10.1109/cvpr.2017.243.
URL https://doi.org/10.1109/cvpr.2017.243

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016. doi:10.1109/cvpr.2016.308.
URL https://doi.org/10.1109/cvpr.2016.308

[39] K. Palanisamy, D. Singhania, A. Yao, Rethinking cnn models for audio classification, arXiv:2007.11154 (Unpublished results).

[40] V. A. Sindagi, V. M. Patel, A survey of recent advances in CNN-based single image crowd counting and density estimation, Pattern Recognition Letters 107 (2018) 3–16. doi:10.1016/j.patrec.2017.07.007.
URL https://doi.org/10.1016/j.patrec.2017.07.007

[41] G. Gao, J. Gao, Q. Liu, Q. Wang, Y. Wang, Cnn-based density estimation and crowd counting: A survey, arXiv:2003.12783 (Unpublished results).

[42] Y. Zhang, D. Zhou, S. Chen, S. Gao, Y. Ma, Single-image crowd counting via multi-column convolutional neural network, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016. doi:10.1109/cvpr.2016.70.
URL https://doi.org/10.1109/cvpr.2016.70

[43] C. Zhang, H. Li, X. Wang, X. Yang, Cross-scene crowd counting via deep convolutional neural networks, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015. doi:10.1109/cvpr.2015.7298684.
URL https://doi.org/10.1109/cvpr.2015.7298684

[44] F. Dai, H. Liu, Y. Ma, J. Cao, Q. Zhao, Y. Zhang, Dense scale network for crowd counting, arXiv:1906.09707 (Unpublished results).

[45] D. Hu, L. Mou, Q. Wang, J. Gao, Y. Hua, D. Dou, X. X. Zhu, Ambient sound helps: Audiovisual crowd counting in extreme conditions, arXiv:2005.07097 (Unpublished results).

[46] Y. Li, X. Zhang, D. Chen, CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2018. `doi:10.1109/cvpr.2018.00120`.
URL `https://doi.org/10.1109/cvpr.2018.00120`

[47] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL `http://arxiv.org/abs/1409.1556`

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009. `doi:10.1109/cvpr.2009.5206848`.
URL `https://doi.org/10.1109/cvpr.2009.5206848`

[49] 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, K. Wilson, CNN architectures for large-scale audio classification, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017. `doi:10.1109/icassp.2017.7952132`.
URL `https://doi.org/10.1109/icassp.2017.7952132`

[50] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: F. Bach, D. Blei (Eds.),

Proceedings of the 32nd International Conference on Machine Learning, Vol. 37 of Proceedings of Machine Learning Research, PMLR, Lille, France, 2015, pp. 448–456.
URL `http://proceedings.mlr.press/v37/ioffe15.html`

[51] T. Hu, T. Chen, H. Wang, Z. Wang, Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference, in: ICLR, 2020.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[53] A. LeNail, NN-SVG: Publication-ready neural network architecture schematics, Journal of Open Source Software 4 (33) (2019) 747. `doi:10.21105/joss.00747`.
URL `https://doi.org/10.21105/joss.00747`

[54] A. Araujo, W. Norris, J. Sim, Computing receptive fields of convolutional neural networks, DistillHttps://distill.pub/2019/computing-receptive-fields (2019). `doi:10.23915/distill.00021`.

[55] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6230–6239. `doi:10.1109/CVPR.2017.660`.

[56] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE Transactions on Pattern Analy-

sis & Machine Intelligence 40 (04) (2018) 834–848. doi:10.1109/TPAMI.2017.2699184.

[57] A. Krizhevsky, Learning multiple layers of features from tiny images (Unpublished results).

[58] P. Warden, Speech commands: A dataset for limited-vocabulary speech recognition, arXiv:1804.03209 (Unpublished results).

[59] G. Tzanetakis, P. Cook, Musical genre classification of audio signals, IEEE Transactions on Speech and Audio Processing 10 (5) (2002) 293–302. doi:10.1109/tsa.2002.800560.
URL https://doi.org/10.1109/tsa.2002.800560

[60] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL http://arxiv.org/abs/1412.6980

[61] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.
URL https://openreview.net/forum?id=Bkg6RiCqY7

[62] J. Frankle, D. J. Schwab, A. S. Morcos, Training batchnorm and only batchnorm: On the expressive power of random features in cnns, arXiv:2003.00152 (Unpublished results).

[63] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

7.12 Feedforward Neural Networks Initialization based on Discriminant Learning

The appended paper follows.

Feedforward Neural Networks Initialization based on Discriminant Learning

Kateryna Chumachenko[a,*], Alexandros Iosifidis[b], Moncef Gabbouj[a]

[a]*Faculty of Information Technology and Communication Sciences, Tampere University, FI-33720 Tampere, Finland*
[b]*Department of Electrical and Computer Engineering, Aarhus University, DK-8200 Aarhus, Denmark*

Abstract

In this paper, a novel data-driven method for weight initialization of Multilayer Perceptrons and Convolutional Neural Networks based on discriminant learning is proposed. The approach relaxes some of the limitations of competing data-driven methods, including unimodality assumptions, limitations on the architectures related to limited maximal dimensionalities of the corresponding projection spaces, as well as limitations related to high computational requirements due to the need of eigendecomposition on high-dimensional data. We also consider assumptions of the method on the data and propose a way to account for them in a form of a new normalization layer. The experiments on three large-scale image datasets show improved accuracy of the trained models compared to competing random-based and data-driven weight initialization methods, as well as better convergence properties in certain cases.

Keywords: Neural networks initialization, discriminant learning

1. Introduction

In recent years, Deep Learning became the dominant paradigm in the fields of Machine Learning and Computer Vision owing to the availability of large public data and computational resources. Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) are being widely utilized for a variety of tasks, including object detection [1, 2, 3], object tracking [4], semantic image segmentation [5] and action recognition [6, 7].

With the rise of Deep Learning, methods for weight initialization in neural networks received increased attention, and weight initialization strategies that can accelerate the training process while leading to competitive performance remain an open research problem. Multiple approaches have been proposed to

*Corresponding author
Email addresses: `kateryna.chumachenko@tuni.fi` (Kateryna Chumachenko), `ai@ece.au.dk` (Alexandros Iosifidis), `moncef.gabbouj@tuni.fi` (Moncef Gabbouj)

solve this problem to date. Early works in the field of artificial neural networks were relying on weight initialization from random distributions, further evolving to initialization methods with controlled parameters, such as Glorot [8] or He initialization [9]. Others methods proposed data-driven initialization procedures [10, 11, 12, 13, 14, 15], which are described in more detail in Section 2.1. The main motivation behind the latter approach primarily stems from the nature of training processes of neural networks: since gradient-based optimization of non-convex functions leads to local minima solutions, starting the optimization from a favourable point can result in better performance and faster convergence.

Several data-driven initialization methods were proposed based on statistical learning, primarily focusing on utilization of Principal Component Analysis (PCA) [16] or Linear Discriminant Analysis (LDA) [17] to determine the data transformations in successive layers of the network. Nevertheless, these methods have a number of limitations: PCA only satisfies the criteria of high variance in the data while not enforcing discriminative properties, and LDA assumes unimodal class distributions for the data representations in all the layers of the neural network. Here it should be noted that while data representations at the last hidden layer of a trained neural network equipped with softmax/linear output neurons are expected to form unimodal classes, this is not the case for early layers. Therefore, the assumption of class unimodality throughout the layers of the network for weight initialization limits the potential of the model. Another major limitation comes from the limited dimensionality of the projection directions learnt by these methods, thus limiting the number of neurons/weights that can be initialized by adopting them.

As a remedy for the above-mentioned limitations, in this paper, we propose a novel data-driven weight initialization approach based on discriminant learning that allows to relax the above-mentioned limitations. First, we relax the class unimodality assumption for the data representations at all network layers by representing it with several subclasses and formulating the optimization problem for weights initialization accordingly, hence improving the suitability of a model for real-world scenarios. Second, the proposed approach relaxes limitations to the model architecture, as the maximal number of initialized neurons/filters at a certain layer relies on a controlled parameter, i.e. the total number of subclasses forming the classification problem. Third, the proposed approach does not rely on eigendecomposition that becomes computationally intensive for high-dimensional data, hence providing faster initialization especially for wide CNN architectures, i.e., those with a large number of neurons/filters in each layer.

The main contributions of the paper can be summarized as follows:

- A novel weight initialization procedure for MLPs and CNNs is proposed that leads to flexible network architecture design and potentially better generalization due to its multi-modal formulation. It is experimentally shown that the adoption of the proposed initialization procedure leads to faster convergence of the subsequent gradient-based training process compared to existing approaches.

- A new normalization layer that overcomes limitations related to the assumption of mean-centered data, adopted by the proposed method, as well as other data-driven network initialization methods is proposed.
- Experimental results show that utilization of a small number of data samples generally suffices for effective network initialization, hence, further reducing the computational requirements for training the network.

The remainder of the paper is structured as follows. Section 2 describes the related methods utilized for weight initialization in neural networks, Section 3 describes the proposed weight initialization approach along with the motivation behind it, Section 4 presents the experiments performed to assess the proposed approach, along with the experimental results, and Section 5 provides conclusions of the work.

2. Related Work

Generally, methods for weight initialization of neural networks can be divided into two categories: the first is based on initialization from a random distribution and the second follows a data-driven process. For a long time, the most widely-used and straightforward initialization approach was the initialization from a random distribution: a Gaussian distribution with zero mean and small hand-tuned standard deviation, or from a Uniform distribution in the range of $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$, where n is the number of input neurons in the corresponding layer. It has been further observed that such initialization often leads to poor convergence, and saturated activations. In [8], it was shown that the commonly-used activation functions, namely, sigmoid, hyperbolic tangent, and softsign suffer from saturation of activation in the top layers of the network, when initialized from random uniform distribution. As a remedy, a new weight initialization method was proposed, with an objective of preserving the variance of activation vectors between the layers during the forward propagation, and the variance of the gradients between the layers during backward propagation. In practice, the following initialization approach is utilized, approximately satisfying the above-mentioned objectives:

$$\mathbf{W}_j \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right], \tag{1}$$

where $\mathbf{W}_j$ is the weight matrix at layer j, $U[\cdot]$ denotes the Uniform distribution, and n_j and n_{j+1} denote the number of neurons at layers j and $j+1$, respectively. Hereafter, we refer to this approach as Glorot initialization (also commonly referred to as Xavier initialization) [8] . Here we should note that, in its derivation, the method assumes linear activations at the initialization and that the input feature variances are equal.

A further step towards controlling the statistics of the distribution from which the weights are initialized was taken in [9], where a similar motivation to that of Glorot is utilized for initialization. Unlike the work in [8], the authors consider ReLU activation, and show that the proposed approach outperforms the Glorot initialization especially when used for deep neural networks. The initialization is done as follows:

$$\mathbf{W}_j \sim \mathcal{N}\left[0, \frac{\sqrt{2}}{\sqrt{n_j}}\right], \tag{2}$$

i.e., the weights of layer j are initialized from a Gaussian distribution with zero mean and variance of $\frac{2}{n_j}$. Additionally, fully-randomized methods based on stochastic configuration algorithms have been proposed [18].

As opposed to methods based on random initialization, multiple approaches exploiting certain data properties have recently been proposed. The most notable one is initialization by pre-training on a larger dataset of similar domain[1], such as ImageNet [20] for Computer Vision tasks. Nevertheless, such initialization was questioned in [21], where it was shown that the benefits arising from weights initialization based on pre-training generally lie in faster convergence in earlier iterations, but not necessarily leading to better performance as compared to random initialization. Other notable data-driven approaches include initialization from cluster centroids obtained by applying (spherical) clustering on whitened data, hence capturing statistical properties of the dataset [22, 10, 23]. Another method performs normalization of networks' weights based on the empirical statistics of the network activation obtained from the training data samples, as well as its gradients [10] Notably, the approach presented in [10] applies the normalization to both k-means and PCA initialized networks.

2.1. Weight initialization via subspace learning

A set of data-driven weight initialization methods that were proven beneficial for weight initialization in neural networks relies on utilization of subspace learning techniques. The early works utilizing subspace learning for determining the weights of neural networks include PCANet [13] and LDANet [14]. These methods focus on supervised image classification in a CNN-like manner, where a set of patches are extracted from the training images and flattened to form a data matrix. From this data representation, a weight matrix is obtained by applying Principal Component Analysis [16] or Linear Discriminant Analysis [17]. The resulting weight matrix is subsequently reshaped to obtain a set of convolutional filters, which are convolved with the training images to obtain the data representations at the output of the first layer. This process is applied for several layers[2], followed by a pooling operation and an activation function. In these approaches, no subsequent fine-tuning of the network's parameters via back-propagation is performed, while the pooling operation as well as the uti-

[1]This approach is commonly referred to as transfer learning [19].

[2]The original LDANet and PCANet methods apply this process twice to determine the filters of two convolutional layers.

lized activation function are specially designed, i.e. they are not among the commonly-used ones in the field of deep learning. However, these methods can be perceived as the first baselines drawing the connection between the subspace learning and deep learning methods.

Further notable attempts of linking subspace learning with deep learning architectures include LDA-based weight initialization proposed in [15, 24]. By its nature, this work is more similar to our proposed approach in that the weights obtained by a discriminant learning method are used for initialization of the neural network which is further trained with backpropagation, instead of solely considering the forward propagation scenario. LDA is employed to initialize the weights of a layer, and each subsequent layer is initialized from the weight matrix obtained by LDA applied to the outputs of the preceding layer. Similarly to PCANet and LDANet, the weight matrix is learnt from patches extracted from the outputs of the previous layer and is flattened to obtain a rectangular data matrix. The last classification layer is initialized with the discriminant matrix of LDA, and the network is subsequently trained with backpropagation.

The authors in [11, 12] proposed a feedforward design approach for initializing the layers in CNN based on data statistics from the output of their preceding layers. The weights in convolutional layers are obtained from a variant of Principal Component Analysis proposed by the authors, namely, Subspace Approximation with Adjusted Bias (Saab). The dense layers that are added after the convolutional layers are trained by applying a linear regression using subclass labels obtained by clustering the data. The last fully-connected layer is trained by linear regression to true class labels. This method focuses on the forward propagation scenario too.

3. Initialization based on Discriminant Learning

Let us consider a standard dense feedforward neural network. Given a vector $\mathbf{x} \in \mathbb{R}^D$ as input, a neural network with L layers applies a hierarchical transformation

$$\mathbf{y} = f_a^L(\mathbf{W}_L^T f_a^{L-1}(\ \mathbf{W}_{L-1}^T \dots f_a^1(\mathbf{W}_1^T \mathbf{x} + b_1) + b_{L-1}) + b_L), \tag{3}$$

where $f_a^l(\cdot)$ is the (element-wise) activation function at layer l, $\mathbf{W}_l \in \mathbb{R}^{D_l \times D_{l+1}}$ is the corresponding weight matrix, and b_l is the bias term. For the sake of simplicity of notation, here we assume that the bias terms are accounted for by using an augmented version of the data representations of the network layers and, thus, are incorporated in the corresponding weight matrices $\mathbf{W}_l$, $l = 1, \dots, L$. Similarly, a CNN performs a hierarchical data transformation of the form

$$\mathbf{y} = f_a^L(\hat{\mathbf{W}}_L * f_a^{L-1}(\ \hat{\mathbf{W}}_{L-1} * \ \dots \ * f_a^1(\hat{\mathbf{W}}_1 * \mathbf{x} + b_1) + b_{L-1}) + b_L), \tag{4}$$

where $\hat{\mathbf{W}}_l$ is a set of convolutional filters at layer l, b_l is the bias term, and $f_a^l(\cdot)$ is the activation function. For CNNs which combine convolutional and dense layers, the corresponding data transformation is obtained by simply combining data transformations of the form in (4) and (3) in a hierarchical manner.

3.1. Motivation

Most of the earlier data-driven methods primarily focused on the affine transformation of $\mathbf{y} = \mathbf{W}_l^T \mathbf{x}^{(l)}$, where $\mathbf{x}^{(l)}$ is the representation of the input sample at the feature space defined at layer l. To deal with the convolution operation $\mathbf{y} = \hat{\mathbf{W}}_l * \mathbf{x}^{(l)}$ in (4), the convolution operation is transformed to a vector-based affine transformation by sampling patches from the input $\mathbf{x}^{(l)}$, flattening them to create vectors and determining an affine transformation matrix $\mathbf{W}_l$, which is further reshaped to form $\hat{\mathbf{W}}_l$. Several works [14, 15] utilize LDA for learning the matrix $\mathbf{W}_l$, i.e., the projection is obtained by solving the eigendecomposition problem of $\mathbf{S}_w^{(l)} \mathbf{w} = \lambda \mathbf{S}_b^{(l)} \mathbf{w}$ and selecting eigenvectors corresponding to smallest eigenvalues, where $\mathbf{S}_w^{(l)}$ and $\mathbf{S}_b^{(l)}$ are the within-class and between-class scatter matrices defined on the data representations at the layer l. Others [13, 11, 12] have applied Principal Component Analysis, i.e, the matrix $\mathbf{W}$ is obtained by performing eigendecomposition on the covariance matrix of the data representations at layer l, i.e. $\mathbf{S}_t^{(l)}$.

Both of these approaches have certain limitations. Being an unsupervised method, PCA does not take advantage of the class label information of the data. Therefore, one of its limitations lies in the fact that the learnt subspace is only optimal in terms of preserving the variance of the projected data; however, no discriminative properties are enforced. Besides, PCA can only learn a (sub)space with dimensionality at most equal number of dimensions to that of the original space. This leads to the inability of learning enough meaningful (i.e., those having discriminative properties) filters/neurons, as the number of filters of the first layers is generally significantly higher than that of dimensions in the the input data, especially in the case of CNNs.

Linear Discriminant Analysis provides a remedy to the limitation of PCA related to the disregard of the class label information of data, finding a subspace where the classes are discriminated. However, it relies on an assumption of unimodality of data of each class, which is rarely the case in real-world scenarios, and especially on the earlier layers of the networks. As a result, such an assumption leads to limitations in the learning potential of the model. Besides, the limitation of LDA with regard to the ability to learn a reasonable amount of meaningful neurons or filters is even higher than that of PCA, as the dimensionality of the learnt subspace is bounded by the rank of the between-class scatter matrix, which is, in turn, bounded by the number of classes. Therefore, the use of LDA for initialization only allows to obtain a very limited number of meaningful projection dimensions, and, consequently, a limited number of meaningful neuron weights in the layer, putting limitations on the network architectures that can be initialized using it.

In addition to the above-mentioned limitations, one can notice that both LDA and PCA rely on eigendecomposition of $D \times D$ matrices that becomes computationally intensive especially for high-dimensional data. At the same time, especially in the case of CNN, the data is likely to reach significantly high dimensionality: given the data matrix is created similarly to [10, 15, 13], the dimensionality of the patch matrix corresponding to layer j reaches $k^2 n_j$, where k is the filter size, and n_j is the number of filters. Considering commonly-used CNN models, where the number of filters of convolutional layers generally ranges from 32 to 512, and a commonly-used filter size of 5 pixels, this leads to dimensionality ranging from 800 up to 12800, which is substantially high in terms of computational requirements of eigendecomposition-based subspace learning methods. For example, in this case the computational complexity of initialization based on LDA or PCA would reach $N(k^2 n_j)^2 + (k^2 n)^3$ [25], while for the proposed approach it is proportional to $Nk^2 n_j d$ or $k^2 n_j N^2$ if $N < k^2 n_j$ and $N(k^2 n_j)^2$ if $N > k^2 n_j$ [26], where N is the number of samples and d is the dimensionality of the learnt space, which is in either case less than the complexity of initialization based on LDA or PCA.

3.2. Proposed approach

In this section we consider the limitations of already existing methods and propose steps for their relaxation. More specifically, we consider assumptions on unimodality of data representations in the layers of a network, limitations in the number of neurons/filters that can be initialized, and the high computational requirements in high-dimensional settings. A first step towards overcoming these limitations can be taken by employing Subclass Discriminant Analysis [27], that relaxes the assumptions on unimodality of classes. To recall, this is achieved by expressing each class with a set of subclasses determined by applying some clustering algorithm on the data of each class. Similarly to LDA, SDA optimizes the Fisher-Rao's criterion. Considering the optimization problem to be solved for initializing the weights of the l-th layer, the generalized eigenanalysis problem $\mathbf{S}_t^{(l)}\mathbf{w} = \lambda \mathbf{S}_b^{(l)}\mathbf{w}$ is solved, where

$$\mathbf{S}_t^{(l)} = \sum_{i=1}^{N} (\mathbf{x}_i^{(l)} - \boldsymbol{\mu}^{(l)})(\mathbf{x}_i^{(l)} - \boldsymbol{\mu}^{(l)})^T, \tag{5}$$

$$\mathbf{S}_b^{(l)} = \sum_{i=1}^{C-1} \sum_{n=i+1}^{C} \sum_{j=1}^{K_i} \sum_{h=1}^{K_n} p_{ij}^{(l)} p_{nh}^{(l)} (\boldsymbol{\mu}_{ij}^{(l)} - \boldsymbol{\mu}_{nh}^{(l)})(\boldsymbol{\mu}_{ij}^{(l)} - \boldsymbol{\mu}_{nh}^{(l)})^T, \tag{6}$$

where C is the number of classes, K_i is the number of subclasses in class i, $\boldsymbol{\mu}^{(l)}$ is the mean of the data representations in layer l, i and n are class labels, and j and h are subclass labels. $p_{ij}^{(l)}$ and $p_{lh}^{(l)}$ are the subclass priors, i.e. $p_{ij}^{(l)} = \frac{N_{ij}}{N}$, where N_{ij} is the number of samples in subclass j of class i and N is the total number of samples in $\mathbf{X}^{(l)} = [\mathbf{x}_1^{(l)}, \ldots, \mathbf{x}_N^{(l)}] \in \mathbb{R}^{D_l \times N}$. The matrix $\mathbf{W}_l \in \mathbb{R}^{D_l \times D_{l+1}}$ can be then formed by the eigenvectors corresponding to the D_{l+1} smallest eigenvalues.

