Master Thesis of Sound and Music Computing

Universitat Pompeu Fabra

Acoustic Scene Classification using Convolutional Neural Networks on Multivariate Audio

An Investigation into Limens for Machine Listening

Marc Kaivon Jones

Supervisor: Frederic Font Corbera

Co-Supervisor: Eduardo Fonseca

September, 2019

Contents

Dedication

I dedicate this work to the proverbial village of otherworldly individuals that have helped me every step of the way. With special consideration for my mother and grandmother, whose resilience and sacrifice in the face of adversity made possible the opportunities available to me. I also dedicate this work to the scores of fellow queer and minority scientists carving their path in environments that are far too often hostile, unwelcoming, or antagonistic otherwise: *I see you, you are worthy, you belong, and you are needed, now more than ever.*

Acknowledgement

I want to use this space to express my sincerest gratitude to my advisor Frederic Font Corbera, my co-advisor Eduardo Fonseca, professors Alastair Porter and Xavier Favory, along with the rest of the Audio Signal Processing Lab at UPF's Music Technology Group for all their patience and direction throughout this journey. Furthermore, I am forever appreciative of the support of my family, friends, and cohort over the last two years. Without everyone's guidance and steadfast encouragement none of this would have been possible.

Thank you,

Kaivon

Abstract

Research has shown the efficacy of using convolutional neural networks (CNN) with audio spectrograms in machine listening tasks such as acoustic scene classification (ASC). There is, however, a knowledge gap when it comes to standardizing preprocessing practices for this form of ASC. Researchers using these methods have been moving forward in relative darkness about how to best represent their audio data for consumption by a CNN, often relying on transfer learning from adjacent machine listening tasks. This work explores the relationship of frequency limens and channel depth on ASC accuracy with CNNs of three different varieties: generic, deep, and wide. Results show that variability in the representation of spectral audio information plays a crucial role in classifier performance. Classification accuracy improved when using multi-channel representations of audio data over a single channel alternative. Classification accuracy also decreased when the representative spectra contained less frequency information, albeit to a lesser degree. This pattern was nearly consistent across each of the proposed CNN architectures. These findings have direct implications for several academic and industrial machine listening applications. In the academic realm, they work towards codifying audio data preprocessing practices and network architectural decisions. In industry, the results open the door for exploring the usage of substandard microphones in technologies that employ machine listening such as commodity hardware.

Keywords: acoustic scene classification; computational auditory scene analysis; machine listening; convolutional neural networks, commodity hardware;

Chapter 1

Introduction

"The proselytisation of machine learning has preached its endless applicability; the idea that all phenomenon will be explicable with enough data and enough computational power professes to have no limitation." -Martin Disley

1.1 Background and Motivation

The era of big data is upon us. Machine learning is powering technological advancements in nearly every facet of our electronically connected world. Its influence permeates into all corners of science, from autonomous systems to computer vision to artificial intelligence assisted healthcare to smart home and other internet-of-things (IoT) devices. These are just some of the many innovative technologies, all built upon the marriage of machine learning and big data. From a layman's perspective, machine perception is perhaps one of today's more recognizable forms of machine learning. Machine perception most notably comes in the form of computer vision[1] and machine listening[2].

[1] image and video recognition

[2] recognition of sounds such as voice or music

A burgeoning subgenre of machine listening is Computational Auditory Scene Analysis (CASA) - "the study of auditory scenes by acoustic means," [1] where auditory (or "acoustic") scenes refer to taxonomized contexts understood through psychophysical descriptors, e.g., in nature, at a sports arena, in the kitchen, on a bus, etc. While researchers have been working on CASA related tasks for decades, the community has received a significant amount of interest in the last five years or so. The advancement of acoustic event and scene analysis has been propelled by its deployable industrial value be it in surveillance, healthcare industries, context-aware applications, or a host of other sectors that require advanced media retrieval [2]. Evidence of the growing interest in CASA research can not only be found from indicators like the rise of public evaluation campaigns or the increase of research community growth but also in the participation of significant industrial players [3, 4, 5].

One field of CASA research with high levels of engagement is acoustic scene classification (ASC) [6]. Increased involvement in ASC research has led to a compilation of ASC related datasets ripe for analysis. Although none are perfect, or perhaps as comprehensive as some of the foundational datasets that fueled the proliferation of computer vision [7, 8, 9], each has improved with time, helping invite more of the research community to participate in machine listening tasks. At the forefront of CASA public evaluation campaigns is the now-annual Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge. Using their acoustic scenes dataset I have set out to answer the following question: *can a machine learn to accurately classify an acoustic scene when presented with substandard audio?* Breaking down this question from a macro level reveals a number of smaller problems that need to be reconciled: finding an appropriate machine learning architecture, defining "substandard audio", identifying a means of extracting relevant information from said audio, determining evaluation criteria, and finally implementing it all. The answers to these questions have profound implications on the production of machine listening systems designed to be contextually aware. Considering the rampant growth of IoT devices, context-aware applications, and "smart"

commodity hardware in the marketplace, this research is relevant to any stakeholder with ties to these technologies. Understanding the threshold of audio information necessary to successfully classify acoustic scenes has ramifications that span from greater efficiency, such as improved machine learning processes, to economic incentives, in the form of reduced computational resources or product design decisions.

1.2 Organization

With the intention of elucidating untrodden territory in ASC research, making plainly clear the processes of this work, and relaying its importance, the remaining contents of this study are structured in the following manner: 2) State of the Art - an exhaustive summary, cataloging the current state of affairs of ASC research, telling its story from past to present towards future implications, highlighting the technologies and techniques used along the way. 3) Methodology - a look at the procedures taken to create a viable research environment. These include the approach to sourcing relevant data, defining the problem set, data preprocessing techniques, the proposed system architecture along with the various parameters and technologies needed to make its implementation possible. 4) Experiment - a detailed survey into the investigation undertaken, conveying the expected outcomes, all while accounting for every variable and parameter tuned in the process. 5) Discussion - an examination of the experiments' empirical findings, communicated in a variety of contexts, e.g., graphical, tabular, and written. 6) Conclusions - closing comments on the findings together with their shortcomings, practical implications, and actionable insights on both future work and industrial value.