Such representation is particularly beneficial in the CNN case, where each data sample constitutes a representation of a patch from an image. Assuming that patches within the same class corresponding to non-essential background and those representing the object of interest or certain useful features are clustered into different subclasses, there is no penalization for them being matched far from each other in the learnt feature space. In contrast, LDA forces all data samples belonging to the same class to lie close to each other in the projection space, enforcing unnecessary similarity requirements for essential features and background patches. Moreover, by utilizing SDA the potential dimensionality of the subspace is bounded by the total number of subclasses forming the problem at hand. That is, the maximum number of discriminant directions that can be determined is increased to $\sum_{i=1}^{C} K_i$. The potential set of architectures is, therefore, significantly expanded compared to LDA. However, it is still bounded by the dimensionality of input data. We propose to overcome this limitation by following a process inspired by Graph Embedding [28] and Spectral Regression [29] in the following.

The criterion function of SDA can be reformulated utilizing Graph Embedding framework [28]. For data centered at $\boldsymbol{\mu}^{(l)}$, it can be seen that

$$\mathbf{S}_t^{(l)} = \mathbf{X}^{(l)}\mathbf{X}^{(l)T}, \tag{7}$$

$$\mathbf{S}_b^{(l)} = \mathbf{X}^{(l)}\mathbf{L}_b^{(l)}\mathbf{X}^{(l)T}, \tag{8}$$

where $\mathbf{L}_b^{(l)}$ is the Laplacian matrix defined on the data representations at the l-th layer of the network for the between-class matrix:

$$\mathbf{L}_b^{(l)}(i,j) = \begin{cases} \frac{N-N_{c_i}}{N^2 N_{ch}}, & \text{if } z_i^{(l)} = z_j^{(l)} = h \\ 0, & \text{if } z_i^{(l)} \neq z_j^{(l)},\, c_i = c_j \\ -\frac{1}{N^2}, & \text{if } c_i \neq c_j \end{cases}, \tag{9}$$

where c_i is the class label of $\mathbf{x}_i^{(l)}$, and $z_i^{(l)}$ is the subclass label of $\mathbf{x}_i^{(l)}$, N_c is the number of samples in class c and $N_{ch}^{(l)}$ is the number of samples in subclass h of class c at layer l.

Exploiting the new formulations of $\mathbf{S}_b^{(l)}$ and $\mathbf{S}_t^{(l)}$, and Spectral Regression [29], the solution can be obtained by following several steps:

1. The between-class Laplacian matrix $\mathbf{L}_b^{(l)}$ is created following Eq. 9.
2. Assuming there exists such $\mathbf{t}$ that $\mathbf{t} = \mathbf{X}^{(l)T}\mathbf{w}$, the eigenanalysis problem $\mathbf{L}_b^{(l)}\mathbf{t} = \lambda\mathbf{t}$ is solved and the matrix $\mathbf{T}^{(l)}$ is created out of the obtained vectors.
3. The regression of $\mathbf{T}^{(l)}$ to $\mathbf{W}^{(l)}$ is performed as

$$\mathbf{W}^{(l)} = \left(\mathbf{X}^{(l)}\mathbf{X}^{(l)T} + \alpha\mathbf{I}\right)^{-1}\mathbf{X}^{(l)}\mathbf{T}^{(l)T}. \tag{10}$$

The matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{D_l \times D_{l+1}}$ can be further orthogonalized or l_2-normalized. In practice, we observed that l_2-normalization results in better performance. Moreover, when applying l_2-normalization instead of orthogonalization, the number of projection directions D_{l+1} can be expanded beyond the dimensionality D_l of the data representations at layer l. This is achieved by performing a class-wise clustering process to determining $\sum_{i=1}^{C} K_i > D_{l+1}$ subclasses, and using the eigenvectors of $\mathbf{L}_b^{(l)}$ corresponding to the largest D_{l+1} eigenvalues to form $\mathbf{T}^{(l)}$. Such an approach allows us to define the number of neurons in layer $l+1$ by controlling the total number of subclasses in layer l, leading to the initialization of as many meaningful neurons as is required by the architecture of the network. Note that, due to the block structure of $\mathbf{L}_b^{(l)}$, the first $C-1$ dimensions are guaranteed to encode the class discriminant information, similarly to LDA. In this sense, the layer initialized using the proposed approach is guaranteed to have at least the same discriminative power as using LDA.

Here we should note that the use of clustering and subsequent cluster label information has been previously performed in [22, 11]. In [22], clustering is applied to the whole dataset and the cluster centroids are used for initialization. In [11], clustering is applied to the whole dataset and one-hot encoded vectors are created using the obtained cluster labels, followed by a least-squares regression to obtain the projection matrix used for initialization. In both of these settings, however, the class label information is not considered. Therefore, the use of such methods in a supervised setting is rather limited. Besides, the proposed approach determines the projection directions in which the data achieves optimal subclass separability, rather than regressing directly to the cluster labels.

The proposed approach can further be extended to improve the computational efficiency on large datasets, where eigendecomposition of $\mathbf{L}_b^{(l)}$ becomes infeasible. The speed-up is achieved by observing that $\mathbf{L}_b^{(l)}$ has a certain block structure, therefore its eigenvectors have a similar block structure as well. Given that a vector of ones is an eigenvector of $\mathbf{L}_b^{(l)}$, we can create the $\sum_{i=1}^{C} K_i - 1$ target vectors of random values with desired structure and orthogonalize them starting from a vector of ones. The detailed procedure for creation of target vectors is shown in Algorithm 1. The approach has recently been shown beneficial in a conventional subspace learning setting for speeding up eigendecopmposition-based SDA [26, 30], and an incremental solution was proposed [31]. While the methods in [26, 30, 31] were proposed for purely shallow statistical learning, here we investigate the utilization of similar ideas for data-driven neural network initialization. The suitability of the proposed ideas for network initialization is dictated by a range of advantages provided by the method in terms of accounting for potential multi-modality present in intermediate layers of the network, faster initialization compared to conventional data-driven methods, as well as absence of restrictions in terms of width of neural network layers. At the same time we investigate ways of addressing the limitations of the methodology in terms of assumptions on data properties. For the sake of clarity, hereafter we refer to the proposed initialization approach as fastSDA as defined in [26, 30, 31] in contrary to eigendecomposition-based SDA.

Algorithm 1: Discriminant target vectors calculation

```
Function getTargets($y$,$y_{cl}$,$C$,$Z$,$N$,$D$):
    Input: $y$ : $N \times 1$ vector with class labels; $y_{cl}$ : $N \times 1$ vector with the cluster
           labels; $Z$ : number of clusters in each class; $C$ : number of classes; $N$ :
           number of elements; $D$ : dimensionality of data;
    %class-level vectors;
    for $i \leftarrow$ iterate through 1:C do
        $RVals$ = random(1,$C$-1)
        $T^{(l)}[y == i, :]$ = tile($RVals$, len($y$==$i$),1)
    end
    $S \leftarrow$ unique numbers of elements in each class sorted in ascending order;
    %cluster level vectors;
    for $s \leftarrow$ iterate through $S$ do
        $k \leftarrow$ classes with $s$ elements; $m \leftarrow$ length($k$);
        $RVals$ = random($m * Z$, $m$*($Z$ - 1))
        for $i \leftarrow$ iterate through $k$ do
            for $j \leftarrow$ iterate through 1:Z do
                $ixs$ = where($y$ == $i$ & $y_{cl}$ == $j$)
                $Tclust^{(l)}[ixs,:] \leftarrow$ tile($RVals$, (length($ixs$),1))
            end
        end
        $T^{(l)} \leftarrow$ append $Tclust^{(l)}$ as columns on the right;
    end
    $T^{(l)} \leftarrow$ append $N \times 1$ vector of ones as a column on the left;
    Orthogonalize $T^{(l)}$; remove first column of $T^{(l)}$;
    return $T^{(l)T}$
```

Algorithm 2: Initialization of l^{th} Dense layer.

```
Function dense_init($X^{(l)}$,$y$,$N\_neur$,$C$,$D$):
    Input: $X^{(l)}$ : $D \times N$ data representation at $l^{th}$ layer; $y$ : $N \times 1$ vector with
           class labels;$N\_neur$ :number of neurons;
    $Z \leftarrow$ceil($N\_neur/C$)
    $X^{(l)} \leftarrow \frac{X^{(l)} - mean(X^{(l)})}{\sqrt{var(X^{(l)}) + \epsilon}}$
    $y_{cl} \leftarrow$ Cluster($X^{(l)}$, $Z$)
    $T^{(l)} \leftarrow$ getTargets($y$,$y_{cl}$,$C$,$Z$,$N$,$D$)
    if $D < N$ then
        $R \leftarrow (\text{chol}(X^{(l)} X^{(l)T}))^{-1}$
        $W^{(l)} \leftarrow R R^T X^{(l)} T^{(l)T}$
    else
        $R \leftarrow (\text{chol}(X^{(l)T} X^{(l)}))^{-1}$
        $W^{(l)} \leftarrow X^{(l)} R^T R T^{(l)T}$
    end
    $W^{(l)} \leftarrow$Select first $N\_neur$ dimensions of $W^{(l)}$ and normalize with $l2$ norm
return $W^{(l)}$
```

3.3. Initialization procedures

The proposed approach can be used for initializing Dense and Convolutional layers following equations (3) and (4). The initialization procedure starts from learning the weight matrix of the first layer on the input data. The data is further transformed with the learnt matrix and transformations defined by the architecture, e.g., Activation and Pooling. The transformed data is subsequently used for initializing the next Dense or Convolutional layer, and the process continues until the Output layer, which is initialized randomly[3]. After the initialization of the whole network, it is trained with backpropagation in the conventional manner. The procedures for initializing weight matrix $\mathbf{W}^{(l)}$ and filters $\hat{\mathbf{W}}^{(l)}$ for the l^{th} Dense or Convolutional layer are shown in Algorithms 2 and 3, respectively.

In order to account for the mean-centering assumption on the data, the input data at each layer is standardized during the weight initialization step. Therefore, to take this into account during the backpropagation step, we add Batch Normalization layers before each of the Dense layers in the architecture.

3.4. Vector Batch Normalization

Initializing the parameters of a neural network with the proposed method requires the training data to be mean-centered, such that Eqs. (7) and (8) express the total and the between-class scatter of the training data. For initializing Dense layers, this is accounted by means of Batch Normalization. For Convolutional layers, the standard Batch Normalization does not satisfy our needs, since the normalization is done using the per-channel mean and variance. Instead, we would like to normalize the feature maps in a way that would produce the mean-centered rectangular patch matrix. In other words, we seek to standardize each non-overlapping $k \times k \times d_{l-1}$ patch with the mean and standard deviation of all such patches (or alternatively, all patches in a mini-batch). Therefore, to account for mean-centering in Convolutional layers, we introduce a new normalization layer that we further refer to as Vector Batch Normalization. We extract all non-overlapping $k \times k \times d_{l-1}$ dimensional patches from the input appropriately padded with zeros. Further, each patch is flattened to a $1 \times k^2 d_{l-1}$ vector and the mean and variance are calculated from the resulting $NN_p \times k^2 d_{l-1}$ data matrix. The feature maps are then normalized as follows:

$$\hat{\mathbf{x}}_i^k = \frac{\mathbf{x}_i^k - \boldsymbol{\mu}_{\mathcal{B}}}{\sqrt{\boldsymbol{\sigma}_{\mathcal{B}}^2 + \epsilon}}, \tag{11}$$

$$\mathbf{y}_i^k = \gamma \hat{\mathbf{x}}_i^k + \beta, \tag{12}$$

where $\mathbf{x}_i^k$ is the i^{th} flattened patch, $\boldsymbol{\mu}_{\mathcal{B}}$ and $\boldsymbol{\sigma}_{\mathcal{B}}^2$ are the $1 \times k^2 d_{j-1}$-dimensional mean and variance vectors of the vectorized patches in the minibatch, and γ and β are the learnt parameters controlling the scale and offset, initialized as

[3]Least-squares regression to class labels can also be applied to initialize the last layer. However, we observed that in most cases random initialization of the output layer results in better generalization performance of the models during the subsequent training using backpropagation.

Algorithm 3: Initialization of l^{th} Convolutional layer

```
Function VectorBNorm($X^{(l)}$,$f\_size$):
    Input: $X^{(l)}$ : $N \times S1 \times S2 \times D$ data representation at $l^{th}$ layer; $f\_size$ : filter size
    $X^{(l)} \leftarrow$ Zero-pad $X^{(l)}$ to shape divisible by $f\_size$
    $X^{(l)}_{fl} \leftarrow$ Extract and vectorize all $f\_size \times f\_size$ non-overlapping patches from $X^{(l)}$
    $\mu \leftarrow$ mean($X^{(l)}_{fl}$); $\sigma \leftarrow$ var($X^{(l)}_{fl}$);
    for $patch \leftarrow$ iterate through $all\ non\text{-}overlapping\ f\_size \times f\_size\ patches\ in\ X^{(l)}$ do
        $patch \leftarrow \frac{flatten(patch)-\mu}{\sqrt{\sigma+\epsilon}}$
        $patch \leftarrow$ Reshape patch to $(f\_size \times f\_size \times D)$
    end
    return $X^{(l)}$
Function conv_init($X^{(l)}$,$y$,$N\_filt$, $f\_size$):
    Input: $X^{(l)}$ : $N \times S1 \times S2 \times D$ data representation at $l^{th}$ layer; $y$ : $N \times 1$
           vector with class labels; $N\_filt$ :number of filters; $f\_size$ :filter size;
    $Z \leftarrow$ ceil($N\_filt/C$)
    $X^{(l)} \leftarrow$ VectorBNorm($X^{(l)}, f\_size$)
    $X^{(l)}_{fl} \leftarrow$ Extract and vectorize all $f\_size \times f\_size$ non-overlapping patches from $X^{(l)}$
    $y_{cl} \leftarrow$ Cluster($X^{(l)}_{fl}, Z$)
    $T^{(l)} \leftarrow$ getTargets($y$,$y_{cl}$,$C$,$Z$,$N$,$D$)
    if $D < N$ then
        $R \leftarrow (\text{chol}(X^{(l)}_{fl} X^{(l)T}_{fl}))^{-1}$
        $W^{(l)} \leftarrow RR^T X^{(l)}_{fl} T^{(l)T}$
    else
        $R \leftarrow (\text{chol}(X^{(l)T}_{fl} X^{(l)}_{fl}))^{-1}$
        $W^{(l)} \leftarrow X^{(l)}_{fl} R^T R T^{(l)T}$
    end
    $W^{(l)} \leftarrow$Select first $N\_filt$ dimensions of $W^{(l)}$ and normalize with $l2$ norm
    $\hat{W}^{(l)} \leftarrow$Reshape $W^{(l)}$ to $(N\_filt, f\_size, f\_size, D)$
    return $\hat{W}^{(l)}$
```

1 and 0, respectively. Similarly to conventional Batch Normalization, moving mean and moving variance are estimated for normalization during inference.

4. Experimental setup

In order to evaluate the proposed network initialization approach, we ran experiments on three image classification datasets: CIFAR-10 [32], MNIST [32], and Linnaeus-5 [33]. CIFAR-10 dataset contains images of 32 × 32 pixels with 3 channels and 10 object categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. MNIST dataset contains grayscale images of size 28×28 posing a handwritten digit recognition problem. Linnaeus-5 dataset contains RGB images of 32×32 dimensionality of 5 object categories: berry, bird, dog, flower, and other. We use the provided train-test splits for evaluation. In CIFAR-10 dataset, training set is split into 48,000 images used for training, and 12,000 for validation. In MNIST dataset, training set is split into 40,000 and 10,000 images for training and validation, respectively. In both datasets, 10,000 images are used for testing. In Linneaus-5 datasets, 4,800 images are used for training, 1,200 for validation, and 2,000 for testing. Sample images from each dataset are shown in Fig. 1. Additionally, we employ two non-image datasets: CovType [34] and KDD [35]. CovType poses the task of classification of forest cover type from cartographic variables, having 7 classes with 54 attributes. We utilize 348,612 samples for training, and 116,200 for testing and validation. The KDD dataset poses the task of classification of network traffic into different types of attacks. The dataset has 23 classes and 41 attributes. We utilize a subset of 296,431 samples for training, and 98,795 for validation and testing.

Figure 1: Examples of dataset images from Linnaeus-5 (top), MNIST (middle) and CIFAR-10 (bottom) datasets.

In this work, we focus on settings that require small models, as well as on settings where large datasets might not be available, and hence data-dependant initialization is especially required for model performance. Experiments with initialization of deeper models are therefore left for future work. We evaluate our approach on two CNN architectures with 5 and 6 hidden layers, and MLPs with 4 and 5 hidden layers. Recall that following the proposed methodology, the maximum number of neurons in MLPs or filters in CNNs at a certain layer is equal to $\sum_{i=1}^{C} K_i$, where K_i is the number of subclasses for class i. We set $K_i = Z$ subclasses for all classes, leading to $CZ-1$ neurons in MLPs or filters in CNNs at a certain layer. In our experimental setup we construct the networks starting from 16 or 32 subclasses and reducing the number of subclasses by a factor of 2 with each subsequent layer. This results in two architectures with the layers having width of$\{319, 159, 79, 39, 19\}$ or $\{159, 79, 39, 19\}$ neurons/filters for MNIST and CIFAR datasets, and $\{159, 79, 39, 19, 9\}$ or $\{79, 39, 19, 9\}$ neurons/filters for Linnaeus-5 dataset. In CNN case, another fully-connected layer of 128 neurons is added after the last convolutional layer, initialized following Algorithm 2. The output layer consists of 5 or 10 neurons depending on the dataset, and a softmax activation function.

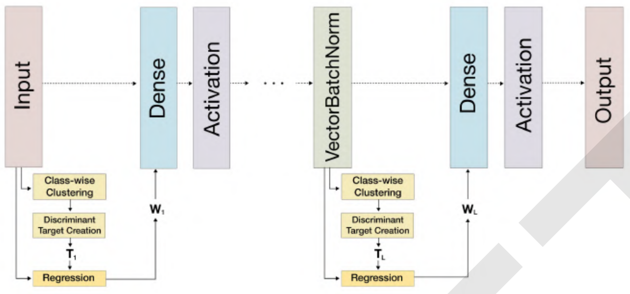


Figure 2: Structure of fastSDA-initialized Dense network

The overall architecture structure for MLPs is outlined in Fig. 2. We apply an activation function after each Dense layer, and a Batch Normalization layer before each Dense layer except the output layer (assuming the input data is standardized). The overall structure of the CNN architectures is shown in Fig. 3. We apply a Vector Batch Normalization layer before every convolution layer, followed by Max Pooling and Activation. After the last convolutional block, data is flattened and Batch Normalization is applied, followed by a Dense layer with 128 neurons, an activation function, and an output layer with softmax activation. For all the networks we perform experiments with three commonly-used activation functions: ReLU, LeakyReLU with α=0.3, and Tanh (hyperbolic tangent). The output layer is initialized randomly from a Gaussian distribution with zero mean and standard deviation equal to 0.05. In CNN, the bias terms are omitted in all models, and in MLPs they are initialized from zeros. To obtain the cluster labels during fastSDA initialization, mini-batch k-means clustering is performed [36].

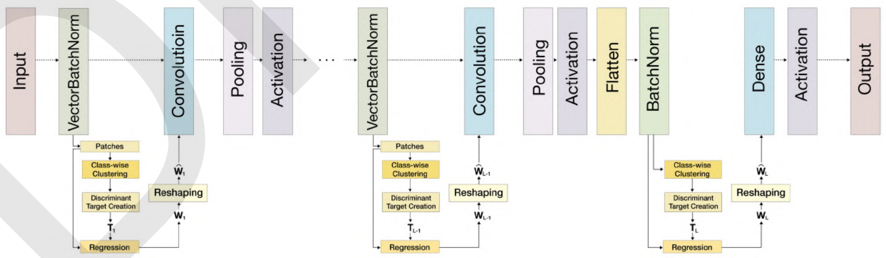


Figure 3: Structure of fastSDA-initialized CNN

In MLPs we compare the proposed initialization approach with random initialization from Gaussian distribution with $\mu = 0$ and $\sigma = 0.05$ (RNorm), random initialization from uniform distribution in the range $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ (RUni), where n is the number of input neurons in the corresponding layers. We also provide comparisons with Glorot initialization [8] and He initialization [9]. We also compare the results with data-driven approaches by substituting the fastSDA step with either K-Means initialization (KM), LDA, or PCA. For K-Means initialization, we whiten the data and apply spherical clustering into n clusters, subsequently initializing each neuron with one of the cluster centroids following [37]. In LDA and PCA initialization, we initialize the neurons to the eigenvectors of the corresponding weight matrices, similarly to [15, 24]. Since LDA and PCA can return at maximum $C-1$ and D eigenvectors, respectively, in the case that the number of eigenvectors corresponding to non-zero eigenvalues are lower than the number of neurons required by the architecture, we initialize them randomly from a Gaussian distribution with zero mean and standard deviation of 0.05. In LDA and PCA, eigenvector matrices are normalized such that the $\updownarrow - 2$ norm of each column is equal to 1, similarly to the proposed approach, to ensure that any difference in performance arises from the utilized statistical learning method rather than from normalization. The output layers are initialized randomly, similarly to our proposed approach. All the initialization methods are evaluated on the same architectures as the proposed approach.