Chapter 2

State of the Art

2.1 Acoustic Scene Classification

The DCASE community formally defines Acoustic Scene Classification as "[the] recognition of the environment in which a recording has been made, relying on the assumption that an acoustic scene, as a general characterization of a location or situation, is distinguishable from others based on its general acoustic properties" [10]. From a computational standpoint, ASC is a multi-label classification problem, one that looks to identify the contextual information of a given audio clip. As its industrial value has increased, so have the efforts towards solving this problem. Proof of these endeavors is found at the heart of the DCASE community, which has served as one of the most salient congregations of ASC research to-date. Starting as an Institute of Electrical and Electronics Engineers challenge in 2013, DCASE sought to help the audio signal processing (ASP) community in advance its research in a uniform manner. Since its inception, DCASE has successfully elicited hundreds of academic papers introducing innovative solutions working towards an optimal acoustic scene classifier [11].

The first DCASE challenge submissions were very much of their time, using the best-known practices with what data was available. Often the technology and procedures

used in these early proposals were procured from proximate research circles working in a similar domain, but wholly different problem set: speech recognition. Procedurally, one of the hallmarks of state-of-the-art speech recognition was its approach to feature engineering[1]. Speech processors relied on extracting audio level features, e.g., low-level, tonal, energy, spectral, and cepstral information along with some of their derivatives. As a result of these pre-existing solutions, the representations of acoustic scenes in ASC research followed suit [12]. Early challenge submissions featured classifiers built atop Gaussian Mixture Models (GMM), Hidden Markov Models (HMM), and Support Vector Machines (SVM); with the latter showing out as one of the best performers. As the access to improved computational power and strongly labeled audio data has been made available, ASC task solutions have evolved accordingly.

2.2 Classifiers Used in Early ASC

Hidden Markov Models are statistical models founded upon the Markov chain used to predict a sequence of state changes. The term "hidden" refers to the notion that the Markov processes are unobserved, indicating that its output, not the states, are available to the external observer. The model itself is a finite set of states generally associated with a probability distribution, wherein transition probabilities determine each state change. The predictive abilities of the model are informed by its state transition probabilities, which alongside the emission, or "output" probabilities, effectively serve as its learnable parameters for classification.

Gaussian Mixture Models are probabilistic models that treat each point of input data as if it came from a Gaussian distribution. In doing so, it clusters data in accordance with what it believes to be alike. The GMM works similarly to the much simpler K-means clustering [13]: they both look for latent variables[2] to cluster into classes, but unlike K-means, the GMM incorporates covariance found within the data,

[1] i.e., the process of extracting relevant variables from raw data for machine learning

[2] variables that are not directly observed, but instead inferred from their observable counterparts

all while accounting for a mixture of unimodal Gaussian distributions[3]. In the context of a classification task, a trained GMM classifies examples based on the cluster with which it most closely aligns. GMMs are often used in unsupervised learning tasks but are also applicable in other domains.

Support Vector Machines work to find a hyperplane[4] that capably separates classes in a multidimensional feature space. In the context of a classification task, SVMs create a model representation of the input data such that each class is grouped with as much distance between them as possible[5]. When examples are introduced for classification, they are projected into the model and predicted based on their proximity to predetermined class boundaries. Given this trait, SVMs are typically employed in supervised learning tasks.

All of these classifiers were popular choices because they had been successfully used in other audio related tasks [14]. Through transfer learning, it was reasonable to assume that they might exhibit some efficacy when presented with an adjacent classification problem. These models were ideal because they could perform a classification task without the massive amount of data needed for effective deep learning[6]. The preliminary classifiers scratched the surface of ASC's potential given their relative computational efficiency and the lack of widely circulated audio datasets (for the explicit purpose of ASC research). Nevertheless, these models were not without their faults. HMMs require manual selection of optimal parameters for it to be useful, a process that involves a significant amount of trial and error. GMMs are generally limited to problems that don't include a high number of dimensions, meaning that as the number of components the GMM is asked to cluster increases, its effectiveness decreases. SVMs also require fine-tuning parameters to be effective, with the understanding that those same parameters

[3] often referred to as a "bell curve"

[4] a subspace whose dimension is one less than that of its ambient space, for example: a 3D ambient space will have a 2D hyperplane

[5] imagine points on a graph with boundary lines drawn between each cluster of categories

[6] "deep learning" refers to training [sometimes very large] neural networks; any network with more than 2-layers, input layer excluded, can be considered a "deep" neural network

may not readily translate to another classification task. With these limitations in mind, the ASC research community began a gradual shift away from these technologies, towards those powering the exponential growth in machine perception research. The onset of strongly labeled audio data in CASA facilitated the potency of more complex classifier models, so much so, that the top-performing algorithms in the most recent DCASE challenge were almost exclusively made up of solutions that involved neural networks[7] or ensemble-based classifiers [15].

2.3 The Rise of Neural Networks

Neural networks (NNs), or at least their foundational algorithms, have been around for decades. Only with powerful computers and robust datasets could we leverage their theorems into reality. This work was pioneered in the late 20th century by Geoffrey Hinton, David Rumelhart, and Ronald Williams [16] in their seminal paper, "Learning representations by back-propagating errors", then further evangelized in the new millennium by countless members of the machine learning research community. Without their work, we would not see the applications of neural networks diffusing through everyday modern life. Of all machine perception sub-tasks, computer vision has arguably matured the most[8] in the last ten years. Its growth is directly attributable to the scaling of both robust datasets and enhanced computing power. Therefore, it should be no surprise that with the organization, standards, and resources made available by the CASA research community, ASC work has seen an observable uptick in academic and industrial engagement. As of late, the overwhelming majority of ASC task submissions have incorporated a neural network of some kind. In taking a look at how they work, we can start to understand why they are the popular modality for solving these problems.

Neural networks comprise of mathematical functions called "neurons" (or "nodes") - which are stacked together in successive layers. Neurons accept numerical inputs which,

[7] in the form of multilayer perceptrons, convolutional layers, and recurrent units.
[8] a trend particularly evident in the context of consumer electronics

alongside their respective weights (w) and biases (b)[9], are used to compute an output, often a form of logistic regression. In the context of a neural network, this output is referred to as an "activation." The resulting activations are then passed through a nonlinear function, known as an "activation function"[10].

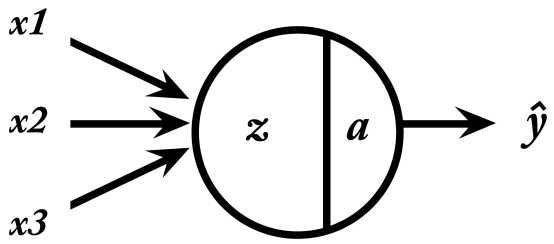


Figure 1: A neuron (or node), with a regression function $z = w^t x + b$ followed by an activation function $a = \sigma(z)$ where $\hat{y}$ is the predicted output

While there are countless possible arrangements for nodes in a neural network, there are almost always at least three requisite layers consistent to a standard NN architecture: the input layer, the hidden[11] layer[s][12], and the output layer[13].

The input layer consumes the training data represented as a set of feature vectors, and outputs - through its activation function - into a hidden layer. In a process known as "forward propagation," continual regression functions are then computed using the successive data feed, where it is sent to the output layer and ultimately classified. Each complex set of operations within the processes of a neural network has its own level of import in terms of successful classification. Therefore every operation needs to be meticulously accounted for when designing a NN based classifier. Below I'll examine some of the essential building blocks that make neural networks useful classifiers.

[9] the learnable parameters in a trained model

[10] Activation functions set the foundation for neural networks to be well equipped to handle nonlinearity (unlike a GMM which requires smoothing, or an SVM that requires the kernel method to achieve nonlinearity).

[11] The term "hidden" refers to fact that during training, the true values of nodes in a given hidden layer are not observed; this means that the expected input and outputs are known, however the processes by which they are computed are effectively a black box.

[12] While only one hidden layer is required, more can help varying problem sets; the greater the amount of hidden layers the "deeper" the network.

[13] this is actually a two-layer neural network, the input layer is not counted

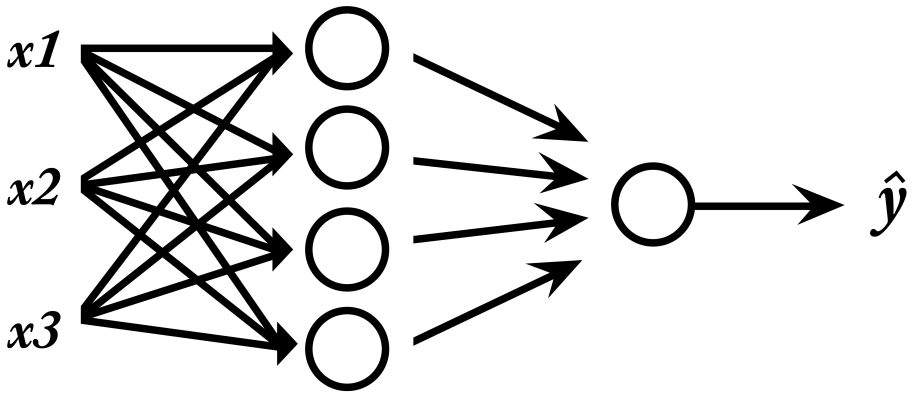


Figure 2: A basic NN with an input layer, hidden layer, output layer (left to right).

Parameters

According to the Universal Approximation Theorem [17], a neural network with at least a single hidden layer *given the appropriate parameters* is capable of representing almost any continuous function, thus making it an effective "universal approximator." Their parameters are the aforementioned weights (w) and biases (b) that neurons rely on for their computations. Fine-tuning these parameters involve a whole subset of hyper-parameters which can influence the learning process.

Loss & Cost Functions

A loss function is computed during forward propagation and represents the error calculated for a single training example of the input data. While a loss function focuses on a single example, a cost function is an indicator as to how the classifier is doing on the whole training set. You can think of these as the distance between how well your classifier is performing, and where you want it to be. Mathematically the cost function is the average of loss functions on the entire training set, measuring the classification performance given the influence of parameters w and b.

Gradient Descent

Gradient descent works to train the parameters, such that the cost function minimizes towards global optima. This process starts at a given point in the intersectional feature

space of the dataset, and quite literally *descends the gradient* towards the global minima[14]. The choice for each progressive step of descent is informed by slope derivatives, where the greatest derivative - or steepest downward slope - is the next best step down towards converging on the minima.