Similarly, in CNN, we compare the proposed initialization approach with Glorot initialization [8], He initialization [9], random Gaussian and random uniform distributions with the parameters similar to the ones utilized in MLPs, K-Means initialization, and PCA initialization. We use the same architecture as shown in Fig. 3 for all initialization methods. Besides, for random initializations, He, and Glorot methods we provide the results for the architectures where Vector Batch Normalization is replaced with conventional Batch Normalization, to ensure that the accuracy gain obtained with the proposed approach does not result solely from the new normalization layer.

It can be noted that the patch extraction in the initialization of CNN results in a significant increase in the number of data samples used to learn the projection space, which might lead to undesired overhead during the clustering step of the proposed approach. As a remedy for this, we show that a small number of samples is generally sufficient to learn a good projection space that leads to competitive performance. To showcase this, we provide the results in which only a limited number of training samples is used during the initialization step. Specifically, we test the proposed approach with 200 and 500 samples per class (i.e., the total of 2000 or 5000 samples in CIFAR-10 and MNIST, and 1000 or 2500 samples in Linnaeus-5 dataset). Besides, we also evaluate the methods without utilization of any type of Batch Normalization. In this case, normalization of data is also not performed during learning of the projection space initialization of the weights, and the solution is, therefore, approximate.

We train the models with Stochastic Gradient Descent with a learning rate of 0.001, a batch size of 32, and categorical cross-entropy as the loss function until the accuracy on the validation set stops improving for 10 epochs. The model that resulted in the best validation accuracy is then used for reporting the results on the test set. Data in MLP experiments is standardized and images in CNN experiments are mean-centered and rescaled to match the range of 0 to

Table 1: Accuracies on MLP architectures

Linnaeus-5

	LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
RNorm	37.25	31.30	30.40	29.55	32.50	33.60
RUni	35.05	35.25	31.35	30.95	30.95	32.65
He	36.70	32.45	32.95	38.55	29.65	32.00
Glorot	**39.55**	31.35	33.90	37.00	30.60	36.25
KM	38.65	34.70	35.60	**40.75**	31.00	32.40
LDA	32.65	32.70	33.00	37.25	31.40	33.00
PCA	32.95	34.60	33.90	33.30	31.90	35.65
fSDA	39.10	38.10	36.80	38.60	**40.35**	36.50
fSDA$_{500}$	37.05	**38.45**	**37.00**	37.80	37.80	**37.05**
fSDA$_{200}$	36.30	35.30	34.40	35.20	30.20	34.05

CIFAR-10

	LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
RNorm	47.46	45.69	39.78	47.10	43.28	38.36
RUni	45.68	42.09	36.77	45.42	42.49	38.12
He	47.26	43.89	40.30	46.72	41.65	39.00
Glorot	47.16	44.61	42.21	47.14	41.66	40.86
KM	47.15	45.81	42.35	48.47	45.44	41.81
LDA	46.09	44.59	40.44	46.44	43.46	39.69
PCA	**48.85**	45.04	40.66	47.29	42.91	40.24
fSDA	48.20	46.32	**44.51**	47.97	48.18	**43.49**
fSDA$_{500}$	48.56	**47.32**	43.65	**48.77**	**48.48**	42.66
fSDA$_{200}$	47.55	46.19	43.55	46.93	46.40	42.62

MNIST

	LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
RNorm	96.54	96.44	95.78	96.42	96.26	96.06
RUni	95.85	95.11	93.59	96.33	95.70	94.46
He	96.33	96.06	95.44	96.22	95.18	95.29
Glorot	96.85	96.38	96.15	96.55	96.13	95.80
KM	96.52	96.38	96.17	96.76	96.25	96.29
LDA	96.12	95.75	95.98	96.44	95.79	95.71
PCA	96.72	96.44	96.11	96.59	96.10	95.67
fSDA	96.85	96.56	**96.34**	96.72	96.70	96.35
fSDA$_{500}$	96.81	**96.89**	**96.34**	96.95	**97.29**	96.53
fSDA$_{200}$	**97.22**	96.72	95.92	**97.12**	96.68	**96.83**

1.

4.1. Results

The accuracy for MLP models with different initialization methods is shown in Table 1 and Table 2, where we report results on three activation functions and two architectures, i.e., LReLU$_{16}$ stands for architecture corresponding to 16 subclasses as described earlier and Leaky ReLU activation function. Similarly, Tables 3 and 4 show the accuracies for CNNs without and with normalization layers, respectively. The best accuracy is highlighted in bold.

Table 2: Classification results of linear methods in COVTYPE and KDD datasets

COVTYPE

	ReLU16	LReLU16	Tanh16	LReLU32	ReLU32	Tanh32
RNorm	58.81	60.11	63.65	57.35	52.08	52.14
RUni	59.19	59.25	60.37	54.08	59.90	56.33
He	59.61	61.48	59.15	54.84	52.22	54.37
Glorot	63.42	59.96	63.88	53.68	51.29	53.71
KM	59.80	63.01	61.46	55.91	59.35	55.05
LDA	60.68	60.20	**65.07**	53.42	56.01	54.27
PCA	59.89	59.24	64.13	52.51	56.10	56.91
fSDA	**63.97**	57.42	63.14	**57.65**	**60.83**	54.60
fSDA500	61.39	**63.84**	62.58	51.54	57.06	55.73
fSDA200	63.66	62.65	63.13	56.84	53.29	**58.04**

KDD

	ReLU16	LReLU16	Tanh16	LReLU32	ReLU32	Tanh32
RNorm	97.10	95.88	96.79	98.96	97.88	99.13
RUni	98.48	96.75	98.34	97.73	97.52	99.05
He	90.29	94.28	97.92	97.48	97.70	**99.20**
Glorot	96.98	96.78	98.31	97.62	97.95	9823
KM	96.88	95.99	**98.68**	98.43	97.70	98.00
LDA	97.12	96.05	98.36	97.92	98.12	98.94
PCA	97.19	96.63	97.73	98.85	96.64	99.17
fSDA	96.62	96.86	97.56	98.18	97.31	98.87
fSDA500	**98.73**	**98.08**	96.96	**99.17**	**98.42**	98.05
fSDA200	97.30	96.68	98.50	99.03	97.67	98.46

Table 3: CNN results without normalization

Linnaeus-5

	LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
RNorm	63.05	55.70	60.85	57.40	60.00	59.10
RUni	48.15	50.95	54.65	52.55	50.40	56.30
He	55.70	55.95	61.85	58.85	56.15	58.60
Glorot	61.85	**61.75**	60.40	61.50	60.90	62.10
KM	61.90	52.00	46.50	63.25	63.85	45.45
PCA	64.50	54.05	59.35	61.30	**64.70**	62.15
fSDA	**64.75**	48.85	**62.10**	**64.55**	61.90	**63.55**

CIFAR-10

	LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
RNorm	68.54	68.29	67.08	66.35	65.40	65.67
RUni	64.02	62.02	66.53	69.01	68.65	70.92
He	68.33	67.75	70.68	68.77	69.25	**72.61**
Glorot	70.27	**70.49**	71.00	71.01	70.93	71.28
KM	70.46	69.75	70.72	72.25	71.41	70.70
PCA	70.31	70.20	70.70	71.87	70.62	71.10
fSDA	**71.10**	69.83	**71.01**	**72.66**	**71.52**	70.46

MNIST

	LReLU$_{16}$	ReLU$_{16}$	Tanh$_{16}$	LReLU$_{32}$	ReLU$_{32}$	Tanh$_{32}$
RNorm	98.99	98.96	99.23	98.81	98.98	99.18
RUni	98.78	99.01	99.13	98.93	99.05	**99.35**
He	98.85	98.92	99.24	98.91	98.86	99.30
Glorot	98.79	98.79	99.09	99.00	98.68	99.24
KM	99.03	99.02	99.21	99.01	99.14	99.10
PCA	99.01	**99.13**	99.27	98.97	99.08	99.18
fSDA	**99.05**	99.03	**99.28**	**99.08**	**99.17**	99.26

As can be seen from Table 1, in the majority of architectures and datasets, the proposed initialization outperforms other competing methods in terms of accuracy. In the CNN scenario, the proposed approach often outperforms competing methods already without considering mean-centering and the use of any type of normalization layers. We can see that mean-centering of the data during the initialization and the subsequent use of VectorBatchNormalization layers result in improved accuracy even further in the vast majority of the scenarios. Note that such normalization also leads to improved accuracy of PCA and K-Means initialization in most of the cases.

Considering the initialization using smaller number of samples, we observe that in the CNNs, both 200 and 500 samples are often sufficient for outperforming the competing methods (in the case fSDA$_{200}$ or fSDA$_{500}$ outperforms competing methods except fSDA, it is underlined in the tables). Considering the MLP initialization, the results with regard to initialization with a smaller number of samples are rather similar to that of CNN and the use of a small number of samples generally leads to a fair performance. Another fact worth

Table 4: CNN results with normalization layers

Linnaeus-5

		$LReLU_{16}$	$ReLU_{16}$	$Tanh_{16}$	$LReLU_{32}$	$ReLU_{32}$	$Tanh_{32}$
BNorm	RNorm	52.55	51.45	48.80	51.75	50.10	46.15
	RUni	59.00	55.25	49.65	58.50	58.30	47.70
	He	55.00	52.20	50.55	57.30	53.95	50.10
	Glorot	54.50	54.95	53.00	57.20	56.35	52.60
VecBNorm	RNorm	49.55	47.05	44.85	50.80	47.70	41.85
	RUni	53.90	51.55	44.30	51.85	52.90	43.80
	He	55.15	53.40	50.75	57.20	54.10	51.15
	Glorot	56.40	52.90	53.00	58.85	57.05	52.90
	KM	60.70	62.40	60.85	61.55	58.70	57.65
	PCA	60.90	62.90	60.00	**63.90**	60.35	59.75
	fSDA	**64.25**	62.30	61.75	59.75	62.70	**61.75**
	$fSDA_{500}$	62.00	**64.35**	**62.10**	61.45	**64.20**	61.70
	$fSDA_{200}$	60.65	62.90	59.70	60.95	64.05	59.20

CIFAR-10

		$LReLU_{16}$	$ReLU_{16}$	$Tanh_{16}$	$LReLU_{32}$	$ReLU_{32}$	$Tanh_{32}$
BNorm	RNorm	66.17	63.67	59.35	64.89	62.05	63.78
	RUni	69.50	69.38	62.32	71.66	70.23	64.88
	He	68.49	68.74	65.89	70.18	71.34	64.57
	Glorot	71.71	72.04	68.41	73.77	73.51	66.94
VecBNorm	RNorm	63.65	61.89	59.96	64.50	60.68	62.71
	RUni	64.99	65.71	54.70	67.31	66.65	56.32
	He	69.17	68.17	64.24	71.54	70.11	64.88
	Glorot	71.45	71.75	65.79	73.73	72.88	68.56
	KM	72.03	75.01	68.52	**77.18**	76.20	67.03
	PCA	72.67	74.40	68.06	72.71	77.17	71.65
	fSDA	**75.02**	**75.36**	**70.59**	76.66	**77.79**	71.87
	$fSDA_{500}$	74.13	74.35	69.80	71.29	76.22	**72.60**
	$fSDA_{200}$	70.32	74.57	69.35	75.71	77.33	71.39

MNIST

		$LReLU_{16}$	$ReLU_{16}$	$Tanh_{16}$	$LReLU_{32}$	$ReLU_{32}$	$Tanh_{32}$
BNorm	RNorm	98.82	98.78	98.45	98.94	98.70	98.41
	RUni	99.16	99.20	99.03	99.11	99.19	99.04
	He	99.00	99.17	99.03	99.19	99.30	99.10
	Glorot	99.10	99.30	99.23	99.22	**99.35**	99.24
VecBNorm	RNorm	98.76	98.35	98.30	98.76	98.63	98.19
	RUni	98.99	98.81	98.49	99.10	98.92	98.58
	He	99.03	99.14	98.95	99.13	99.14	98.87
	Glorot	**99.18**	99.21	99.08	99.15	99.16	99.22
	KM	99.10	99.07	99.12	99.21	99.20	99.18
	PCA	98.82	99.18	99.20	**99.25**	99.24	**99.34**
	fSDA	98.67	**99.26**	**99.24**	**99.25**	99.24	99.28
	$fSDA_{500}$	98.98	99.14	99.17	97.92	99.13	99.10
	$fSDA_{200}$	98.65	99.04	95.16	99.17	99.18	99.05

noticing is that in a few cases, the use of a smaller number of samples leads to performance improved compared to using the full dataset. A possible interpretation of this is that the model trained on a smaller number of samples overfits less to the training data, thus providing better generalization properties.

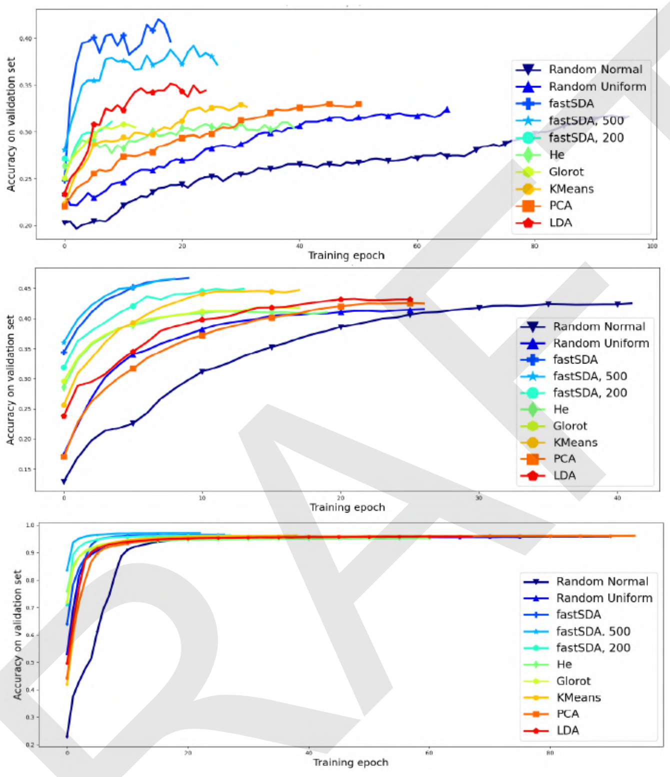


Figure 4: Convergence plots on MLPs. Datasets top to bottom: Linnaeus-5, CIFAR-10, MNIST

Figures 4 and 5 show the convergence speed of different methods, where we plot the accuracy on the validation set versus the number of training epochs. For the sake of variety, we provide the results on architectures corresponding to 32 subclasses and ReLU activation function for MLP architectures, and 16 subclasses and LeakyReLU activation function for CNN architecture. The plots outline several essential points: we observe that fastSDA-initialized models generally start from a higher accuracy compared to other methods, and generally they also take less epochs to converge. This is clearly seen especially on the MLP architectures. In addition, we can see that utilization of a larger number of samples for initialization results in a higher initial accuracy and a faster convergence compared to using a smaller number of samples. In CNN, we observe that the convergence properties are not as good as in the MLP case, and our proposed methods are mostly doing on-par with competing ones. However, this

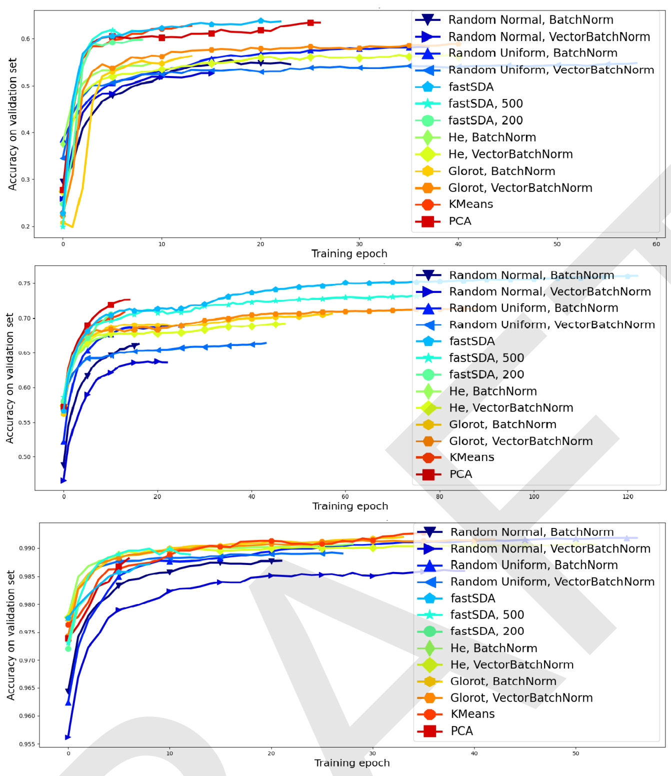


Figure 5: Convergence plots on CNNs. Datasets top to bottom: Linnaeus-5, CIFAR-10, MNIST

is compensated by the fact that our methods are able to achieve a better overall accuracy, and a more detailed investigation on the convergence properties of CNNs is left as a future work. Overall, these observations support our intuition that fastSDA initialization allows to start the optimization process from a more favourable point in the feature space.

For reference, we provide the initialization times in seconds for larger architecture corresponding to 32 subclasses for MLPs and CNNs in Table 5. As can be seen, the speed of initialization depends both on dimensionality and dataset size (recall that MNIST has 1 channel unlike CIFAR-10 and Linnaeus-5 that have 3 channels, and Linnaeus-5 is the smallest dataset). In MLPs, the overhead created by clustering plays a bigger role compared to dimensionality of data, leading to fastSDA with full training data being slower than PCA. However, in CNN and when using a smaller number of images for initialization, our approach is generally faster.

Table 5: Times for initialization in 32-subclass architecture (seconds)

MLP

	KM	LDA	PCA	fSDA	fSDA$_{500}$	fSDA$_{200}$
CIFAR	108	1251	25	108	42	26
MNIST	73	68	3	41	11	8
LIN	23	176	23	25	16	8

CNN

	KM	PCA	fSDA	fSDA$_{500}$	fSDA$_{200}$
CIFAR	22020	12364	6315	807	532
MNIST	16301	6158	4516	521	294
LIN	958	1208	369	216	152

5. Conclusion

In this paper we proposed a novel data-driven approach for weight initialization based on discriminant learning. The proposed initialization was formulated for dense and convolutional layers appearing in Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs). In addition, we considered some of the limitations of the method caused by assumptions on the data and proposed ways to remedy them. Experimental results show that the proposed approach provides several benefits compared to competing ones, including improved training accuracy and initial accuracy, while achieving equal or faster convergence and initialization time. In addition, we showed that the initialization time can be improved even further by applying the initialization based on a small number of samples with no degrading effect on accuracy.

Acknowledgment

This work is supported by Business Finland under project 5G Vertical Integrated Industry for Massive Automation (5G-VIIMA). A. Iosifidis acknowledges funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337 (MARVEL).

References

[1] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, Q. Tian, Centernet: Keypoint triplets for object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6569–6578.

[2] Z.-Q. Zhao, P. Zheng, S.-t. Xu, X. Wu, Object detection with deep learning: A review, IEEE transactions on neural networks and learning systems 30 (11) (2019) 3212–3232.

[3] K. Chumachenko, A. Männistö, A. Iosifidis, J. Raitoharju, Machine learning based analysis of finnish world war ii photographers, IEEE Access 8 (2020) 144184–144196.

[4] S. Yun, J. Choi, Y. Yoo, K. Yun, J. Y. Choi, Action-driven visual object tracking with deep reinforcement learning, IEEE Transactions on Neural Networks and Learning Systems 29 (6) (2018) 2239–2252.

[5] C. Li, W. Xia, Y. Yan, B. Luo, J. Tang, Segmenting objects in day and night: Edge-conditioned cnn for thermal image semantic segmentation, IEEE Transactions on Neural Networks and Learning Systems (2020).

[6] X. Chen, J. Weng, W. Lu, J. Xu, J. Weng, Deep manifold learning combined with convolutional neural networks for action recognition, IEEE transactions on neural networks and learning systems 29 (9) (2017) 3938–3952.

[7] A. Iosifidis, A. Tefas, I. Pitas, "view-invariant action recognition based on artificial neural networks, IEEE Transactions on Neural Networks and Learning Systems 23 (3) (2012) 412–424.

[8] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.

[9] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[10] P. Krähenbühl, C. Doersch, J. Donahue, T. Darrell, Data-dependent initializations of convolutional neural networks, arXiv preprint arXiv:1511.06856 (2015).