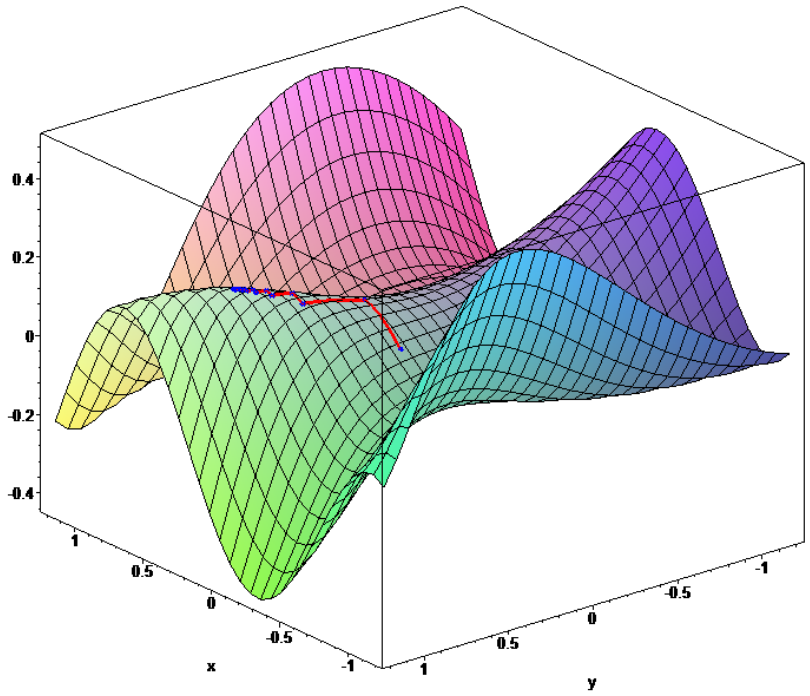


Figure 3: A graphic representation of gradient descent in a multidimensional feature space. Each blue dot references one epoch of training towards global optima [18].

Backpropagation

Backpropagation is used to minimize the loss function by taking the derivative of a function, cognizant of variable dependencies, and identifying which variables need to be modified. Backpropagation breaks down into four distinct steps: the forward pass, the loss function, the backward pass, and the weight update. Mathematically it is the derivative of the forward propagating linear and activation functions, which, when backtraced to the first layer, can adjust the weight parameters accordingly.

[14] i.e., where the cost function is lowest

Hyper-Parameters

A hyper-parameter is a parameter set *before* the training process begins:

Learning Rate

Learning rate, represented mathematically as alpha (α), determines how your parameters evolve. By controlling how much the weights are adjusted relative to the calculated loss, it modifies the slope at which a network-in-training descends the gradient.

Epochs

A single training iteration in a neural network is known as an epoch. This formally includes one round of forward and backward propagation. Finding the right number of epochs needed to train a model can make the difference between creating an accurate classifier versus one that is overfitted[15] and underperforms on the test data.

Units & Layers

The number of neurons in a given layer is referred to as the number of units in that layer, as such the units on a hidden layer are referred to as "hidden units." The number of units on each layer is a reference to the "width" of said layer [19]. Meanwhile, the number of hidden layers between the input and output layers is a reference to the networks "depth", which directly informs the amount of time and power needed to compute an epoch. Although having a single hidden layer allows a neural network to serve as a universal approximator, it must be a very wide network to accomplish this. Therefore there is a real benefit trade-off between network depth and width to consider when designing a neural network-based classifier.

Activation Functions

Activation functions transform the output of a given neuron. Not only do they help avoid the reliance on linear functions, which would effectively introduce identity func-

[15] biased towards the training data

tions to the neural network[16], they simultaneously highlight notable latent variables while working to prevent the vanishing gradient problem[17]. Four of the more commonly used nonlinear activation functions are the hyperbolic tangent function (TanH), the sigmoid function, the rectified linear unit (ReLU), and the leaky rectified linear unit.

$$\textbf{TanH}:\ f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \quad \textbf{Sigmoid}:\ f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\textbf{ReLU}:\ f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad \textbf{Leaky ReLU}:\ f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

While these aren't the only hyper-parameters at play[18], the rest will be attended to as needed in the contexts which make them relevant. Similarly to SVMs and HMMs, hyper-parameters do not readily transfer from task to task, and optimizing this process is still an area of opportunity for research. Relative to the shortcomings of other classification modes, neural networks make for the obvious choice. Their lack of uniform hyper-parameter optimization easy to look past, especially when considering the benefits drawn from having robust data. The most significant benefit neural networks offer is that they improve with the amount of data that they ingest. With large audio-specific datasets made available such as Google's AudioSet, NYU's UrbanSound8K, CHiME Home, and the DCASE ASC set, neural networks have plenty to learn from when solving CASA tasks. Further, as the size of a neural network increases, so does its ability to represent high-level features in the input data. The union of these two resources have shown outstanding classification accuracy compared to the earlier classification methods.

Like all other classifiers, there is not a one-size-fits-all solution to each subset of ma-

[16] combining linear functions will only lead to more linear functions, resulting in a linear classifier

[17] when activation values decrease or increase exponentially as a function of the number of layers in the network

[18] e.g., momentum, batch size, regularization/dropout, normalization, weight initialization, optimization, etc.

chine learning tasks, as different classification problems lend themselves to distinct NN architectures. Large scale data science operations such as recommender systems do well with multilayer perceptrons. Machine translation and other tasks that involve sequenced data are well served by a recurrent neural network such as gated recurrent units or long-short-term memory units. Meanwhile, machine perception tasks such as computer vision have been effectively tackled using convolutional neural networks

2.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a class of neural networks that have found remarkable success in image analysis[19]. Like any standard neural network, CNNs are composed of layers that, through backpropagation, are able to teach learnable parameters to make classifications. The trademark difference of a CNN is in its use of convolutional layers as opposed to fully connected ones. Unlike a multi-layer perceptron, convolutional layers are sparsely connected and its feature vector is represented as a multidimensional[20] matrix of values. For instance, an RGB encoded image can be represented as a stacked set of three two-dimensional matrices, one for each color channel (red, blue, & green), or because it has a single color channel, a grayscale image could be represented as a single two-dimensional matrix. Audio can be similarly represented as an image, making convolutional neural networks an effective classifier choice for ASC work.

CNNs perform calculations on subsections of the input feature vector using a multidimensional grid of values[21] called a "filter" or "kernel." Filters stride[22] across a matrix representing the feature vector, performing element-wise multiplication, then use the

[19] so much so that CNN usage is responsible for the boon of computer vision

[20] expressed as *height* x *width* x *depth* (channels)

[21] By design, the size of a filter must be less than that of the input, therefore it can only process the data in chunks; while the height and width of a filter can vary from the input data, its depth must match the number of channels in the input; though less frequently used, $1x1$ filters have very specific applications.

[22] stride length is a hyper-parameter, along with others such as filter size, # of filters, padding, etc.

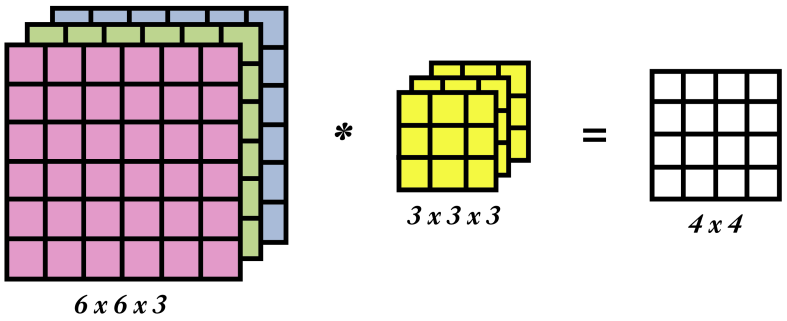


Figure 4: A 3x3x3 filter (with a stride of 1) operating on a 6x6x6 feature vector will produce a 4x4x1 matrix, effectively reducing the dimensionality

product to reduce the matrix dimensionality from layer to layer. Like any other hyper-parameter, there are many possible choices for the best set of values to work with when considering a filter. While there are popular static filters to choose from, there is a greater benefit to learning the best filter set to meet your classification needs. Treating filter values as a parameter introduces the ability for backpropagation to find an optimal set of numbers which best captures the statistical dynamism of your data. Additionally, unless a chosen filter size & stride combination works out to equally cover every element in the feature vector, there is a chance that some values will be over-represented while others go under-represented. This pitfall can be accounted for by adding a border of constant values to the outer edges of the input matrix, in a process known as "padding." The use of padding helps capture additional features found in these bordering spaces. Utilizing multiple filters can help capture features at a varying level of granularity, their shapes and sizes ultimately influencing the volume of its successive layers.

By design, the first layer in a CNN is always a convolutional layer, while successive layers in can come in the form of either more convolutional layers, pooling layers[23], or fully connected layers. Fully connected layers are flattened into a single dimension akin to those found in a multilayer perceptron. Alternatively, pooling layers reduce the size of the feature vector in successive layers. Once the network finds a feature in

[23]also referred to as a "downsampling" layer; formally a convolutional layer immediately followed by a pooling layer composes one full layer in a CNN

the original input volume[24], its location in the input matrix becomes less important than its relative position to the other features. Downsampling effectively reduces the computation time in successive layers by whittling the number of parameters while also controlling for overfitting on the whole feature vector.

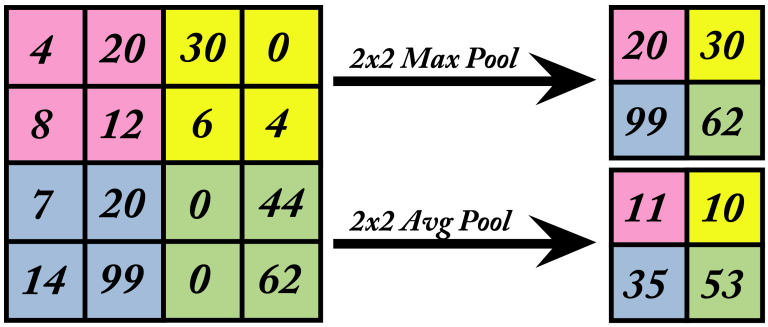


Figure 5: An example of *max-pooling*, where the highest value is passed onto subsequent layers (winner takes all), compared to an example of *average-pooling*, where the mean of values captured are passed on.

Conventional downsampling techniques include max-pooling and average-pooling. Max-pooling takes the most significant value seen by the filter and projects it into a smaller successive layer, while average pooling takes the mean of the numbers seen by the filter. Although pooling layers have no learnable parameters, they have three[25] hyper-parameters: filter size, stride, and type of pooling (average or max); each of which plays a role in determining the success of a CNN's classifying abilities.

In the context of building out an acoustic scene classifier, CNNs make for a clear choice because the network architecture has proven itself in the context of machine perception. It has been observed as one of the top-performing networks in the DCASE ASC challenge year after year. When comparing it to other neural network classes, CNNs are particularly advantageous in that they often feature fewer parameters. CNNs accomplish this through two key attributes: *parameter sharing* and *sparse connectivity*. Parameter sharing occurs when a parameter that is useful in finding a feature in one part of the input matrix is also valuable for other regions. Sparse connectivity refers to

[24] by virtue of a high activation value
[25] four if you include the rarely modified padding size on a pooling layer

the fact that in a convolutional layer, output values to the successive layer are dependent only on what is captured by the filter over each stride. This means that the element-wise product is not calculated on the entire input matrix at once. These attributes allow for robust classification without necessarily more computations being performed to accomplish it.

2.5 Areas of Opportunity for Research

Over the years, the increased complexity of solutions submitted to the ASC task of the DCASE challenge has highlighted a tendency of researchers to throw as much computing power towards the problem as one can afford. The trend of eschewing computational efficiency, for maximal classification accuracy is inspiring, but ultimately unrealistic in terms of its practicality for a large subset of real-world applications, especially for firms and researchers with limited computational resources at their disposal. Recently work has been done to show the efficacy of wide, but shallow, convolutional neural networks [20] on ASC tasks. This network architecture provides a reduced need for computational power and is implementable for systems that have hardware limitations such as embeddable devices or commodity hardware. A classification system that can be embedded on inexpensive equipment could be incredibly beneficial to those who seek to reduce production costs. For instance, if a machine listening device can employ a low-fidelity microphone, but still maintain a respectable classification accuracy comparable to more advanced systems, the producers of such a device would be inclined to take the lower cost measure, using software fixes to facilitate the shortcomings of the hardware instead.