[11] C.-C. J. Kuo, M. Zhang, S. Li, J. Duan, Y. Chen, Interpretable convolutional neural networks via feedforward design, Journal of Visual Communication and Image Representation 60 (2019) 346–359.

[12] Y. Chen, Y. Yang, M. Zhang, C.-C. J. Kuo, Semi-supervised learning via feedforward-designed convolutional neural networks, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 365–369.

[13] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, Y. Ma, Pcanet: A simple deep learning baseline for image classification?, IEEE transactions on image processing 24 (12) (2015) 5017–5032.

[14] Y. Ge, J. Hu, W. Deng, Pca-ldanet: A simple feature learning method for image classification, in: 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR), IEEE, 2017, pp. 370–375.

[15] M. Alberti, M. Seuret, V. Pondenkandath, R. Ingold, M. Liwicki, Historical document image segmentation with lda-initialized deep neural networks, in: Proceedings of the 4th International Workshop on Historical Document Imaging and Processing, 2017, pp. 95–100.

[16] R. Duda, P. Hart, D. Stork, Pattern Classification, 2nd Edition, Wiley, New York, NY, USA, 2000.

[17] R. O. Duda, P. E. Hart, D. G. Stork, Pattern classification, John Wiley & Sons, 2012.

[18] D. Wang, M. Li, Stochastic configuration networks: Fundamentals and algorithms, IEEE transactions on cybernetics 47 (10) (2017) 3466–3479.

[19] M. Rosenstein, Z. Marx, L. Kaelbling, T. Dietterich, To transfer or not to transfer, in: Neural Information Processing Workshop on Transfer Learning, 2005, pp. 1–4.

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

[21] K. He, R. Girshick, P. Dollár, Rethinking imagenet pre-training, in: Proceedings of the IEEE international conference on computer vision, 2019, pp. 4918–4927.

[22] A. Coates, A. Y. Ng, Learning feature representations with k-means, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 561–580.

[23] J. Mairal, P. Koniusz, Z. Harchaoui, C. Schmid, Convolutional kernel networks, in: Advances in neural information processing systems, 2014, pp. 2627–2635.

[24] M. Seuret, M. Alberti, M. Liwicki, R. Ingold, Pca-initialized deep neural networks applied to document image analysis, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Vol. 01, 2017, pp. 877–882.

[25] D. Cai, X. He, J. Han, Training linear discriminant analysis in linear time, in: 2008 IEEE 24th International Conference on Data Engineering, IEEE, 2008, pp. 209–217.

[26] K. Chumachenko, J. Raitoharju, A. Iosifidis, M. Gabbouj, Speed-up and multi-view extensions to subclass discriminant analysis, Pattern Recognition 111 (107660) (2020) 1–15.

[27] M. Zhu, A. Martinez, Subclass discriminant analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006).

[28] Y. Shuicheng, X. Dong, B. Zhang, H. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 40–51.

[29] D. Cai, X. He, J. Han, Srda: an efficient algorithm for large scale discriminant analysis, IEEE Transactions on Knowledge and Data Engineering 20 (2007) 1–12.

[30] K. Chumachenko, M. Gabbouj, A. Iosifidis, Robust fast subclass discriminant analysis, in: European Signal Processing Conference, 2020.

[31] K. Chumachenko, J. Raitoharju, M. Gabbouj, A. Iosifidis, Incremental fast subclass discriminant analysis, in: International Conference on Image Processing, 2020.

[32] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).

[33] G. Chaladze, L. Kalatozishvili, Linnaeus 5 dataset for machine learning, Tech. rep., Tech. Rep (2017).

[34] J. A. Blackard, D. J. Dean, Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables, Computers and electronics in agriculture 24 (3) (1999) 131–151.

[35] J. Stolfo, W. Fan, W. Lee, A. Prodromidis, P. K. Chan, Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection, Results from the JAM Project by Salvatore (2000) 1–15.

[36] D. Sculley, Web-scale k-means clustering, in: Proceedings of the 19th international conference on World wide web, 2010, pp. 1177–1178.

[37] A. Coates, A. Y. Ng, Learning feature representations with k-means, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 561–580.

7.13 Within-layer Diversity Reduces Generalization Gap

The appended paper follows.

Within-layer Diversity Reduces Generalization Gap

Firas Laakom [1] Jenni Raitoharju [2] Alexandros Iosifidis [3] Moncef Gabbouj [1]

Abstract

Neural networks are composed of multiple layers arranged in a hierarchical structure jointly trained with a gradient-based optimization. At each optimization step, neurons at a given layer receive feedback from neurons belonging to higher layers of the hierarchy. In this paper, we propose to complement this traditional 'between-layer' feedback with additional 'within-layer' feedback to encourage diversity of the activations within the same layer. To this end, we measure the pairwise similarity between the outputs of the neurons and use it to model the layer's overall diversity. By penalizing similarities and promoting diversity, we encourage each neuron to learn a distinctive representation and, thus, to enrich the data representation learned within the layer and to increase the total capacity of the model. We theoretically and empirically study how the within-layer activation diversity affects the generalization performance of a neural network and prove that increasing the diversity of hidden activations reduces the generalization gap.

1. Introduction

Neural networks are a powerful class of non-linear function approximators that have been successfully used to tackle a wide range of problems. They have enabled breakthroughs in many tasks, such as image classification (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012a), and anomaly detection (Golan & El-Yaniv, 2018). However, neural networks are often over-parameterized, i.e., have more parameters than data. As a result, they tend to overfit to the training samples and not generalize well on unseen examples (Goodfellow et al., 2016). While research on double descent (Belkin et al., 2019; Advani et al., 2020; Nakkiran et al., 2020) shows that over-parameterization does not necessarily lead to overfitting, avoiding overfitting has been extensively studied (Neyshabur et al., 2018; Nagarajan & Kolter, 2019; Poggio et al., 2017) and various approaches and strategies have been proposed, such as data augmentation (Goodfellow et al., 2016; Zhang et al., 2018), regularization (Kukačka et al., 2017; Bietti et al., 2019; Arora et al., 2019), and dropout (Hinton et al., 2012b; Wang et al., 2019; Lee et al., 2019; Li et al., 2016), to close the gap between the empirical loss and the expected loss.

Diversity of learners is widely known to be important in ensemble learning (Li et al., 2012; Yu et al., 2011) and, particularly in deep learning context, diversity of information extracted by the network neurons has been recognized as a viable way to improve generalization (Xie et al., 2017a; 2015). In most cases, these efforts have focused on making the set of weights more diverse (Yang et al.; Malkin & Bilmes, 2009). However, diversity of the activations has not received much attention.

To the best of our knowledge, (Cogswell et al., 2016) is the only work in neural network context which considers diversity of the activations directly. They propose an additional loss term using cross-covariance of hidden activations, which encourages the neurons to learn diverse or non-redundant representations. The proposed approach, known as DeCov, has empirically been proven to alleviate overfitting and to improve the generalization ability of neural network, yet a theoretical analysis to prove this has so far been lacking. Moreover, modeling diversity as the sum of the pair-wise cross-covariance, it can capture only the pairwise diversity between components and is unable to capture the higher-order "diversity".

In this work, we start by theoretically showing that the within-layer activation diversity boosts the generalization performance of neural networks and reduces overfitting. Moreover, we propose a novel approach to encourage activation diversity within a layer. We propose complementing the 'between-layer' feedback with additional 'within-layer' feedback to penalize similarities between neurons on the same layer. Thus, we encourage each neuron to learn a distinctive representation and to enrich the data representation

[1]Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland [2]Programme for Environmental Information, Finnish Environment Institute, Jyväskylä, Finland [3]Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark. Correspondence to: Firas Laakom <firas.laakom@tuni.fi>.

Presented at the ICML 2021 Workshop on Information-Theoretic Methods for Rigorous, Responsible, and Reliable Machine Learning. Copyright 2021 by the author(s).

learned within each layer. We propose three variants for our approach that are based on different global diversity definitions.

Our contributions in this paper are as follows:

- Theoretically, we analyse the effect of the within-layer activation diversity on the generalization error bound of neural network. As shown in Theorems 1-6, we express the upper-bound of the estimation error as a function of the diversity factor. Thus, we provide theoretical evidence that the within-layer activation diversity can help reduce the generalization error.
- Methodologically, we propose a new approach to encourage the 'diversification' of the layers' output feature maps in neural networks. The main intuition is that by promoting the within-layer activation diversity, neurons within a layer learn distinct patterns and, thus, increase the overall capacity of the model.
- Empirically, we show that the proposed within-layer activation diversification boosts the performance of neural networks.

2. Generalization error analysis

In this section, we analyze how the within-layer activation diversity affects the generalization error of a neural network. Generalization theory (Zhang et al., 2017; Kawaguchi et al., 2017) focuses on the relation between the empirical loss and the expected risk defined as follows:

$$L(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{Q}}[l(f(\boldsymbol{x}), y)], \tag{1}$$

where $\mathcal{Q}$ is the underlying distribution of the dataset. Let $f^* = \arg\min_f L(f)$ be the expected risk minimizer and $\hat{f} = \arg\min_f \hat{L}(f)$ be the empirical risk minimizer. We are interested in the estimation error, i.e., $L(f^*) - L(\hat{f})$, defined as the gap in the loss between both minimizers (Barron, 1994). The estimation error represents how well an algorithm can learn. It usually depends on the complexity of the hypothesis class and the number of training samples (Barron, 1993; Zhai & Wang, 2018).

In this work, we are interested in the effect of the within-layer activation diversity on the estimation error. In order to study this effect, we assume that with a high probability τ, the distance between the output of each pair of neurons, $(\phi_n(\boldsymbol{x}) - \phi_m(\boldsymbol{x}))^2$, is lower bounded by d_{min}^2 for any input $\boldsymbol{x}$. Intuitively, if two neurons n and m have similar outputs for many samples, their corresponding similarity d_{min} will be small. Otherwise, their similarity d_{min} is small and they are considered "diverse". By studying how d_{min} affects the generalization of the model, we can theoretically understand how diversity affects the performance of neural networks. To this end, we derive generalization bounds for neural networks using d_{min}.

Several techniques have been used to quantify the estimation error, such as PAC learning (Hanneke, 2016; Arora et al., 2018), VC dimension (Sontag, 1998; Harvey et al., 2017; Bartlett et al., 2019), and the Rademacher complexity (Xie et al., 2015; Zhai & Wang, 2018; Tang et al., 2020). The Rademacher complexity has been widely used as it usually leads to a tighter generalization error bound (Sokolic et al., 2016; Neyshabur et al., 2018; Golowich et al., 2018). In this work, we also rely on the Rademacher complexity to study diversity. We seek a tighter upper bound of the estimation error and show how the within-layer diversity, expressed with d_{min}, affects the bound. We start by deriving such an upper-bound for a simple network with one hidden layer trained for a regression task and then we extend it for a general multi-layer network and for different losses. The proofs are provided as supplementary material.

2.1. Single hidden-layer network

Here, we consider a simple neural network with one hidden-layer with M neurons and one-dimensional output trained for a regression task. The full theoretical characterization of the setup can be summarized in the assumptions presented in Appendix 6.1

Our main goal is to analyze the estimation error bound of the neural network and to see how its upper-bound is linked to the diversity, expressed by d_{min}, of the different neurons. The main result is presented in Theorem 1.

Theorem 1. *Under Assumptions 1, there exist a constant A, such that with probability at least $\tau^{\mathcal{Q}}(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq \Big(\sqrt{\mathcal{J}} + C_2\Big)A + \frac{1}{2}(\sqrt{\mathcal{J}} + C_2)^2\sqrt{\frac{2\log(2/\delta)}{N}}, \tag{2}$$

where $\mathcal{J} = C_4^2\big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\big)$ and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Theorem 1 provides an upper-bound for the estimation error. We note that it is a decreasing function of d_{min}. Thus, we say that a higher d_{min}, i.e., more diverse activations, yields a lower estimation error bound. In other words, by promoting the within-layer diversity, we can reduce the generalization error of neural networks.

2.2. Binary classification

We now extend our analysis of the effect of the within-layer diversity on the generalization error in the case of a binary classification task, i.e., $y \in \{-1, 1\}$. The extensions of Theorem 1 in the case of a hinge loss and a logistic loss are presented in Theorems 2 and 3, respectively.

Theorem 2. *Using the hinge loss, there exist a constant A, such that with probability at least $\tau^Q(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq A + (1+\sqrt{\mathcal{J}})\sqrt{\frac{2\log(2/\delta)}{N}}, \quad (3)$$

where $\mathcal{J} = C_4^2\big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\big)$ and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Theorem 3. *Using the logistic loss $l(f(x), y) = \log(1 + \boldsymbol{e}^{-yf(x)})$, there exist a constant A such that, with probability at least $\tau^Q(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq \frac{A}{1+\boldsymbol{e}^{\sqrt{-\mathcal{J}}}} + \log(1+\boldsymbol{e}^{\sqrt{\mathcal{J}}})\sqrt{\frac{2\log(2/\delta)}{N}}, \quad (4)$$

where $\mathcal{J} = C_4^2\big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\big)$ and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

As we can see, also for the binary classification task, the error bounds of the estimation error for the hinge and logistic losses are decreasing with respect to d_{min}. Thus, employing a diversity strategy can improve the generalization also for the binary classification task.

2.3. Multi-layer networks

Here, we extend our result for networks with P (> 1) hidden layers. We assume that the pair-wise distances between the activations within layer p are lower-bounded by d_{min}^p with a probability τ^p. In this case, the main theorem is extended as follows:

Theorem 4. *There exist a constant A such that, with probability of at least $\prod_{p=0}^{P-1}(\tau^p)^{Q^p}(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq (\sqrt{\mathcal{J}^P} + C_2)A + \frac{1}{2}\left(\sqrt{\mathcal{J}^P} + C_2\right)^2\sqrt{\frac{2\log(2/\delta)}{N}}, \quad (5)$$

where Q^p is the number of neuron pairs in the p^{th} layer, defined as $Q^p = \frac{M^p(M^p-1)}{2}$, and $\mathcal{J}^P$ is defined recursively using the following identities: $\mathcal{J}^0 = C_3^0 C_1$ and $\mathcal{J}^p = M^p C^{p2}\big(M^{p2}(L_\phi C_3^{p-1}\mathcal{J}^{p-1} + \phi(0))^2 - M(M-1)\frac{{d_{min}^p}^2}{2}\big)$, for $p = 1, \ldots, P$.

In Theorem 4, we see that $\mathcal{J}^P$ is decreasing with respect to d_{min}^p. Thus, we see that maximizing the within-layer diversity, we can reduce the estimation error of a multi-layer neural network.

2.4. Multiple outputs

Finally, we consider the case of a neural network with a multi-dimensional output, i.e., $\boldsymbol{y} \in R^D$. This yields the following two theorems:

Theorem 5. *For a multivariate regression trained with the squared error, there exist a constant A such that, with probability at least $\tau^Q(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq (\sqrt{\mathcal{J}} + C_2)A + \frac{D}{2}(\sqrt{\mathcal{J}} + C_2)^2\sqrt{\frac{2\log(2/\delta)}{N}}, \quad (6)$$

where $\mathcal{J} = C_4^2\big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\big)$ and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Theorem 6. *For a multi-class classification task using the cross-entropy loss, there exist a constant A such that, with probability at least $\tau^Q(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq \frac{A}{D-1+\boldsymbol{e}^{-2\sqrt{\mathcal{J}}}} + \log\left(1 + (D-1)\boldsymbol{e}^{2\sqrt{\mathcal{J}}}\right)\sqrt{\frac{2\log(2/\delta)}{N}}, \quad (7)$$

where $\mathcal{J} = C_4^2\big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\big)$ and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Theorems 5 and 6 extend our result for the multi-dimensional regression and classification tasks, respectively. Both bounds are inversely proportional to the diversity factor d_{min}. We note that for the classification task, the upper-bound is exponentially decreasing with respect to d_{min}. This shows that increasing diversity within the layer yields a tighter generalization gap and, thus, theoretically guarantees a stronger generalization performance.

3. Within-layer activation diversity

As shown in the previous section, promoting diversity of activations within a layer can lead to tighter generalization bound and can theoretically decrease the gap between the empirical and the true risks. In this section, we propose a novel diversification strategy, where we encourage neurons within a layer to activate in a mutually different manner, i.e., to capture different patterns. To this end, we propose an additional within-layer loss which penalizes the neurons that activate similarly. The standard loss function $\hat{L}(f)$ is augmented as follows: $\hat{L}_{aug}(f) = \hat{L}(f) + \lambda\sum_{i=1}^{P} J^i$, where J^i expresses the overall similarity of the neurons within the i^{th} layer and λ is the penalty coefficient for the diversity loss. Our proposed diversity loss can be applied to a single layer or multiple layers in a network. For simplicity, let us focus on a single layer.

Let $\phi_n^i(\boldsymbol{x}_j)$ and $\phi_m^i(\boldsymbol{x}_j)$ be the outputs of the n^{th} and m^{th} neurons in the i^{th} layer for the same input sample $\boldsymbol{x}_j$. The similarity s_{nm} between the the n^{th} and m^{th} neurons can be obtained as the average similarity measure of their outputs for N input samples. We use the radial basis function to express the similarity:

$$s_{nm} = \frac{1}{N}\sum_{j=1}^{N} \exp\big(-\gamma||\phi_n^i(\boldsymbol{x}_j) - \phi_m^i(\boldsymbol{x}_j)||^2\big), \quad (8)$$

where γ is a hyper-parameter. The similarity s_{nm} can be computed over the whole dataset or batch-wise. Intuitively, if two neurons n and m have similar outputs for many samples, their corresponding similarity s_{nm} will be high. Otherwise, their similarity s_{mn} is small and they are considered "diverse". Based on these pair-wise similarities, we propose three variants for the overall similarity J^i:

- **Direct:** $J^i = \sum_{n \neq m} s_{nm}$. In this variant, we model the global layer similarity directly as the sum of the pairwise similarities between the neurons.
- **Det:** $J^i = -\det(\mathbf{S})$, where $\boldsymbol{S}$ is defined as $\boldsymbol{S}_{nm} = s_{nm}$. This variant is inspired by the Determinantal Point Process (DPP) (Kulesza & Taskar, 2010; 2012).
- **Logdet:** $J^i = -\text{logdet}(\mathbf{S})$[1]. This variant has the same motivation as the second one. We use logdet instead of det as logdet is a convex function over the positive definite matrix space.

It should be noted here that the first proposed variant, i.e., direct, similar to Decov (Cogswell et al., 2016), captures only the pairwise diversity between components and is unable to capture the higher-order "diversity", whereas the other two variants consider the global similarity and are able to measure diversity in a more global manner.

4. Experiments

To demonstrate the effectiveness of our approach and its ability to reduce the generalization gap in neural networks, we conduct image classification experiments on the ImageNet-2012 dataset (Russakovsky et al., 2015) using the ResNet50 model (He et al., 2016). The diversity term is applied on the last intermediate layer, i.e., the global average pooling layer. The training protocol is presented in the appendix 6.3.

We analyse the effect of the two parameters: γ, which is the RBF parameter used to measure the pair-wise similarity between two units, and λ, which controls the contribution of the global diversity term to the global loss, on both the final performance of the models and its generalization ability, i.e., generalization gap. The analysis is presented in Figure 1. As it can be seen, promoting the within-layer diversity consistently reduces overfitting and decreases the generalization gap for most of the hyperparameters values.

[1]This is defined only if $\boldsymbol{S}$ is positive definite. It can be shown that in our case $\boldsymbol{S}$ is positive semi-definite. Thus, in practice we use a regularized version ($\boldsymbol{S} + \epsilon\boldsymbol{I}$) to ensure the positive definiteness.

Moreover, we note that global modeling of diversity, i.e., det and logdet variants, yield tighter generalization gaps between the train and test errors compared to the non-global direct approach. In fact, while direct variant decreases the generalization gap compared to the standard approach, it decreases it only by 0.5% for most hyperparameter values, whereas, for the more global approaches, i.e., det and logdet, the generalization gap is less than 1.1% in multiple cases compared to the gaps 2.87% and 2.50% achieved by the standard approach and the direct variant, respectively.

For the direct variant (the curves in blue), we note that the performance of the method is not sensitive the hyperparameters, and the method achieves its best performance for low values of λ and γ. For the det variant (the curves in orange), we note that it significantly improves the generalization ability of the model. However, there is a trade-off between the generalization gap and the final error. In fact, emphasizing diversity and using a high weight for the diversity term significantly decreases the generalization gap. This damages the performance of the model compared to the standard approach. For example, with $\lambda = 0.01$ and $\gamma = 10$, the generalization gap of the model is 0.9% compared to 2.87% of the standard. However, the test error for this model gets up to 24.42% compared to 23.87% for the standard. For lower values of λ, the model is able to significantly outperform the standard approach on both the test error and the generalization gap. For the logdet variant (green curves), we note that, in terms of generalization gap, the approach consistently outperforms the standard approach. Using a small value for λ, the model yields lower error rates than the standard approach. For high values of λ, the error rates become similar to the standard approach but with a lower generalization gap. This variant is not sensitive to the hyperparameter γ. Additional empirical results are presented in appendix 6.4.

5. Conclusions

In this paper, we proposed a new approach to encourage 'diversification' of the layer-wise feature map outputs in neural networks. The main motivation is that by promoting within-layer activation diversity, neurons within the same layer learn to capture mutually distinct patterns. We proposed an additional loss term that can be added on top of any layer. This term complements the traditional 'between-layer' feedback with an additional 'within-layer' feedback encouraging diversity of the activations. We theoretically proved that the proposed approach decreases the estimation error bound and, thus, improves the generalization ability of neural networks. This analysis was further supported by experimental results showing that such a strategy can indeed improve the performance of state-of-the-art networks.

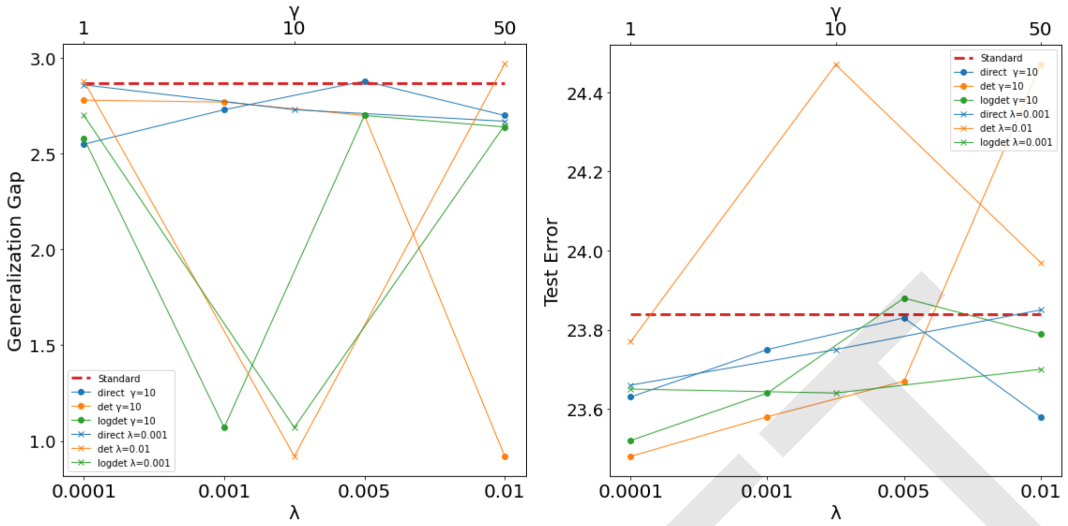


Figure 1. Sensitivity analysis of λ and γ on both the model accuracy and its generalization ability

Acknowledgments

This work was funded by the NSF-Business Finland Center for Visual and Decision Informatics (CVDI) project AMALIA. Jenni Raitoharju acknowledges funding from the Academy of Finland under project No 324475. Alexandros Iosifidis acknowledges funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337 (MARVEL).

References

Advani, M. S., Saxe, A. M., and Sompolinsky, H. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. Stronger generalization bounds for deep nets via a compression approach. pp. 254–263. Proceedings of Machine Learning Research, 10–15 Jul 2018.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 7413–7424, 2019.

Barron, A. R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, pp. 930–945, 1993.

Barron, A. R. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, pp. 115–133, 1994.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, pp. 463–482, 2002.

Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, pp. 63–1, 2019.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Bietti, A., Mialon, G., Chen, D., and Mairal, J. A kernel perspective for regularizing deep neural networks. In *International Conference on Machine Learning*, pp. 664–674, 2019.

Cogswell, M., Ahmed, F., Girshick, R. B., Zitnick, L., and Batra, D. Reducing overfitting in deep networks by decorrelating representations. In *International Conference on Learning Representations*, 2016.

Golan, I. and El-Yaniv, R. Deep anomaly detection using geometric transformations. In *Advances in Neural Information Processing Systems*, pp. 9758–9769, 2018.

Golowich, N., Rakhlin, A., and Shamir, O. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pp. 297–299, 2018.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*. MIT Press, 2016.

Hanneke, S. The optimal sample complexity of PAC learning. *Journal of Machine Learning Research*, pp. 1319–1333, 2016.

Harvey, N., Liaw, C., and Mehrabian, A. Nearly-tight vc-dimension bounds for piecewise linear neural networks. In *Conference on Learning Theory*, pp. 1064–1068, 2017.

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

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal processing magazine*, 29(6):82–97, 2012a.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012b.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Kukačka, J., Golkov, V., and Cremers, D. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017.

Kulesza, A. and Taskar, B. Structured determinantal point processes. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2010.

Kulesza, A. and Taskar, B. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012.

Lee, H. B., Nam, T., Yang, E., and Hwang, S. J. Meta dropout: Learning to perturb latent features for generalization. In *International Conference on Learning Representations*, 2019.

Li, N., Yu, Y., and Zhou, Z.-H. Diversity regularized ensemble pruning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 330–345, 2012.

Li, Z., Gong, B., and Yang, T. Improved dropout for shallow and deep learning. In *Advances in Neural Information Processing Systems*, pp. 2523–2531, 2016.

Malkin, J. and Bilmes, J. Multi-layer ratio semi-definite classifiers. In *International Conference on Acoustics, Speech and Signal Processing*, pp. 4465–4468, 2009.

Nagarajan, V. and Kolter, J. Z. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 11615–11626, 2019.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1g5sA4twr`.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2018.

Poggio, T., Kawaguchi, K., Liao, Q., Miranda, B., Rosasco, L., Boix, X., Hidary, J., and Mhaskar, H. Theory of deep learning III: explaining the non-overfitting puzzle. *arXiv preprint arXiv:1801.00173*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Sokolic, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. Lessons from the rademacher complexity for deep learning. 2016.

Sontag, E. D. VC dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences*, pp. 69–96, 1998.

Tang, Y., Wang, Y., Xu, Y., Shi, B., Xu, C., Xu, C., and Xu, C. Beyond dropout: Feature map distortion to regularize deep neural networks. In *Association for the Advancement of Artificial Intelligence*, pp. 5964–5971, 2020.

Wang, H., Yang, W., Zhao, Z., Luo, T., Wang, J., and Tang, Y. Rademacher dropout: An adaptive dropout for deep neural network via optimizing generalization gap. *Neurocomputing*, pp. 177–187, 2019.

Xie, B., Liang, Y., and Song, L. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics*, pp. 1216–1224, 2017a.

Xie, P., Deng, Y., and Xing, E. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv preprint arXiv:1511.07110*, 2015.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017b.

Yang, K., Gkatzelis, V., and Stoyanovich, J. Balanced ranking with diversity constraints. In *International Joint Conference on Artificial Intelligence*, pp. 6035–6042.

Yu, Y., Li, Y.-F., and Zhou, Z.-H. Diversity regularized machine. In *International Joint Conference on Artificial Intelligence*, 2011.

Zhai, K. and Wang, H. Adaptive dropout with rademacher complexity regularization. In *International Conference on Learning Representations*, 2018.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations 2018*, 2018.

6. Appendix

6.1. Theoretical assumptions

The full theoretical characterization of the setup can be summarized in the following assumptions:

Assumptions 1.

- *The activation function of the hidden layer, $\phi(t)$, is a positive L_ϕ-Lipschitz continuous function.*
- *The input vector $\boldsymbol{x} \in \mathbb{R}^D$ satisfies $||\boldsymbol{x}||_2 \leq C_1$.*
- *The output scalar $y \in \mathbb{R}$ satisfies $|y| \leq C_2$.*
- *The weight matrix $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_M] \in \mathcal{R}^{D\times M}$ connecting the input to the hidden layer satisfies $||\boldsymbol{w}_m||_2 \leq C_3$.*
- *The weight vector $\boldsymbol{v} \in \mathbb{R}^M$ connecting the hidden-layer to the output neuron satisfies $||\boldsymbol{v}||_2 \leq C_4$.*
- *The hypothesis class is $\mathcal{F} = \{f|f(\boldsymbol{x}) = \sum_{m=1}^M v_m\phi_m(\boldsymbol{x}) = \sum_{m=1}^M v_m\phi(\boldsymbol{w}_m^T\boldsymbol{x})\}$.*
- *Loss function set is $\mathcal{A} = \{l|l(f(\boldsymbol{x}), y) = \frac{1}{2}|f(\boldsymbol{x}) - y|^2\}$.*
- *With a probability τ, for $n \neq m$, $(\phi_n(\boldsymbol{x}) - \phi_m(\boldsymbol{x}))^2 = (\phi(\boldsymbol{w}_n^T\boldsymbol{x}) - \phi(\boldsymbol{w}_m^T\boldsymbol{x}))^2 \geq d_{min}^2$.*

6.2. Section 2 proofs:

We recall the following two lemmas related to the estimation error and three Rademacher complexity:

Lemma 1. *(Bartlett & Mendelson, 2002) For $\mathcal{F} \in \mathbb{R}^{\mathcal{X}}$, assume that $g : \mathbb{R} \to \mathbb{R}$ is a L_g-Lipschitz continuous function and $\mathcal{A} = \{g \circ f : f \in \mathcal{F}\}$. Then we have*

$$\mathcal{R}_N(\mathcal{A}) \leq L_g\mathcal{R}_N(\mathcal{F}). \tag{9}$$

Lemma 2. *(Xie et al., 2015; Bartlett & Mendelson, 2002) With a probability of at least $1 - \delta$*

$$L(\hat{f}) - L(f^*) \leq 4\mathcal{R}_N(\mathcal{A}) + B\sqrt{\frac{2\log(2/\delta)}{N}} \tag{10}$$

for $B \geq \sup_{\boldsymbol{x},y,f}|l(f(\boldsymbol{x}), y)|$, where $\mathcal{R}_N(\mathcal{A})$ is the Rademacher complexity of the loss set $\mathcal{A}$.

Lemma 3. *(Xie et al., 2015) Under Assumptions 1, the Rademacher complexity $\mathcal{R}_N(\mathcal{F})$ of the hypothesis class $\mathcal{F} = \{f|f(\boldsymbol{x}) = \sum_{m=1}^M v_m\phi_m(\boldsymbol{x}) = \sum_{m=1}^M v_m\phi(\boldsymbol{w}_m^T\boldsymbol{x})\}$ can be upper-bounded as follows:*

$$\mathcal{R}_N(\mathcal{F}) \leq \frac{2L_\phi C_{134}\sqrt{M}}{\sqrt{N}} + \frac{C_4|\phi(0)|\sqrt{M}}{\sqrt{N}}, \tag{11}$$

where $C_{134} = C_1C_3C_4$ and $\phi(0)$ is the output of the activation function at the origin.

Lemma 2 bounds the estimation error using the Rademacher complexity and the supremum of the loss class Lemma 3 provides an upper-bound of the Rademacher complexity for the hypothesis class.

In the following proofs, we use Lipschitz analysis. In particular, a function $f : \mathbb{A} \to \mathbb{R}$, $\mathbb{A} \subset \mathbb{R}^n$, is said to be L-Lipschitz, if there exist a constant $L \geq 0$, such that $|f(\boldsymbol{a}) - f(\boldsymbol{b})| \leq L||\boldsymbol{a} - \boldsymbol{b}||$ for every pair of points $\boldsymbol{a}, \boldsymbol{b} \in A$. Moreover:

- $\sup_{\boldsymbol{x}\in A} f \leq \sup(L||\boldsymbol{x}|| + f(0))$.
- if f is continuous and differentiable, $L = \sup|f'(\boldsymbol{x})|$.

6.2.1. Proof of Theorem 1

In order to find an upper-bound for our estimation error, we start by deriving an upper bound for $\sup_{\boldsymbol{x},f}|f(\boldsymbol{x})|$;

Lemma 4. *Under Assumptions 1, with a probability at least τ^Q, we have*

$$\sup_{\boldsymbol{x},f} |f(\boldsymbol{x})| \leq \sqrt{\mathcal{J}}, \tag{12}$$

where Q is equal to the number of neuron pairs defined by M neurons, i.e. $Q = \frac{M(M-1)}{2}$, and $\mathcal{J} = C_4^2\big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\big)$ and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Proof.

$$\begin{aligned} f^2(\boldsymbol{x}) &= \left(\sum_{m=1}^{M} v_m \phi_m(\boldsymbol{x})\right)^2 \leq \left(\sum_{m=1}^{M} ||\boldsymbol{v}||_\infty \phi_m(\boldsymbol{x})\right)^2 \\ &\leq ||\boldsymbol{v}||_\infty^2 \left(\sum_{m=1}^{M} \phi_m(\boldsymbol{x})\right)^2 \leq C_4^2 \left(\sum_{m=1}^{M} \phi_m(\boldsymbol{x})\right)^2 \\ &= C_4^2 \left(\sum_{m,n} \phi_m(\boldsymbol{x})\phi_n(\boldsymbol{x})\right) \\ &= C_4^2 \left(\sum_{m} \phi_m(\boldsymbol{x})^2 + \sum_{m\neq n} \phi_n(\boldsymbol{x})\phi_m(\boldsymbol{x})\right) \end{aligned} \tag{13}$$

We have $\sup_m w, \boldsymbol{x}\phi(\boldsymbol{x}) < \sup(L_\phi|\boldsymbol{w}^T\boldsymbol{x}| + \phi(0))$ because ϕ is L_ϕ-Lipschitz. Thus, $||\phi||_\infty < L_\phi C_1 C_3 + \phi(0) = C_5$. For the first term in equation 13, we have $\sum_m \phi_m(\boldsymbol{x})^2 < M(L_\phi C_1 C_3 + \phi(0))^2 = MC_5^2$. The second term, using the identity $\phi_m(\boldsymbol{x})\phi_n(\boldsymbol{x}) = \frac{1}{2}\left(\phi_m(\boldsymbol{x})^2 + \phi_n(\boldsymbol{x})^2 - (\phi_m(\boldsymbol{x}) - \phi_n(\boldsymbol{x}))^2\right)$, can be rewritten as

$$\sum_{m\neq n} \phi_m(\boldsymbol{x})\phi_n(\boldsymbol{x}) = \frac{1}{2}\sum_{m\neq n} \phi_m(\boldsymbol{x})^2 + \phi_n(\boldsymbol{x})^2 - \Big(\phi_m(\boldsymbol{x}) - \phi_n(\boldsymbol{x})\Big)^2. \tag{14}$$

In addition, we have with a probability τ, $||\phi_m(\boldsymbol{x}) - \phi_n(\boldsymbol{x})||_2^2 \geq d_{min}$ for $m \neq n$. Thus, we have with a probability at least τ^Q:

$$\begin{aligned} \sum_{m\neq n} \phi_m(\boldsymbol{x})\phi_n(\boldsymbol{x}) &\leq \frac{1}{2}\sum_{m\neq n} (2C_5^2 - d_{min}^2) \\ &= M(M-1)(C_5^2 - d_{min}^2/2). \end{aligned} \tag{15}$$

By putting everything back to equation 13, we have with a probability τ^Q,

$$f^2(\boldsymbol{x}) \leq C_4^2\Big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\Big) = \mathcal{J}. \tag{16}$$

Thus, with a probability τ^Q,

$$\sup_{\boldsymbol{x},f} |f(\boldsymbol{x})| \leq \sqrt{\sup_{\boldsymbol{x},f} f(\boldsymbol{x})^2} \leq \sqrt{\mathcal{J}}. \tag{17}$$

□

Note that in Lemma 4, we have expressed the upper-bound of $\sup_{\boldsymbol{x},f}|f(\boldsymbol{x})|$ in terms of d_{min}. Using this bound, we can now find an upper-bound for $\sup_{\boldsymbol{x},f,y}|l(f(\boldsymbol{x}), y)|$ in the following lemma:

Lemma 5. *Under Assumptions 1, with a probability at least τ^Q, we have*

$$\sup_{\boldsymbol{x},y,f} |l(f(\boldsymbol{x}), y)| \leq \frac{1}{2}(\sqrt{\mathcal{J}} + C_2)^2 \tag{18}$$

Proof. We have $\sup_{\boldsymbol{x},y,f}|f(\boldsymbol{x}) - y| \leq \sup_{\boldsymbol{x},y,f}(|f(\boldsymbol{x})| + |y|) = (\sqrt{\mathcal{J}} + C_2)$. Thus $sup_{x,y,f}|l(f(x), y)| \leq \frac{1}{2}(\sqrt{\mathcal{J}} + C_2)^2$. □

The main goal is to analyze the estimation error bound of the neural network and to see how its upper-bound is linked to the diversity, expressed by d_{min}, of the different neurons. Now we can prove our main Theorem 1:

Theorem 1 *Under Assumptions 1, there exist a constant A, such that with probability at least $\tau^Q(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq \Big(\sqrt{\mathcal{J}} + C_2\Big)A + \frac{1}{2}(\sqrt{\mathcal{J}} + C_2)^2\sqrt{\frac{2\log(2/\delta)}{N}} \tag{19}$$

where $\mathcal{J} = C_4^2\big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\big)$, and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Proof. Given that $l(\cdot)$ is K-Lipschitz with a constant $K = sup_{\boldsymbol{x},y,f}|f(\boldsymbol{x}) - y| \leq (\sqrt{\mathcal{J}} + C_2)$, and using Lemma 1, we can show that $\mathcal{R}_N(\mathcal{A}) \leq K\mathcal{R}_N(\mathcal{F}) \leq (\sqrt{\mathcal{J}} + C_2)\mathcal{R}_N(\mathcal{F})$. For $\mathcal{R}_N(\mathcal{F})$, we use the bound found in Lemma 3. Using Lemmas 2 and 5, we have

$$\begin{aligned} L(\hat{f}) - L(f^*) &\leq 4\Big(\sqrt{\mathcal{J}} + C_2\Big)\Big(2L_\phi C_{134} + C_4|\phi(0)|\Big)\frac{\sqrt{M}}{\sqrt{N}} \\ &+ \frac{1}{2}(\sqrt{\mathcal{J}} + C_2)^2\sqrt{\frac{2\log(2/\delta)}{N}} \end{aligned} \tag{20}$$

where $C_{134} = C_1C_3C_4$, $\mathcal{J} = C_4^2\big(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2)\big)$, and $C_5 = L_\phi C_1 C_3 + \phi(0)$. Thus, taking $A = 4\Big(2L_\phi C_{134} + C_4|\phi(0)|\Big)\frac{\sqrt{M}}{\sqrt{N}}$ completes the proof. □

6.2.2. Proof of Theorems 2 and 3

Similar to the proofs of Lemmas 7 and 8 in (Xie et al., 2015), we can show the following two lemmas:

Lemma 6. *Using the hinge loss, we have with probability at least $\tau^Q(1-\delta)$*

$$\begin{aligned} L(\hat{f}) - L(f^*) &\leq 4\Big(2L_\phi C_{134} + C_4|\phi(0)|\Big)\frac{\sqrt{M}}{\sqrt{N}} \\ &+ (1 + \sqrt{\mathcal{J}})\sqrt{\frac{2\log(2/\delta)}{N}} \end{aligned} \tag{21}$$

where $C_{134} = C_1C_3C_4$, $\mathcal{J} = C_4^2(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2))$, and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Lemma 7. *Using the logistic loss $l(f(x), y) = \log(1 + \boldsymbol{e}^{-yf(x)})$, we have with probability at least $\tau^Q(1-\delta)$*

$$L(\hat{f}) - L(f^*) \leq \frac{4}{1+\boldsymbol{e}^{\sqrt{-\mathcal{J}}}}\Big(2L_\phi C_{134} + C_4|\phi(0)|\Big)\frac{\sqrt{M}}{\sqrt{N}} + \log(1+\boldsymbol{e}^{\sqrt{\mathcal{J}}})\sqrt{\frac{2\log(2/\delta)}{N}} \quad (22)$$

where $C_{134} = C_1C_3C_4$, $\mathcal{J} = C_4^2(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2))$, and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Taking $A = 4\Big(2L_\phi C_{134} + C_4|\phi(0)|\Big)\frac{\sqrt{M}}{\sqrt{N}}$ in Lemma 6 and Lemma 7 completes the proofs.

6.2.3. Proof of Theorem 4

Theorem 4 There exist a constant A such that, with probability of at least $\prod_{p=0}^{P-1}(\tau^p)^{Q^p}(1-\delta)$, we have

$$L(\hat{f}) - L(f^*) \leq (\sqrt{\mathcal{J}^P} + C_2)A + \frac{1}{2}\left(\sqrt{\mathcal{J}^P} + C_2\right)^2\sqrt{\frac{2\log(2/\delta)}{N}} \quad (23)$$

where Q^p is the number of neuron pairs in the p^{th} layer, defined as $Q^p = \frac{M^p(M^p-1)}{2}$, and $\mathcal{J}^P$ is defined recursively using the following identities: $\mathcal{J}^0 = C_3^0 C_1$ and $\mathcal{J}^p = M^p C^{p2}\Big(M^{p2}(L_\phi C_3^{p-1}\mathcal{J}^{p-1} + \phi(0))^2 - M(M-1)\frac{{d_{min}^p}^2}{2})\Big)$, for $p = 1, \ldots, P$.