The need for this work has been noted by the community as there is a lack of research conducted to understand the causal relationship between audio degradation and classification accuracy. The DCASE challenge has attempted to simulate the conditions of commodity hardware by providing filtered and resampled audio for a sound event

detection task[26], however, this has yet to be applied to ASC. There are a wide variety of audio preprocessing practices employed by CASA researchers that are of direct interest to finding some clues towards understanding this relationship. Common practices include downmixing the audio to one channel, or downsampling the audio such that the full range of audible frequencies is not represented in their totality prior to classification. While work has been done showing that the entire audible range of audio information[27] is not needed for certain advanced audio-related tasks [21], recent studies have shown that some of these practices may have detrimental effects on the classifiers performance [22, 23, 24].

Given the direction ASC research is heading, I've set out towards creating a system that emulates conditions of relevant interest to stakeholders in this field. This work seeks to understand the relationship between audio quality and classification performance.

[26]sound event detection, or SED, is similar to ASC in that they both analyze acoustic scenes, but SED also seeks to identify the temporal cues of a distinct sound event

[27]i.e., the breadth of channels and frequency range available in a given piece of audio

Chapter 3

Methodology

3.1 Defining a Standard

Before jumping into the weeds of the problem set, we must first define "substandard," meaning we must also identify the standard from which it deviates. In determining the limens above which a neural network can be an effective acoustic scene classifier, we look to the perceptible qualities of digitally represented sound and define what is considered standard, or "full-range" audio. For our purposes, *full-range* is representative of the totality of sound perceivable by a fully-abled human. The human experience of sound can be detailed with variables like *volume*, *channels*, and *frequency range* [25]. Volume, in an auditory context, is the degree of loudness (or intensity) of a sound. Volume is often represented in terms of decibels, a wholly subjective mode of expression relative to the scale from which it is derived. In an acoustic context decibels[1] are most commonly in reference to sound pressure level[2] (or dB SPL) relative to the threshold of human hearing, however, in a digital context decibels are a measurement of waveform's amplitude. Audio channels reflect the number of directions from which a single sound is

[1]Decibels (dB) are fundamentally ratios, therefore they're only intelligible when it is understood what they are in reference to, e.g., dB SPL does not represent the same ratio as dbV, or dbSWL, etc.

[2]in the air

recorded. Mono audio is indicative of a single audio channel, while stereo and binaural audio indicates the presence of two channels: left and right[3]. Frequency is a reference to the pitch of the sound and is measured in Hertz[4] (Hz). High pitched sounds will resonate at a higher frequency while low pitched sounds will do so at lower frequencies. Studies have shown that a healthy, fully developed human can hear frequencies between 20 Hz and 20 kHz. Recording technology allows us to capture audio into the digital realm within the contexts of such variables. Given these parameters of perception, our "standard" is the aforementioned full range. Therefore any audio, or representation thereof, that does not capture this full range can be deemed *substandard*.

3.2 Data Source

As discussed, a large acoustic scene dataset is needed to accomplish the goal of training a NN-based classifier for ASC work. While other source material was available from institutions such as the previously mentioned Google AudioSet, NYU's UrbanSound8K dataset, and the Real World Computing Group's Sound Scene Database, I found the 2019 DCASE ASC task[5] dataset to be the most robust in its offerings. The audio data provided comes from a controlled setup that includes full-range binaural recordings, is neatly organized, and is the most recently updated[6]. In the development set there are 40 hours worth of accurately labeled recordings of acoustic scenes, taxonomized into the following 10 categories: "airport," "indoor shopping mall," "metro station," "pedestrian street," "public square," "street with medium level of traffic," "traveling by tram," "traveling by bus," "traveling by underground metro," and "urban park." Each scene contains recordings from several different cities, providing tonal variability to the training data. There are a total of 14,400 ten-second segments in the development set,

[3]While multi-channel audio greater than two channels exists in a number of capacities, the overwhelming majority of digitally represented audio comes in 1-ch or 2-ch format.

[4]number of cycles per second, where 1 Hz = one cycle per second

[5]The ASC challenge includes various sub-tasks, this work uses the set [26] relevant to "task 1A", however in prior iterations of the challenge this was known simply as "task 1."

[6]as of this writing

evenly divided such that each scene has 1,440 samples, totaling 240 minutes of audio per scene. Given its size, preparation, and organization, the DCASE ASC dataset very directly meets the needs of this work given the scope of the research question. The sheer size of Google's AudioSet made it a compelling alternative option. However, their audio comes from YouTube; therefore, they have no control mechanism in place to standardize recording quality. Using audio from a variety of recording setups risks allowing the classifier to key into differences from recording conditions as opposed to the intended audio scene properties. Considering my proximity to researchers who have either participated in previous challenges or are regularly engaged with the DCASE community, working with a DCASE affiliated dataset was an easy choice.

3.3 Preprocessing & Organization

We'll use ASP domain knowledge to engineer an optimal representation of the audio data. Although optimal classification accuracy isn't guaranteed, a well-tailored data model can help eliminate margins for error throughout the experimental process. Preparing the dataset for consumption by a convolutional neural network involves a host of preprocessing steps. To represent the audio appropriately, we must first extract its features to generate an image from the sound. Luckily, numerous works have demonstrated this process by generating a spectrogram from audio. Spectrograms serve as a visual representation of a given signal's frequency as it varies over time. We can scale the frequency axis logarithmically to better represent the importance of frequency ranges most relevant to the human ear. This step helps capture the nuances of a sound using a spectrogram. The resulting graphic can be described just like an image represented by a multi-dimensional matrix of numerical values. Instead of its height, width, and depth corresponding to pixel (h) x pixel (w) x color channels (d), the spectrogram's matrix is measured as frequency (h) x time (w) x sound channels (d). In this type of "image," mono audio is represented as $d = 1$ while stereo or binaural audio is represented as $d = 2$.