Proof. Lemma 5 in (Xie et al., 2015) provides an upper-bound for the hypothesis class. We denote by $\boldsymbol{v}^p$ denote the outputs of the p^{th} hidden layer before applying the activation function:

$$\boldsymbol{v}^0 = [\boldsymbol{w}_1^{0^T}\boldsymbol{x}, ..., \boldsymbol{w}_{M^0}^{0^T}\boldsymbol{x}] \quad (24)$$

$$\boldsymbol{v}^p = [\sum_{j=1}^{M^{p-1}} w_{j,1}^p \phi(\boldsymbol{v}_j^{p-1}), ..., \sum_{j=1}^{M^{p-1}} w_{j,M^p}^p \phi(v_j^{p-1})] \quad (25)$$

$$\boldsymbol{v}^p = [\boldsymbol{w}_1^{pT}\boldsymbol{\phi}^p, ..., \boldsymbol{w}_{M^p}^{p\ T}\boldsymbol{\phi}^p], \quad (26)$$

where $\boldsymbol{\phi}^p = [\phi(v_1^{p-1}), \cdots, \phi(v_{M^{p-1}}^{p-1})]$. We have

$$||\boldsymbol{v}^p||_2^2 = \sum_{m=1}^{M^p}(\boldsymbol{w}_m^{p\ T}\boldsymbol{\phi}^p)^2 \quad (27)$$

and $\boldsymbol{w}_m^{p\ T}\boldsymbol{\phi}^p \leq C_3^p \sum_n \phi_n^p$. Thus,

$$||\boldsymbol{v}^p||_2^2 \leq \sum_{m=1}^{M^p}(C_3^p\sum_n \phi_n^p)^2 = M^p C_3^{p2}(\sum_n \phi_n^p)^2 = M^p C_3^{p2}\sum_{mn}\phi_m^p\phi_n^p. \quad (28)$$

We use the same decomposition trick of $\phi_m^p\phi_n^p$ as in the proof of Lemma 4. We need to bound $\sup_x \phi^p$:

$$\sup_x \phi^p < \sup(L_\phi|\boldsymbol{w}_j^{p-1^T}\boldsymbol{v}^{p-1}| + \phi(0)) < L_\phi||\boldsymbol{W}^{p-1}||_\infty||\boldsymbol{v}^{p-1}||_2^2 + \phi(0). \quad (29)$$

Thus, we have

$$||\boldsymbol{v}^p||_2^2 \leq M^p C^{p2}\big(M^2(L_\phi C_3^{p-1}||\boldsymbol{v}^{p-1}||_2^2 + \phi(0))^2 - M(M-1)d_{min}^2/2)\big) = \mathcal{J}^P. \quad (30)$$

We found a recursive bound for $||\boldsymbol{v}^p||_2^2$, we note that for $p = 0$, we have $||\boldsymbol{v}^0||_2^2 \leq ||W^0||_\infty C_1 \leq C_3^0 C_1 = \mathcal{J}^0$. Thus,

$$\sup_{\boldsymbol{x}, f^P\in\mathcal{F}^P}|f(\boldsymbol{x})| = \sup_{\boldsymbol{x}, f^P\in\mathcal{F}^P}|\boldsymbol{v}^P| \leq \sqrt{\mathcal{J}^P}. \quad (31)$$

By replacing the variables in Lemma 2, we have

$$L(\hat{f}) - L(f^*) \leq 4(\sqrt{\mathcal{J}^P} + C_2)\Bigg(\frac{(2L_\phi)^P C_1 C_3^0}{\sqrt{N}}\prod_{p=0}^{P-1}\sqrt{M^p}C_3^p + \frac{|\phi(0)|}{\sqrt{N}}\sum_{p=0}^{P-1}(2L_\phi)^{P-1-p}\prod_{j=p}^{P-1}\sqrt{M^j}C_3^j\Bigg) + \frac{1}{2}\left(\sqrt{\mathcal{J}^P} + C_2\right)^2\sqrt{\frac{2\log(2/\delta)}{N}}$$

Taking $A = (\frac{(2L_\phi)^P C_1 C_3^0}{\sqrt{N}}\prod_{p=0}^{P-1}\sqrt{M^p}C_3^p + \frac{|\phi(0)|}{\sqrt{N}}\sum_{p=0}^{P-1}(2L_\phi)^{P-1-p}\prod_{j=p}^{P-1}\sqrt{M^j}C_3^j)$ completes the proof. □

6.2.4. Proofs of Theorems 5 and 6

Theorem 5 *For a multivariate regression trained with the squared error, there exist a constant A such that, with probability at least $\tau^Q(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq (\sqrt{\mathcal{J}} + C_2)A + \frac{D}{2}(\sqrt{\mathcal{J}} + C_2)^2\sqrt{\frac{2\log(2/\delta)}{N}} \quad (32)$$

where $\mathcal{J} = C_4^2(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2))$ and $C_5 = L_\phi C_1 C_3 + \phi(0)$

Proof. The squared loss $||f(\boldsymbol{x}) - \boldsymbol{y}||^2$ can be decomposed into D terms $(f(\boldsymbol{x})_k - y_k)^2$. Using Theorem 1, we can derive the bound for each term and thus, we have:

$$L(\hat{f}) - L(f^*) \leq 4D(\sqrt{\mathcal{J}} + C_2)\Big(2L_\phi C_{134} + C_4|\phi(0)|\Big)\frac{\sqrt{M}}{\sqrt{N}} + \frac{D}{2}(\sqrt{\mathcal{J}} + C_2)^2\sqrt{\frac{2\log(2/\delta)}{N}}, \quad (33)$$

where $C_{134} = C_1C_3C_4$, $\mathcal{J} = C_4^2(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2))$, and $C_5 = L_\phi C_1 C_3 + \phi(0)$. Taking $A = 4D\Big(2L_\phi C_{134} + C_4|\phi(0)|\Big)\frac{\sqrt{M}}{\sqrt{N}}$ completes the proof. □

Theorem 6 *For a multi-class classification task using the cross-entropy loss, there exist a constant A such that, with probability at least $\tau^Q(1-\delta)$, we have*

$$L(\hat{f}) - L(f^*) \leq \frac{A}{D-1+\boldsymbol{e}^{-2\sqrt{\mathcal{J}}}} + \log\left(1+(D-1)\boldsymbol{e}^{2\sqrt{\mathcal{J}}}\right)\sqrt{\frac{2\log(2/\delta)}{N}} \quad (34)$$

where $\mathcal{J} = C_4^2(MC_5^2 + M(M-1)(C_5^2 - d_{min}^2/2))$ and $C_5 = L_\phi C_1 C_3 + \phi(0)$.

Proof. Using Lemma 9 in (Xie et al., 2015), we have $\sup_{f,\boldsymbol{x},\boldsymbol{y}} l = \log\left(1+(D-1)\boldsymbol{e}^{2\sqrt{\mathcal{J}}}\right)$ and l is $\frac{D-1}{D-1+\boldsymbol{e}^{-2\sqrt{\mathcal{J}}}}$-Lipschitz. Thus, using the decomposition property of the Rademacher complexity, we have

$$\mathcal{R}_n(\mathcal{A}) \leq \frac{4D(D-1)}{D-1+\boldsymbol{e}^{-2\sqrt{\mathcal{J}}}}\left(\frac{2L_\phi C_{134}\sqrt{M}}{\sqrt{N}} + \frac{C_4|\phi(0)|\sqrt{M}}{\sqrt{N}}\right). \quad (35)$$

Taking $A = 4D(D-1)\left(\frac{2L_\phi C_{134}\sqrt{M}}{\sqrt{N}} + \frac{C_4|\phi(0)|\sqrt{M}}{\sqrt{N}}\right)$ completes the proof. □

6.3. Experimental protocol

we conduct image classification experiments on the ImageNet-2012 classification dataset (Russakovsky et al., 2015) using the ResNet50 model (He et al., 2016). The diversity term is applied on the last intermediate layer, i.e., the global average pooling layer for both DeCov and our method. We use the standard augmentation practice for this dataset as in (Zhang et al., 2018; Huang et al., 2017; Cogswell et al., 2016). All the models are trained with a batch size of 256 for 100 epoch using SGD with Nesterov Momentum of 0.9 and a weight decay of 0.0001. The learning rate is initially set to 0.1 and decreases at epochs 30, 60, 90 by a factor of 10.

6.4. Additional experiments

We start by evaluating our proposed diversity approach on two image datasets: CIFAR10 and CIFAR100 (Krizhevsky et al., 2009). They contain 60,000 (50,000 train/10,000 test) 32×32 images grouped into 10 and 100 distinct categories, respectively. We split the original training set (50,000) into two sets: we use the first 40,000 images as the main training set and the last 10,000 as a validation set for hyperparameters optimization. We use our approach on three state-of-the-art CNNs: **ResNext 29-8-16**: we consider the standard ResNext Model (Xie et al., 2017b) with a 29-layer architecture, a cardinality of 8, and a width of 16. **DenseNet-12**: we use DenseNet (Huang et al., 2017) with the 40-layer architecture and a growth rate of 12. **ResNet50**: we consider the standard ResNet model (He et al., 2016) with 50 layers. We compare against the standard networks as well networks trained with DeCov diversity strategy (Cogswell et al., 2016).

Table 1. Average classification errors on CIFAR10 and CIFAR100 over three iterations

		Top-1 test Error	
Model	Method	CIFAR10	CIFAR100
DenseNet-12	Standard	07.07	29.25
	DeCov	07.18	29.17
	Ours direct	06.95	29.16
	Ours det	07.04	28.78
	Ours logdet	06.96	29.15
ResNext-29-08-16	Standard	06.93	26.73
	DeCov	06.84	26.70
	Ours direct	06.74	26.54
	Ours det	06.67	26.67
	Ours logdet	06.70	26.67
ResNet50	Standard	08.27	34.06
	DeCov	08.03	32.26
	Ours direct	07.86	32.15
	Ours det	07.73	32.12
	Ours logdet	07.91	32.20

All the models are trained using stochastic gradient descent (SGD) with a momentum of 0.9, weight decay of 0.0001, and a batch size of 128 for 200 epochs. The initial learning rate is set to 0.1 and is then decreased by a factor of 5 after 60, 120, and 160 epochs, respectively. We also adopt a standard data augmentation scheme that is widely used for these two datasets (He et al., 2016; Huang et al., 2017). For all models, the additional diversity term is applied on top the last intermediate layer. For the hyperparameters: The loss weight is chosen from $\{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01\}$ for both our approach and Decov and γ in the radial basis function is chosen from $\{0.01, 0.1.1, 10, 50, 100\}$. For each approach, the model with the best validation performance is used in the test phase. Each experiment is repeated three times and we report the average performance over three iterations.

Table 1 reports the average top-1 errors of the different approaches with the three basis networks. We note that, compared to the standard approach, employing a diversity strategy generally boosts the results for all the three models and that our approach consistency outperforms both compet-

Table 2. Performance of ResNet50 with different diversity strategies on ImageNet dataset

	Top-1 Errors		Generalization
Method	Training	Testing	Gap
Standard	20.97	23.84	2.87
DeCov	20.92	23.62	2.70
Ours direct	20.88	23.58	2.70
Ours det	20.81	23.62	2.77
Ours logdet	22.57	23.64	1.07

ing methods (standard and DeCov) in all the experiments. For DenseNet-12, our direct and det variants yield the best performance over CIFAR10 and CIFAR100, respectively. For ResNext-29-08-16, our approach with det term yields on average 0.25% accuracy improvement compared to the standard approach, 0.17% improvement compared to DeCov on CIFAR10. On CIFAR100, the best performance is achieved by the direct variant of our approach. For ResNet50, the three variants of our proposed approach significantly reduce the test errors over both datasets: $0.36\% - 0.54\%$ improvement on CIFAR10 and $1.86\% - 1.94\%$ on CIFAR100.

To further demonstrate the effectiveness of our approach and its ability to reduce the generalization gap in neural networks, we conduct additional image classification experiments on the ImageNet-2012 classification dataset and ResNet50. For the hyperparameters, we use the best ones for each approach obtained from CIFAR10 and CIFAR100 experiments.

Table 2 reports the test errors of the different diversity strategies. To study the effect of diversity on the generalization gap, we also report the final training errors and the generalization gap, i.e., training accuracy - test accuracy. As it can be seen, diversity (our approach and DeCov) reduces the test error of the model and yields a better performance. The best performance is achieved by our direct variant. We note that, in accordance with our theoretical findings in Section 2, using diversity indeed reduces overfitting and decreases the empirical generalisation gap of neural networks. In fact, our logdet variant reduces the empirical generalization gap of the model by 1.8% compared to the standard approach.

7.14 Enriched Music Representations with Multiple Cross-modal Contrastive Learning

The appended paper follows.

Enriched Music Representations with Multiple Cross-modal Contrastive Learning

Andres Ferraro, Xavier Favory, Konstantinos Drossos, Yuntae Kim and Dmitry Bogdanov

***Abstract*—Modeling various aspects that make a music piece unique is a challenging task, requiring the combination of multiple sources of information. Deep learning is commonly used to obtain representations using various sources of information, such as the audio, interactions between users and songs, or associated genre metadata. Recently, contrastive learning has led to representations that generalize better compared to traditional supervised methods. In this paper, we present a novel approach that combines multiple types of information related to music using cross-modal contrastive learning, allowing us to learn an audio feature from heterogeneous data simultaneously. We align the latent representations obtained from playlists-track interactions, genre metadata, and the tracks' audio, by maximizing the agreement between these modality representations using a contrastive loss. We evaluate our approach in three tasks, namely, genre classification, playlist continuation and automatic tagging. We compare the performances with a baseline audio-based CNN trained to predict these modalities. We also study the importance of including multiple sources of information when training our embedding model. The results suggest that the proposed method outperforms the baseline in all the three downstream tasks and achieves comparable performance to the state-of-the-art.**

***Index Terms*—Acoustic signal processing, Machine learning, Music information retrieval, Recommender systems**

I. Introduction and Related Work

There are multiple sources and types of information related to the music that can be used for different applications. For example, using audio features showed better performance for predicting musical genres compared to using users' listening data [1]. On the other hand, the latter performed better on music recommendation [2] and mood prediction [3]. Having a numerical feature representation that combines all the relevant information of a song would allow creating better automatic tools that solve problems such as genre prediction, mood estimation and music recommendation.

Advances of deep learning in the past years enabled to improve the performance on multiple tasks by combining different types of data. For example, Oramas et al. [4] propose a multi-modal approach combining text, audio, and images for music auto-tagging and Suris et al. [5] propose a method to combine audio-visual embeddings for cross-modal retrieval.

Deep learning allows learning representations mapping from different input data to an embedding space that can be used for multiple downstream tasks [6]. The most common approach for representation learning in the music domain is to train a audio-based classifier to predict some music aspects such as genre, mood, or instrument and then use the pre-trained model to extract embeddings that could be used in different tasks. Alonso et al. [7] compare different pre-trained architectures for predicting multiple aspects of a song such as danceability, mood, gender and timbre, showing the generalization capabilities of these pre-trained models. Alternative methods in the field of deep metric learning recently shown a better performance across multiple downstream tasks compared to the approach of pre-training classification models [8], [9], demonstrating the great potential of deep metric learning for generalizing to a larger diversity of tasks.

Contrastive learning has gained popularity in the last years [10]. These approaches allow to learn representation by employing a metric learning objective, contrasting similar and dissimilar items. The similar examples are referred as positive examples and the dissimilar are referred as negative examples. Approaches based on triplet loss [11] require to define triplets composed of an anchor, a positive and a negative example. Triplet loss was recently applied in the music domain for retrieval [1] and zero-shot learning [12]. However, the strategy for sampling the triplets is crucial to the learning process and can require significant effort. There are other losses that instead of defining triplets rely on the comparison of paired examples such as *infoNCE* [13] and *NT-Xent* [14]. They have the advantage of involving all the data points within a mini-batch when training without requiring to define a specific strategy for sampling the training examples. Employing these contrastive loss functions in a self-supervised way has led to powerful image [14], sound [15] and music audio [16] representations learned without the need for annotated data. Contrastive learning was also applied in a supervised way [17], [18] with a cross-modal approach using sound (audio) information and associated text metadata in order to learn semantically enriched audio features. The learned features achieve competitive performance in urban sound event and musical instrument recognition [17].

The works mentioned above suggest that methods based on contrastive learning have the potential to exploit different types of data which is promising for improving the performance of deep audio embeddings for a large diversity of tasks. However, to our knowledge, there is no work that focuses on leveraging multiple modalities through contrastive learning in order to learn rich musical audio features. This motivates us to investigate approaches that take advantage of different types of music-related information (i.e. audio, genre, and playlists) to obtain representations from the audio that can perform well in

Submitted for review on 5th of Feb. 2021.

A. Ferraro, X. Favory and D. Bogdanov are with Music Technology Group, Universitat Pompeu Fabra, Spain, email {firstname.lastname}@upf.edu

K. Drossos is with Audio Research Group, Tampere University, Finland, email konstantinos.drossos@tuni.fi

Y. Kim is with Kakao Corp, Korea, email daniel.py@kakaocorp.com

Thanks to Tuomas Virtanen, Soohyeon Lee and Biho Kim for their valuable feedback. This work was partially supported by Kakao Corp. K. Drossos was partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, project MARVEL

multiple downstream tasks such as music genre classification, automatic playlist continuation, and music automatic tagging. Our results show that the proposed contrastive learning approach reaches performance comparable to the state-of-art and outperforms models pre-trained for classification or regression based on the musical aspect.

Our contributions are as follows: i) We propose an updated audio encoder optimized for the music domain based on the approach proposed by Favory et al. [17], [19]. ii) We use the alignment of multi-modal data for exploiting the semantic metadata and collaborative filtering information. iii) We evaluate the obtained representations in three downstream tasks using different datasets comparing with other common approaches based on pre-training for classification or regression. iv) We also include an ablation study by comparing the performance of each source of information independently, which allows us to understand the importance of the different parts of our model. [1]

II. Proposed method

Our method employs the encoders $e_{\text{a}}(\cdot)$, $e_{\text{w}}(\cdot)$, and $e_{\text{cf}}(\cdot)$, encoding audio and embeddings of musical genres and music playlist information, respectively, and a dataset $\mathbb{D} = \{(\mathbf{X}_{\text{a}}, \mathbf{X}_{\text{w}}, \mathbf{x}_{\text{cf}})^m\}_{m=1}^{M}$, of M associated examples, where "a", "w", and "cf" are indices that associate the variables with the encoders. $\mathbf{X}_{\text{a}}^m \in \mathbb{R}^{T_{\text{a}} \times F_{\text{a}}}$ is a sequence of T_{a} vectors of F_{a} features of music audio signals, $\mathbf{X}_{\text{w}}^m \in \mathbb{R}^{T_{\text{w}} \times F_{\text{w}}}$ is a sequence of T_{w} word embeddings of the musical genres assigned to $\mathbf{X}_{\text{a}}^m$ with F_{w} features, and $\mathbf{x}_{\text{cf}}^m \in \mathbb{R}_{\geq 0}^{1 \times F_{\text{cf}}}$ is a vector of F_{cf} features correlating $\mathbf{X}_{\text{w}}^m$ with a human created playlist. By the encoders we obtain three latent representations and their information is mutually aligned using three contrastive losses between associated and non-associated examples. By the joint minimization of the losses, we obtain the optimized $e_{\text{a}}^{\star}$, later used for calculating embeddings of music signals (see Figure 1).

A. Obtaining the latent representations

The audio encoder e_{a} consists of Z stacked 2D-CNN blocks, 2DCNN_z, and a feed-forward block, FFB. Each 2DCNN_z consists of a 2D convolutional neural network (CNN_z) with a square kernel of size K_z and unit stride, a batch normalization process (BN), a rectified linear unit (ReLU), and a pooling operation (PO). The FFB consists of a feed-forward layer, FF_{a1}, another BN process, a ReLU, a dropout with probability p, another feed-forward layer, FF_{a2}, and a layer normalization (LN) process. e_{a} takes as an input $\mathbf{X}_{\text{a}}^m$ and the Z 2D-CNN blocks and the feed-forward block process the input in a serial way. The output of e_{a} is the learned representation $\boldsymbol{\phi}_{\text{a}}^m = e_{\text{a}}(\mathbf{X}_{\text{a}}^m)$, computed as

$$\mathbf{H}_z^m = \text{2DCNN}_z(\mathbf{H}_{z-1}^m), \text{ and} \quad (1)$$

$$\boldsymbol{\phi}_{\text{a}}^m = \text{FFB}(\mathbf{H}_Z^m), \text{ where} \quad (2)$$

$$\text{2DCNN}_z(u) = (\text{PO} \circ \text{ReLU} \circ \text{BN} \circ \text{CNN}_z)(u), \quad (3)$$

$$\text{FFB}(u) = (\text{LN} \circ \text{FF}_{\text{a2}} \circ \text{DP} \circ \text{ReLU} \circ \text{BN} \circ \text{FF}_{\text{a1}})(u), \quad (4)$$

[1]We provide the code to reproduce this work and the pre-trained models: https://github.com/andrebola/contrastive-mir-learning

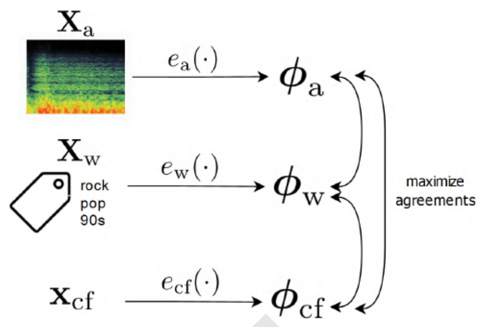


Fig. 1. Diagram with architecture of the method

$\mathbf{H}_z^m \in \mathbb{R}^{T'' \times F''}$, $\mathbf{H}_0^m = \mathbf{X}_{\text{a}}^m$, $\circ$ is the function composition symbol, i.e. $(f \circ g)(x) = f(g(x))$, and values of T'' and F'' depend on the hyper-parameters of CNN_z.