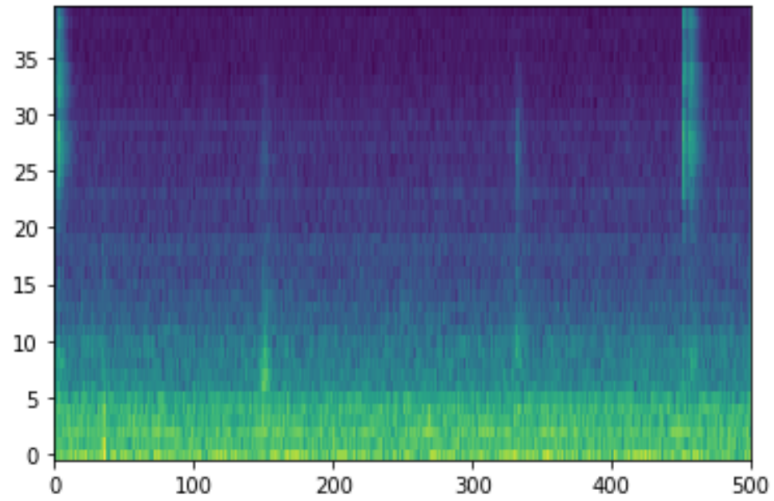


Figure 6: Example 40-band log-Mel TxF spectrogram; x = time, y = frequency.

To understand the limes for a CNN-based classifier, we'll need to simulate a variety of audio conditions. Simulations must include two-channel audio represented in its original recording state as well as downmixed to mono, along with set constraints at several frequency ranges. I've chosen to compare full frequency range audio[7], against audio filtered down[8] to 10 kHz, and even further down to 1 kHz. 10 kHz was an acceptable boundary because of the wide availability of low-budget embeddable microphones that detect only up to this value. 1 kHz was selected as the lower boundary because of the tremendous amount of potentially valuable auditory information that occurs just above 1 kHz [27]. Motivated by similar work in this arena, we'll use Essentia [28] to apply a short-time Fourier transform with 50ms Hamming windows at 50% overlap to create the input spectrogram. The resulting values are then processed through a Mel-filter bank across 40 bands[9]. This step helps us capture the full audible range[10] logarithmically. The spectrogram is then split into equidistant non-overlapping time x frequency (TxF) patches, with the class label appended to each patch. Our log-Mel spectrogram patches thus provide a compact format for audio representation that is digestible by CNNs

[7] sampled at 44.1 kHz
[8] using a low-pass filter as opposed to sample rate reduction
[9] the spectrograms of degraded audio were purposefully not rescaled for improved resolution
[10] up to 22.05 kHz

where each patch accounts for one full second of audio. Furthermore, by using one-second patches which are one-tenth of each provided segment, our data is augmented by turning 14,400 samples in the development set to 144,000 samples.

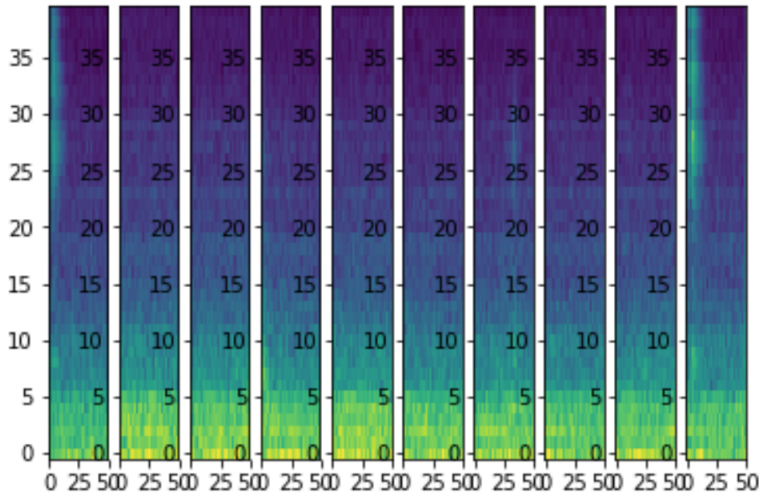


Figure 7: Example TxF spectrogram patches, that are equidistant and non-overlapping generated from the spectrogram in *Figure 6*.

The generated spectrogram patches are then organized for CNN consumption. To make the "train," "test," and "validation" subsets, the development dataset is randomly split 80/20 for training and testing respectively; then the train set is randomly split once more (80/20) for training and validation respectively. The result of this operation leaves 92,160 samples for training, 23,040 for validation, and 28,800 for testing. All data is normalized to zero-mean and unit variance[11] before classification to facilitate efficient learning cycles. This process effectively reduces the overall range of numeric values the classifier has to compute. Normalizing sample data as a whole lends itself to overfitting a classifier and will hinder its predictive capabilities. Scaling the entire dataset, before splitting it into subsets, effectively introduces information about the eventual test and validation data to the training set. By allowing a global scale to be set relative to all the data samples, the trained model will be predicting on "seen" data, and we lose the purity of the model [29]. Instead, the training set is reduced to zero mean and unit

[11] this operation occurs individually on each channel

variance in isolation, with its scale used to transform the validation and test sets. Lastly "one-hot encoding" is used to binarize the appended class labels, allowing a computer to distinguish one label uniquely from another[12].