The encoder $e_{\text{w}}(\cdot)$ is the genre encoder and consists of a self-attention (SA) over the input sequence, a feed-forward layer (FF_{w}), DP with probability p, an LN process, and a skip connection between the input of the feed-forward layer and its output. $e_{\text{w}}(\cdot)$ is after the self-attention mechanism employed in the Transformer model [20], and is used to learn a contextual embedding of its input, similarly to [19]. Each musical genre associated with $\mathbf{X}_{\text{a}}^m$ is first one-hot encoded and then given as an input to the pre-optimized word embeddings model Word2Vec [21]. The output of Word2Vec is $\mathbf{X}_{\text{w}}^m$, which is then given as an input to $e_{\text{w}}(\cdot)$. The output of $e_{\text{w}}(\cdot)$ is the vector $\boldsymbol{\phi}_{\text{w}}^m = e_{\text{w}}(\mathbf{X}_{\text{w}}^m)$, containing the contextual embedding of $\mathbf{X}_{\text{w}}^m$ and calculated as

$$\mathbf{V}'^m = \text{SF}(\mathbf{X}_{\text{w}}^m), \quad (5)$$

$$\mathbf{V}^m = \mathbf{V}'^m + (\text{DP} \circ \text{FF}_{\text{w}})(\mathbf{V}'^m), \text{ and} \quad (6)$$

$$\boldsymbol{\phi}_{\text{w}}^m = \text{LN}(\sum_{i=1}^{T_{\text{w}}} \mathbf{V}_i^m), \quad (7)$$

where $\mathbf{V}'^m$, $\mathbf{V}^m \in \mathbb{R}^{T_{\text{w}} \times F'_{\text{w}}}$.

The third encoder, $e_{\text{cf}}(\cdot)$, is the playlist association encoder and consists of a feed-forward block, similar to $e_{\text{a}}(\cdot)$, Specifically, $e_{\text{cf}}(\cdot)$ consists of a feed-forward layer, FF_{cf1}, a ReLU, a dropout process with probability p, another feed-forward layer, FF_{cf2}, and a LN process. The input to $e_{\text{cf}}(\cdot)$ is a vector, $\mathbf{x}_{\text{cf}}^m$, obtained by a collaborative filtering (CF) process, using M_{pl} playlists created by humans.

The CF process gets as input a binary matrix, $\mathbf{B}_{\text{cf}} \in \{0,1\}^{M \times M_{\text{pl}}}$, that indicates which songs are included on which playlist, where M_{pl} is the amount of playlists. Then, we minimize the WARP loss (Weighted Approximate-Rank Pairwise loss) using SGD and the sampling technique defined in [22], to approximate ranks between playlists and songs efficiently. CF outputs the matrices $\mathbf{X}_{\text{cf}} \in \mathbb{R}_{\geq 0}^{M \times F_{\text{cf}}}$ and $\mathbf{Q}_{\text{cf}} \in \mathbb{R}_{\geq 0}^{F_{\text{cf}} \times M_{\text{pl}}}$, where $\mathbf{B}_{\text{cf}} \simeq \mathbf{X}_{\text{cf}} \cdot \mathbf{Q}_{\text{cf}}$. We use each row of $\mathbf{X}_{\text{cf}}$ as the vector $\mathbf{x}_{\text{cf}}^m$. We employ $e_{\text{cf}}(\cdot)$ to process the $\mathbf{x}_{\text{cf}}^m$, by providing a representation of $\mathbf{x}_{\text{cf}}^m$ that is learned specifically for the alignment process that our method tries to achieve. The output of $e_{\text{cf}}(\cdot)$ is the vector $\boldsymbol{\phi}_{\text{cf}}^m = e_{\text{cf}}(\mathbf{x}_{\text{cf}}^m)$, calculated as

$$\boldsymbol{\phi}_{\text{cf}}^m = (\text{LN} \circ \text{FF}_{\text{cf2}} \circ \text{DP} \circ \text{ReLU} \circ \text{FF}_{\text{cf1}})(\mathbf{x}_{\text{cf}}^m). \quad (8)$$

B. Optimization and alignment of latent representations

We jointly optimize all encoders using $\mathbb{D}$ and three contrastive losses. We expand previous approaches on audio representation learning using multi-modal alignment, by employing multiple cross-modal and single modal alignment processes. Specifically, we align ϕ_{a}^{m} with ϕ_{w}^{m} (audio-to-genre, A2G, alignment), ϕ_{a}^{m} with ϕ_{cf}^{m} (audio-to-playlist, A2P, alignment), and ϕ_{cf}^{m} with ϕ_{cf}^{m} (genre-to-playlist, G2P, alignment).

We use A2G alignment so that ϕ_{a}^{m} keeps information about musical genre. Additionally, we further enhance the information in ϕ_{a}^{m} by the A2P alignment, which is targeted to allow ϕ_{a}^{m} to have information about playlist associations. Finally, we employ G2P alignment, so that we keep genre and playlist related information tied up together and not let them degenerate to some representation that just helps to minimize the employed losses. Specifically, we use the contrastive loss between two paired examples, ψ_{α} and ψ_{b}, defined as [14], [17]

$$\mathcal{L}_{\psi_{\alpha},\psi_{b}} = \sum_{i=1}^{M} -\log \frac{\Xi(\psi_{\alpha}^{i}, \psi_{b}^{i}, \tau)}{\sum\limits_{k=1}^{2M} \mathbb{1}_{[k \neq i]} \Xi(\psi_{\alpha}^{i}, \zeta^{k}, \tau)}, \text{ where} \tag{9}$$

$$\Xi(\mathbf{a}, \mathbf{b}, \tau) = \exp(\text{sim}(\mathbf{a}, \mathbf{b})\tau^{-1}), \tag{10}$$

$$\text{sim}(\mathbf{a}, \mathbf{b}) = \mathbf{a}^{\top}\mathbf{b}(||\mathbf{a}||\ ||\mathbf{b}||)^{-1}, \tag{11}$$

$$\zeta^{k} = \begin{cases} \psi_{a}^{k}, \text{ if } k \leq M \\ \psi_{b}^{k-M} \text{ else} \end{cases}, \tag{12}$$

$\mathbb{1}_{\text{A}}$ is the indicator function with $\mathbb{1}_{\text{A}} = 1$ iff A else 0, and τ is a temperature hyper-parameter.

We identify $\mathcal{L}_{\text{A2G}} = \mathcal{L}_{\phi_{\text{a}},\phi_{\text{w}}} + \mathcal{L}_{\phi_{\text{w}},\phi_{\text{a}}}$ as the loss for A2G alignment, $\mathcal{L}_{\text{A2P}} = \mathcal{L}_{\phi_{\text{a}},\phi_{\text{cf}}} + \mathcal{L}_{\phi_{\text{cf}},\phi_{\text{a}}}$ as the loss for A2P alignment, and $\mathcal{L}_{\text{G2P}} = \mathcal{L}_{\phi_{\text{w}},\phi_{\text{cf}}} + \mathcal{L}_{\phi_{\text{cf}},\phi_{\text{w}}}$ as the loss for G2P alignment. We optimize all of our encoders, obtaining $e_{\text{a}}^{\star}$, by minimizing the

$$\mathcal{L}_{\text{tot}} = \lambda_{\text{A2G}}\mathcal{L}_{\text{A2G}} + \lambda_{\text{A2P}}\mathcal{L}_{\text{A2P}} + \lambda_{\text{G2P}}\mathcal{L}_{\text{G2P}}, \tag{13}$$

where $\lambda_{\cdot}$ are different hyper-parameters used as weighting factors for the losses.

III. Evaluation

To evaluate our method, we employ Melon Playlist Dataset [23] as $\mathbb{D}$, in order to obtain e_{a}. Then, we assess the learned representations by e_{a} applying it in different downstream tasks. Specifically, we focus on genre classification, audio-tagging, and automatic playlist continuation. For each of the tasks, we employ e_{a} as audio encoder, which will provide embeddings to a classifier, trained for the corresponding task.

We assess the benefit of the contribution of each of the encoders of our method, by comparing our method using three encoders (Contr$_{\text{CF-G}}$) with our method but using only e_{a} and e_{w} (Contr$_{\text{G}}$), and using only e_{a} and e_{cf} (Contr$_{\text{CF}}$). In addition, we compare the performance on each task using a baseline architecture that directly predicts the target information from the audio encoder. We refer to these methods as B-line$_{\text{G}}$ for the model trained with genre information, B-line$_{\text{CF}}$ for the model trained to predict CF information and B-line$_{\text{CF-G}}$ for the model trained to predict both types of information at the same time.

A. Melon Playlist Dataset and audio features

The dataset $\mathbb{D}$ used to train the models was originally collected Melon, a Korean music streaming service. The dataset consists of M=649,091 songs, represented by their mel-spectrograms, and M_{pl}=148,826 playlists. The number of unique genres asociated with the songs is 219. In order to train the model we split the songs of the dataset in train (80%), validation (10%) and test (10%). The split was done applying a stratified approach [24] in order to assure a similar distribution of example in all the sets for the genres associated to the songs.

The pre-computed mel-spectrograms provided in the dataset correspond to a range of 20 to 50 seconds with a resolution of $F_{\text{a}} = 48$ mel-bands. Such reduced mel-bands resolution did not negatively affect the performance of the auto-tagging approaches in our previous study [25] and have a significantly lower quality of reconstructed audio which allows to avoid copyright issues. Following the previous work [26], we randomly select sections the songs to train the audio encoder, using $T_{\text{a}} = 256$. [2]

B. Parameters optimization

Following the best performance in previous work [1], [26] the audio encoder use Z=7 layers and K=3. We conducted a preliminary evaluation to select the hyper-parameters of the models, comparing the loss in the validation and training set to prevent the models of overfitting. We defined the dimensions for CF representations to F_{cf}= 300 and genres representations F_{w}= 200 with $T_{\text{w}} <= 10$ genres per song. From the same preliminary evaluation we defined the temperature τ=0.1, batch size of 128, learning rate of 1e-4, dropout of 0.5 and the number of heads for self-attention of 4. We did not experiment with changing the weights λ for the different losses and we used $\lambda_{\text{A2G}} = \lambda_{\text{A2P}} = \lambda_{\text{G2P}} = 1$.

C. Downstream tasks

Once the models are trained with the Melon Playlist Dataset, we use the pre-trained models to generate an embedding from the audio of each song in the different datasets. Then, we use the generated embeddings and compare the performance for each particular task. In the following, we describe each downstream task and the dataset used.

Genre Classification We use the fault-filtered version of the GTZAN dataset [27], [28] consisting of music excepts of 30 seconds, single-labeled using 10 classes and split in pre-computed sets of 443 songs for training and 290 for testing. We train a multilayer perceptron (MLP) of one hidden layer of size 256 with ReLU activations, using the training set and compute its accuracy on the test set. In order to obtain an unbiased evaluation, we repeat this process 10 times and average the accuracies. We consider each embedding frame of a track as a different training instance, and when inferring the genres, we apply a majority voting strategy. We also include the performance of pre-trained embedding models taken from the literature [29]–[31], using the results reported in [30].

[2]We trained using Tesla V100-SXM2 GPU with 32 GB of memory, the training took 19 minutes per epoch approximately.

TABLE I
GTZAN RESULTS

Model	Mean Accuracy $\pm$ STD
B-line$_{\mathrm{G}}$	63.28 $\pm$ 1.19
B-line$_{\mathrm{CF}}$	57.12 $\pm$ 1.82
B-line$_{\mathrm{CF\text{-}G}}$	64.35 $\pm$ 1.10
Contr$_{\mathrm{G}}$	**76.78** $\pm$ 1.22
Contr$_{\mathrm{CF}}$	67.12 $\pm$ 0.94
Contr$_{\mathrm{CF\text{-}G}}$	75.29 $\pm$ 1.32
VGGish Audioset [31]	77.58
OpenL3 Audioset [29]	74.65
musicnn MSD [30]	77.24

Automatic Tagging. We rely on the MTG-Jamendo dataset [32] which contains over 55,000 full audio tracks multi-labeled using 195 different tags from *genre*, *instrument*, and *mood/theme* categories.[3] For this task, we train a MLP that takes our pre-trained audio embeddings as input. We compute the embedding of all the tracks by averaging their embeddings computed on non-overlapping frames with the mean statistic. The model is composed of two hidden layers of size 128 and 64 with ReLU activations, it includes batch normalizations after each layer and a dropout regularization after the penultimate layer. We use the validation sets for early stopping and we finally evaluate the performances on the test sets using ROC AUC. These evaluations are done on the three separated category of tags, each of them uses its own split. We repeat the procedures 10 times and report the mean average.

Playlist Continuation. We make use of the playlists from the Melon Playlist Dataset that contain at least one track in our test set (not used when training our embedding model). This provides 104,410 playlists, for the which we aim at providing 100 continuation tracks. We compute the embedding of all the tracks by averaging their embeddings computed on non-overlapping frames with the mean statistic. Then, for each track in a playlist, we compute the 100 most similar tracks, among the ones from the test set. These tracks are obtained using the cosine similarity in the embedding space.[4] Among all the retrieved similar tracks for a playlist, we finally select the 100 most repeated ones. We compare these to the ground truth using normalized Discounted Cumulative Gain (nDCG) and Mean Average Precision (MAP) [34], which are commonly used to evaluate the performance of music recommendation systems. These ranking metrics evaluate the order of the items for each playlist returned by the prediction. They return a higher score for a given playlist if the predicted ranked list contains items in the test set closer to the top.

IV. RESULTS

Focusing on **genre classification**, the results in Table I show that the performance of the audio embedding when trained using the contrastive loss is always higher than using the models trained directly to predict the modality information (B-line). The best performance is obtained with Contr$_{\mathrm{G}}$ with a similar result to when also considering CF information when training the embedding model (Contr$_{\mathrm{CF\text{-}G}}$). We also see that the performances of the Contr$_{\mathrm{G}}$ model are comparable with state-of-the-art pre-trained embeddings (VGGish audioset) [30], [31]. This is particularly interesting since a large percentage of the Melon Playlist Dataset consists of korean music, which can be different from popular western music from the GTZAN collection.

[3]https://mtg.github.io/mtg-jamendo-dataset/

[4]Similarity searches are computed using Annoy (https://github.com/spotify/annoy) based on Approximate Nearest Neighbors and angular distance [33].

TABLE II
AUTOMATIC TAGGING RESULTS

	ROC AUC $\pm$ STD		
Model	Genre	Mood	Instrument
B-line$_{\mathrm{G}}$	0.840 $\pm$ 0.004	0.722 $\pm$ 0.004	0.781 $\pm$ 0.005
B-line$_{\mathrm{CF}}$	0.836 $\pm$ 0.002	0.722 $\pm$ 0.003	0.770 $\pm$ 0.008
B-line$_{\mathrm{CF\text{-}G}}$	0.845 $\pm$ 0.004	0.727 $\pm$ 0.006	0.785 $\pm$ 0.004
Contr$_{\mathrm{G}}$	**0.847** $\pm$ 0.004	0.732 $\pm$ 0.005	**0.797** $\pm$ 0.005
Contr$_{\mathrm{CF}}$	0.845 $\pm$ 0.004	0.732 $\pm$ 0.004	0.793 $\pm$ 0.007
Contr$_{\mathrm{CF\text{-}G}}$	0.843 $\pm$ 0.004	**0.733** $\pm$ 0.005	0.791 $\pm$ 0.006

TABLE III
PLAYLIST GENERATION RESULTS

Model	NDCG@100	MAP@100
random	0.0005	0.0001
B-line$_{\mathrm{G}}$	0.0044	0.0007
B-line$_{\mathrm{CF}}$	0.0035	0.0007
B-line$_{\mathrm{CF\text{-}G}}$	0.0042	0.0008
Contr$_{\mathrm{G}}$	0.0074	0.0016
Contr$_{\mathrm{CF}}$	0.0076	0.0017
Contr$_{\mathrm{CF\text{-}G}}$	**0.0085**	**0.0020**

Automatic Tagging. From the results in Table II we see that the methods based on contrastive learning outperform the baselines in almost all the cases. The best results for the instrument and genre tags is obtained with the Contr$_{\mathrm{G}}$ model. For the mood tags the best performance is achieved with Contr$_{\mathrm{CF\text{-}G}}$, which takes advantage of the information in the playlists and the genre annotations.

The results for the task of **Automatic playlist continuation** follow the same trend of the other tasks. The models trained using the contrastive loss perform better than the baselines trained directly to predict the genres or the CF representation. The best performance is obtained with the Contr$_{\mathrm{CF\text{-}G}}$ model, which combines genre and CF information.

V. CONCLUSIONS

In this work, we propose a method for learning an audio representation, by combining multiple sources of information related to the music using contrastive learning. We evaluate the method by pre-traing the model using information from the Melon Playlist Dataset and we compare the performance in three downstream tasks in the music domain (genre classification, automatic tagging, and automatic playlist continuation). We see that using contrastive learning allows us to reach higher performance than using the models trained directly to predict the genre or the collaborative filtering information. This indicates that contrastive learning is effective at learning simultaneously from heterogeneous information, enabling us to improve the overall performance across different tasks.

The dataset used for training our embedding model offers additional types of information that we did not use. They include title, playlist tags and authors, as well as other metadata of the tracks. As future work, we propose incorporating this playlist-level information which will require an additional level of abstraction to our architecture.

REFERENCES

[1] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal metric learning for tag-based music retrieval," *arXiv preprint arXiv:2010.16030*, 2020.

[2] Ò. Celma and P. Herrera, "A new approach to evaluating novel recommendations," in *Proceedings of the 2008 ACM conference on Recommender systems*, 2008, pp. 179–186.

[3] F. Korzeniowski, O. Nieto, M. McCallum, M. Won, S. Oramas, and E. Schmidt, "Mood classification using listening data," *arXiv preprint arXiv:2010.11512*, 2020.

[4] S. Oramas, F. Barbieri, O. Nieto, and X. Serra, "Multimodal deep learning for music genre classification," *Transactions of the International Society for Music Information Retrieval. 2018; 1 (1): 4-21.*, 2018.

[5] D. Surís, A. Duarte, A. Salvador, J. Torres, and X. Giró-i Nieto, "Cross-modal embeddings for video and audio retrieval," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 0–0.

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[7] P. Alonso-Jiménez, D. Bogdanov, J. Pons, and X. Serra, "Tensorflow audio models in essentia," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 266–270.

[8] A. Zhai and H.-Y. Wu, "Classification is a strong baseline for deep metric learning," *arXiv preprint arXiv:1811.12649*, 2018.

[9] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Metric learning vs classification for disentangled music representation learning," in *Proc. of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[10] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, 2020.

[11] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification." *Journal of Machine Learning Research*, vol. 10, no. 2, 2009.

[12] J. Choi, J. Lee, J. Park, and J. Nam, "Zero-shot learning for audio-based music classification and tagging," *arXiv preprint arXiv:1907.02670*, 2019.

[13] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.

[15] E. Fonseca, D. Ortego, K. McGuinness, N. E. O'Connor, and X. Serra, "Unsupervised contrastive learning of sound event representations," *arXiv preprint arXiv:2011.07616*, 2020.

[16] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," *arXiv preprint arXiv:2010.10915*, 2020.

[17] X. Favory, K. Drossos, T. Virtanen, and X. Serra, "Coala: Co-aligned autoencoders for learning semantically enriched audio representations," in *International Conference on Machine Learning (ICML), Workshop on Self-supervised learning in Audio and Speech*, 2020.

[18] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *arXiv preprint arXiv:2004.11362*, 2020.

[19] X. Favory, K. Drossos, T. Virtanen, and X. Serra, "Learning contextual tag embeddings for cross-modal alignment of audio and tags," *arXiv preprint arXiv:2010.14171*, 2020.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, pp. 5998–6008, 2017.

[21] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *International Conference of Learning Representations (ICLR)*, 2013.

[22] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," 2011.

[23] A. Ferraro, Y. Kim, S. Lee, B. Kim, N. Jo, S. Lim, S. Lim, J. Jang, S. Kim, X. Serra, and D. Bogdanov, "Melon playlist dataset: a public dataset for audio-based playlist generation and music tagging," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*, 2021.

[24] K. Sechidis, G. Tsoumakas, and I. Vlahavas, "On the stratification of multi-label data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 145–158.

[25] A. Ferraro, D. Bogdanov, X. S. Jay, H. Jeon, and J. Yoon, "How low can you go? Reducing frequency and time resolution in current CNN architectures for music auto-tagging," in *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 131–135.

[26] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of CNN-based automatic music tagging models," in *Proceedings of the SMC2020 - 17th Sound and Music Computing Conference*, 2020.

[27] 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.

[28] C. Kereliuk, B. L. Sturm, and J. Larsen, "Deep learning and music adversaries," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2059–2071, 2015.

[29] 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.

[30] J. Pons and X. Serra, "Musicnn: pre-trained convolutional neural networks for music audio tagging," in *Late-breaking/demo session in 20th International Society for Music Information Retrieval Conference (LBD-ISMIR2019)*, 2019.

[31] 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.

[32] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The MTG-Jamendo dataset for automatic music tagging," in *Proceedings of the Machine Learning for Music Discovery Workshop, 36th International Conference on Machine Learning, ML4MD at ICML 2019*, 2019.

[33] S. Dasgupta and Y. Freund, "Random projection trees and low dimensional manifolds," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 537–546.

[34] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

7.15 Dynamic Split Computing for Efficient Deep Edge Intelligence

The appended preprint follows.

Dynamic Split Computing for Efficient Deep Edge Intelligence

Arian Bakhtiarnia [1] **Nemanja Milošević** [2] **Qi Zhang** [1] **Dragana Bajović** [3] **Alexandros Iosifidis** [1]

Abstract

Deploying deep neural networks (DNNs) on IoT and mobile devices is a challenging task due to their limited computational resources. Thus, demanding tasks are often entirely offloaded to edge servers which can accelerate inference, however, it also causes communication cost and evokes privacy concerns. In addition, this approach leaves the computational capacity of end devices unused. Split computing is a paradigm where a DNN is split into two sections; the first section is executed on the end device, and the output is transmitted to the edge server where the final section is executed. Here, we introduce dynamic split computing, where the optimal split location is dynamically selected based on the state of the communication channel. By using natural bottlenecks that already exist in modern DNN architectures, dynamic split computing avoids retraining and hyperparameter optimization, and does not have any negative impact on the final accuracy of DNNs. Through extensive experiments, we show that dynamic split computing achieves faster inference in edge computing environments where the data rate and server load vary over time.

1. Introduction

The combination of deep learning and Internet of Things (IoT) has tremendous applications in fields such as healthcare, smart homes, transportation and industry (Ma et al., 2019). However, deep learning models typically contain millions or even billions of parameters, making it difficult to deploy these models on resource-constrained devices. One solution is to offload the computation to an edge or cloud server (Wang et al., 2020), as shown in Figure 1 (b). However, since the size of the inputs to deep learning models can be massive, particularly images and videos, this approach consumes a lot of bandwidth and energy, and leads to delays. Moreover, even though IoT devices are limited, they still possess computational capabilities that remain unused when the entire computation is offloaded, and utilizing these capabilities would reduce the load on the servers. In addition, in applications that process personal data such as health records, or in audio or visual streams with voice activity or human presence, privacy regulations such as European Union's GDPR (European Commission) or United States' HIPAA (Centers for Medicare & Medicaid Services, 1996) may apply. These regulations typically forbid direct access to non-anonymized data, leaving the options to either anonymize the data at the cost of additional computation and higher latency, or process the data at the source.

Split computing, depicted in Fig. 1 (c), alleviates these issues by splitting the deep model into a *head* section and a *tail* section (Matsubara et al., 2021). The head model is executed on the device, and its output (the intermediate representation at that particular layer of the deep network) is transmitted to the server, then processed by the tail model to obtain the final output. In a way, split computing is a *partial offloading* of the computation, as opposed to the *full-offloading* approach. Another benefit of split computing over full-offloading is that it can be used as a privacy preserving technique since intermediate representations are being transmitted instead of the actual inputs, and the original inputs cannot be easily reconstructed from the intermediate representations (Jeong et al., 2018). In addition, split computing can be combined with early exiting in order to obtain an early result on the device (Scardapane et al., 2020; Matsubara et al., 2021), as illustrated in Figure 1 (c), which is useful when transmission takes longer than expected.

Since split computing aims to decrease the communication cost, *natural bottlenecks*, which are the layers of the deep network where the size of the intermediate representation is smaller than the input size, can be used as splitting points for deep learning models. In this paper, we show that unlike older popular models, state of the art models such as EfficientNet (Tan & Le, 2019; 2021) possess many natural bottlenecks. Based on this fact, we propose a method called *dynamic split computing* where the best splitting point is automatically and dynamically determined based on input and channel conditions, as shown in Figure 1 (d). Since the underlying deep learning model is not modified, dynamic

[1]DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark [2]Faculty of Sciences, University of Novi Sad, Serbia [3]Faculty of Technical Sciences, University of Novi Sad, Serbia. Correspondence to: Arian Bakhtiarnia <arianbakh@ece.au.dk>.

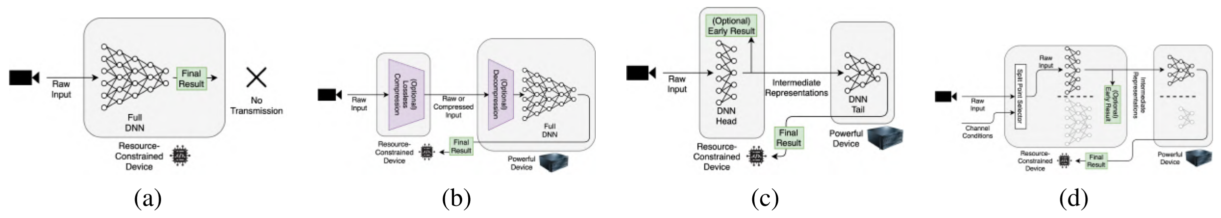


Figure 1. Overview of (a) no-offloading; (b) full-offloading; (c) split computing; and (d) dynamic split computing approaches.

split computing can be used as a plug-and-play method, meaning it can be employed without domain knowledge about the particular deep learning models that are being used. It is important to note that dynamic split computing is a complimentary efficient inference method that can be used in combination with other approaches, including model compression techniques such as pruning and quantization (Choudhary et al., 2020), as well as dynamic inference methods such as early exiting (Bakhtiarnia et al., 2021).[1]

2. Related Work

Several approaches for speeding up the inference of deep neural networks (DNNs) on resource-constrained devices exist in the literature. *Local computing* performs the entire computation on the device, yet modifies the architecture of the neural network in order to decrease the required computation, while causing a minimal negative impact on the accuracy. *Lightweight models* such as MobileNet (Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019) are specifically designed to be deployed on such limited devices, whereas *model compression* techniques (Cheng et al., 2018) alter existing architectures in order to make them more lightweight, for instance, *pruning* removes the less impactful parameters (weights) of the neural network; *quantization* uses less bits to represent each parameter (Liang et al., 2021); and *knowledge distillation* aims to train a more compact model to reproduce outputs similar to a given larger neural network (Gou et al., 2021).

Dynamic inference methods (Han et al., 2021) can alter the architecture of existing neural networks to adapt their inference time at the cost of accuracy, meaning they will produce more accurate outputs the longer they are allowed to execute. Various approaches to dynamic inference exist, such as *early exiting* (Scardapane et al., 2020), where early exit branches are added after intermediate layers of a DNN that produce an output similar to the final output; *layer skipping* (Graves, 2016; Banino et al., 2021; Wang et al., 2018), where the execution of some of the DNN layers are skipped; and *channel skipping* (Gao et al., 2019), where less impactful channels of convolutional layers are ignored.

Even with local computing techniques, many high-performing DNNs exceed the computational capacity of devices, especially when the output is expected within a strict deadline. In such cases, the computation can be offloaded to external servers. When the computation of a DNN is offloaded, the inputs must be transmitted from a device to a server, yet this can introduce massive delays during data transmission, particularly when the input size is large, which may defeat the original purpose of speeding up the computation. This has led to a recent emerging paradigm called *edge computing* (Abbas et al., 2018) where the computation is offloaded to *edge servers* located much closer to end devices compared to cloud servers which are often located in data centers. Even though edge computing reduces the transmission delay, it still has some drawbacks. First, since the original inputs are being transmitted over a network, privacy issues arise. Furthermore, since typically multiple end devices are connected to the same edge server, if all of them offload their computation simultaneously, the edge server may experience a high load while the computational resources of each end device remain unused.

Split computing (Matsubara et al., 2021) (also known as *collaborative intelligence*) is an alternative approach that provides a balance between local computing and full-offloading, where some layers of the DNN are executed on the end device and the intermediate output is then sent over to the edge server where it is processed by the rest of the DNN layers. When the splitting point is chosen such that the size of the intermediate representation is lower than the input size, the transmission delay will consequently be lower than that of full-offloading.

However, not all deep learning models possess such natural bottlenecks, and even if they exist, they may be located in the final layers of the network where the bulk of the computation has already been carried out, and therefore it would not be sensible to offload the remaining computation. For instance, widely used models such as ResNet (He et al., 2016) and Inception (Szegedy et al., 2016) do not contain natural bottlenecks in their early layers (Matsubara et al., 2021). In such cases, *bottleneck injection* can be used, where the architecture of the network is modified to artificially insert a bottleneck (Matsubara et al., 2021). However, this approach requires time-consuming operations such as retraining the

[1]Our code is available at `https://gitlab.au.dk/maleci/dynamicsplitcomputing`.

model and optimizing hyperparameters such as the size of the inserted bottleneck. Furthermore, there is no guarantee that the new architecture can obtain an accuracy comparable to that of the original architecture, particularly when a limitation such as a small bottleneck is imposed. Therefore, bottleneck injection is far from ideal.

3. Dynamic Split Computing

We assume a trained high-performing DNN is to be deployed on a device with access to a server, where the data rate of the communication channel and the number of inputs in the batch (batch size) may vary. The variations in the data rate may be due to fluctuations in wireless channel state or traffic congestion, and the variations in batch size may occur due to a different workload at different times. The goal of our method is to optimize the end-to-end inference time by dynamically detecting the best splitting point for a given DNN based on the communication channel state and batch size. Since we aim to design our method in a "plug-and-play" manner, such that it can be deployed in edge computing systems without creating new trade-offs involving the accuracy or the hassle of retraining, we avoid altering the underlying architecture or any lossy compression techniques that may affect the accuracy of the final result.

Formally, neural networks can be formulated as $f(x) = f_L(f_{L-1}(\dots f_1(x)))$ where x is the input, L is the total number of layers in the neural network and f_i is the operation performed at layer i. The intermediate representation at layer i, which is the output of the i-th layer is recursively formulated as $h_i = f_i(h_{i-1})$ where $h_0 = x$ is the input. Based on this notation, with split computing at layer j, the head and tail parts of the DNN are denoted by $f^h(x) = f_j(f_{j-1}(\dots f_1(x)))$ and $f^t(h_j) = f_L(f_{L-1}(\dots f_{j+1}(h_j)))$, respectively, and h_j is the intermediate representation that is transmitted.

The first step is to find the natural bottlenecks of the DNN by calculating the compression ratio $c_l = |h_l|/|x|$ for each layer l where $|h_l|$ and $|x|$ denote the size of intermediate representation at layer l and the input size, respectively. If $c_l < 1$, layer l is a natural bottleneck of the DNN. However, not all natural bottlenecks are useful in split computing. We define $T^h_{i,j}$ and $T^t_{i,j}$ as the inference time from layer i up to and including layer j $(i < j)$ of the deep neural network measured on the device and the server, respectively. When layers m and n $(m < n)$ have the same compression ratio, in other words when $c_m = c_n$, the total end-to-end inference time with split computing at layer m and layer n are

$$T_m = T^h_{1,m} + c_m T_{\text{FO}} + T^t_{m+1,n} + T^t_{n+1,L}, \quad (1)$$

$$T_n = T^h_{1,m} + T^h_{m+1,n} + c_n T_{\text{FO}} + T^t_{n+1,L}. \quad (2)$$

where T_{FO} is the transmission time of the entire input in full-offloading. Assuming the computational resources of the server are greater than that of the device, then $T^h_{m+1,n} > T^t_{m+1,n}$, thus it is favorable to choose the earlier layer as splitting point. Consequently, only natural bottlenecks with compression ratio lower than all previous natural bottlenecks are useful. We call such bottlenecks *compressive*. Compressive natural bottlenecks are defined by

$$C = \{j | c_j < 1, c_j < c_i \ \forall i < j\}. \quad (3)$$

The total end-to-end inference time for a given batch of inputs when the splitting point of the network is l is

$$T_l = T^h_{1,l} + \frac{D\, c_l}{r} + T^t_{l+1,L}, \quad (4)$$

where D is the data size of the original input, c_l is the compression ratio of the intermediate representation at layer l and r is the data rate of the communication channel. When inputs are images or video frames, the total load in bytes can be calculated as $D = BWHC$, where B is the batch size, W and H are the width and height of the images, and C is the number of channels in the images, for instance, $C = 3$ for color images and $C = 1$ for grayscale.

We define the end-to-end inference time in case of no-offloading as $T_L = T^h_{1,L}$ and in case of full-offloading as

$$T_0 = \frac{D}{r} + T^t_{1,L}. \quad (5)$$

Therefore, the optimal splitting point s_{opt} can be determined by optimizing for

$$s_{opt} = \underset{l \in \{0 \dots L\}}{\arg\min}(T_l). \quad (6)$$

Dynamic split computing finds the optimal split location for a given data rate and batch size based on Eq. (6). When full-offloading cannot be used, for instance, due to privacy requirements, the range is Eq. (6) is reduced to $\{0 \dots L-1\}$. Note that based on previous arguments, only compressive natural bottlenecks need to be investigated, therefore once all compressive natural bottlenecks are identified, we calculate the optimal splitting point for each batch size and data rate by measuring the inference time of head and tail models for each compressive bottleneck. It is important to note that the relationship between inference time of head or tail model and batch size is not strictly linear, therefore it needs to be measured for each batch size. Additionally, when the data rate is too low, it may not be sensible to use any form of offloading since it introduces too much delay. Therefore, dynamic split computing considers the no-offloading option alongside the optimal splitting point and switches between split computing and no-offloading when necessary.

Since different applications and environments may have different ranges for data rate and batch size and a unique pattern for their variations, we need a method to measure how beneficial dynamic split computing is in each specific case. We define a scenario as a sequence of the state of the environment throughout time, i.e., $S = ((B_1, r_1), (B_2, r_2) \ldots, (B_T, r_N))$, where B_i and r_i are the batch size and data rate at time step i, respectively. The relative average gain of dynamic split computing in terms of end-to-end inference time over a specific method, for instance, static split computing at a specific location, can then be calculated by

$$G = \frac{1}{N} \sum_{1 \le i \le N} \frac{|T_{s_{opt}}(B_i, r_i) - T^{\text{SS}}(B_i, r_i)|}{T^{\text{SS}}(B_i, r_i)}, \tag{7}$$

where $T_{s_{opt}}(B_i, r_i)$ and $T^{\text{SS}}(B_i, r_i)$ are the end-to-end inference time using dynamic and static split computing, respectively, with batch size B_i and data rate r_i.

4. Results

We investigate 14 modern DNN architectures: seven variations of EfficientNetV2 (Tan & Le, 2021) and seven variations of EfficientNetV1 (Tan & Le, 2019). All these architectures were originally designed for image classification and have since been applied to various other problems such as speech recognition (Lu et al., 2020). The accuracy of these architectures on the ImageNet dataset (Deng et al., 2009) ranges from 77.1% to 85.7%.

First, we find the compressive natural bottlenecks for each architecture. The number of natural bottlenecks in these architectures ranges from 15 to 68, three to four of which are compressive. For comparison, VGG-16 (Simonyan & Zisserman, 2014), which is an older architecture, has only 5 natural bottlenecks. Subsequently, we find the optimal splitting point for each architecture in a wide range of states. We check data rates ranging from 1 MBps to 128 MBps and batch sizes of 1 to 64. Some larger models such as EfficientNetV2-L run into memory issues with large batch sizes, therefore, we reduce the maximum batch size to 32 or 24 in such cases. For the edge server, we use an Nvidia 2080 Ti GPU, and in order to simulate a resource-constrained device, we underclock the same type of GPU to 300 MHz (the normal GPU frequency is around 1800 MHz).

The results for the EfficientNetV1-B4 architecture are shown in Fig. 2, where for each state (data rate and batch size), the optimal split location derived based on Equation 6 is specified. It can be observed that each compressive bottleneck is an optimal split location in several states. Moreover, no-offloading is the optimal solution in some other states. Therefore, dynamically switching between split locations (as well as no-offloading) based on the state of the communication channel improves inference speed. This is also the case with the other 13 investigated architectures. The relative gain of dynamic split computing over split computing at a fixed location (block 10) for the EfficientNetV1-B4 architecture in terms of inference speed is shown in Fig. 3. This figure can be used to derive the gain of dynamic split computing compared to another method for a specific scenario based on on Equation 7. Notice that in states where split computing at block 10 is optimal, dynamic split computing swiches to this method and thus has no advantage over it, whereas dynamic split computing obtains some gain everywhere else by switching to a different method.

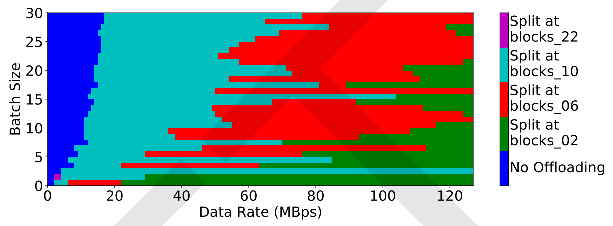


Figure 2. Optimal split location based on batch size and data rate for the EfficientNetV1-B4 architecture.

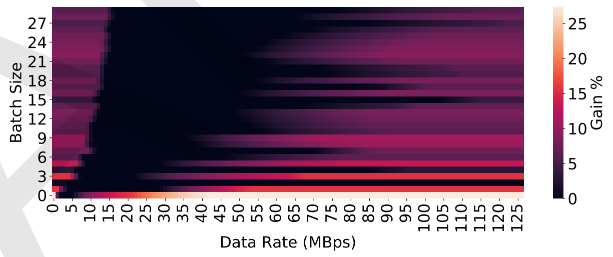


Figure 3. The relative gain of dynamic split computing in terms of end-to-end inference time over static split computing at block 10 in the EfficientNetV1-B4 architecture.

5. Conclusion

In this paper, we showed that dynamic split computing offers improvements in terms of inference time over both no-offloading and split computing with a fixed split location. Moreover, as opposed to full-offloading, dynamic split computing can decrease the computation load on the server by performing parts of the computation on the device. Finally, by transmitting intermediate representations instead of inputs, dynamic split computing circumvents privacy issues that arise when using full-offloading.

Acknowledgements

The work received funding by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, and by the Danish Council for Independent Research under Grant No. 9131-00119B.

References

Abbas, N., Zhang, Y., Taherkordi, A., and Skeie, T. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2018. doi: 10.1109/JIOT.2017.2750180.

Bakhtiarnia, A., Zhang, Q., and Iosifidis, A. Multi-exit vision transformer for dynamic inference. *The 32nd British Machine Vision Conference (BMVC 2021)*, 2021.

Banino, A., Balaguer, J., and Blundell, C. Pondernet: Learning to ponder. *arXiv:2107.05407*, 2021.

Centers for Medicare & Medicaid Services. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Online at http://www.cms.hhs.gov/hipaa/, 1996.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018. doi: 10.1109/MSP.2017.2765695.

Choudhary, T., Mishra, V., Goswami, A., and Sarangapani, J. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 53(7):5113–5155, Oct 2020. ISSN 1573-7462. doi: 10.1007/s10462-020-09816-7. URL https://doi.org/10.1007/s10462-020-09816-7.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

European Commission. 2018 reform of EU data protection rules. Online at https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en.

Gao, X., Zhao, Y., Dudziak, L., Mullins, R. D., and Xu, C. Dynamic channel pruning: Feature boosting and suppression. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=BJxh2j0qYm.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, Jun 2021. ISSN 1573-1405. doi: 10.1007/s11263-021-01453-z. URL https://doi.org/10.1007/s11263-021-01453-z.

Graves, A. Adaptive computation time for recurrent neural networks. *arXiv:1603.08983*, 2016.

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., and Wang, Y. Dynamic neural networks: A survey. *arXiv:2102.04906*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

Jeong, H.-J., Jeong, I., Lee, H.-J., and Moon, S.-M. Computation offloading for machine learning web apps in the edge server environment. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1492–1499, 2018. doi: 10.1109/ICDCS.2018.00154.

Liang, T., Glossner, J., Wang, L., Shi, S., and Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.07.045. URL https://www.sciencedirect.com/science/article/pii/S0925231221010894.

Lu, Q., Li, Y., Qin, Z., Liu, X., and Xie, Y. Speech recognition using efficientnet. In *Proceedings of the 2020 5th International Conference on Multimedia Systems and Signal Processing*, ICMSSP 2020, pp. 64–68, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450377485. doi: 10.1145/3404716.3404717. URL https://doi.org/10.1145/3404716.3404717.

Ma, X., Yao, T., Hu, M., Dong, Y., Liu, W., Wang, F., and Liu, J. A survey on deep learning empowered iot applications. *IEEE Access*, 7:181721–181732, 2019. doi: 10.1109/ACCESS.2019.2958962.

Matsubara, Y., Levorato, M., and Restuccia, F. Split computing and early exiting for deep learning applications: Survey and research challenges. *arXiv:2103.04505*, 2021.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Scardapane, S., Scarpiniti, M., Baccarelli, E., and Uncini, A. Why should we add early exits to neural networks? *Cognitive Computation*, 12(5):954–966, Sep 2020. ISSN 1866-9964. doi: 10.1007/s12559-020-09734-4. URL https://doi.org/10.1007/s12559-020-09734-4.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Tan, M. and Le, Q. EfficientNet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/tan19a.html.

Tan, M. and Le, Q. V. Efficientnetv2: Smaller models and faster training. *arXiv:2104.00298*, 2021.

Wang, X., Yu, F., Dou, Z.-Y., Darrell, T., and Gonzalez, J. E. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Wang, X., Han, Y., Leung, V. C. M., Niyato, D., Yan, X., and Chen, X. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22(2):869–904, 2020. doi: 10.1109/COMST.2020.2970550.