3.4 System Architectures

To ensure that the results of the forthcoming experiment are applicable to CNN based acoustic scene classifiers of all shapes and sizes, we'll use three networks to test the preprocessed data against: *wide*, *deep*, and *generic*. Understanding the effects of sub-standard input on each of these architectural paradigms is key to understanding their effect as a whole.

Generic Network

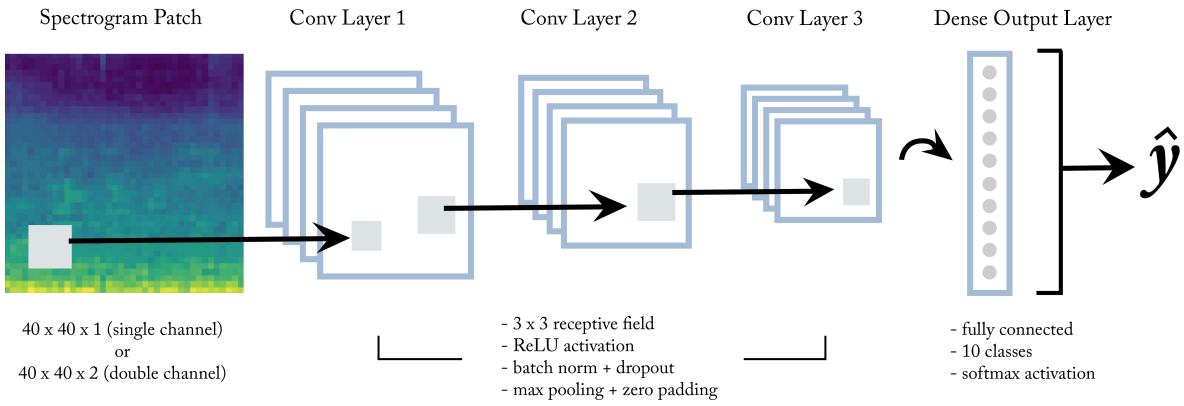


Figure 8: Generic CNN Diagram.

My proposed *generic* network features three convolutional layers and one densely connected output layer. The input layer, or Conv1, use filters with a 3x3 receptive field and unitary stride. Its outputs are channeled through a ReLU activation function, regularized with batch-normalization and dropout, then reduced through max-pooling. The second convolutional layer, or Conv2, accepts the activations from Conv1 and convolves over them with filters, again with a 3x3 receptive field and unitary stride.

[12]Categorical (i.e., non-numerical) data must first be re-contextualized for a machine by converting them into numerical values that are binary representations of the categories.

Conv3 repeats the same process as Conv1 and Conv2 but with additional filters. Ultimately the outputs from Conv3 are condensed into a fully connected output layer which uses softmax activation to make classification distinctions. The motivation for such banal design was to have a generically simple model, one that could, in theory, be trained using a machine that features the technical limitations of commodity hardware, all to show the effect of frequency and channel variance on its classification abilities.

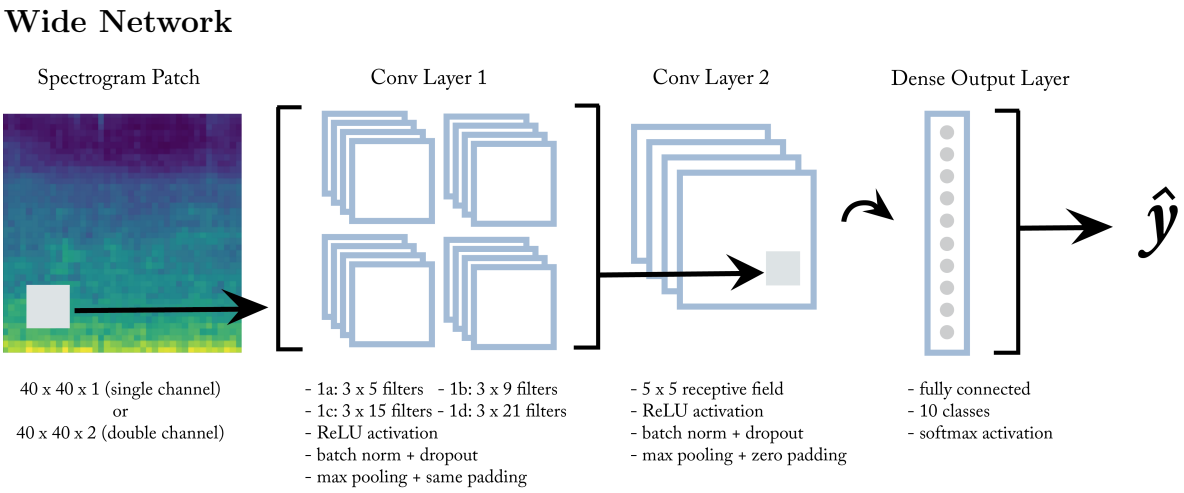


Figure 9: Wide CNN Diagram

My accompanying *wide* CNN is shallower than its predecessor with only two convolutional layers. Conv1 is an ensemble of four convolutional layers, ordered with alphabetic subscripts *Conv1a* through *Conv1d*, concatenated depth-wise along the 3rd axis. Each of the four layers in Conv1 has varying filter sizes with unitary stride, same padding, and relies on a ReLU activation function to output. They also feature non-equilateral filters (taller along the frequency axis) varying in size and quantity: Conv1a has 3x5 shaped filters, Conv1b has 3x9 shaped filters, Conv1c has 3x15 shaped filters, and Conv1d has 3x21 shaped filters. This ensembling allows the input layer to key into a variety of spectro-temporal features [30] before being fed into a dimension reducing second convolutional layer. Conv2 is a traditional convolutional layer (read: no ensembling), it uses 5x5 shaped filters with unitary stride, while also featuring successive batch normaliza-

tion, dropout, and max-pooling layers. The output from Conv2 is flattened into a dense fully connected output layer that uses softmax activation to make classification predictions.

Deep Network

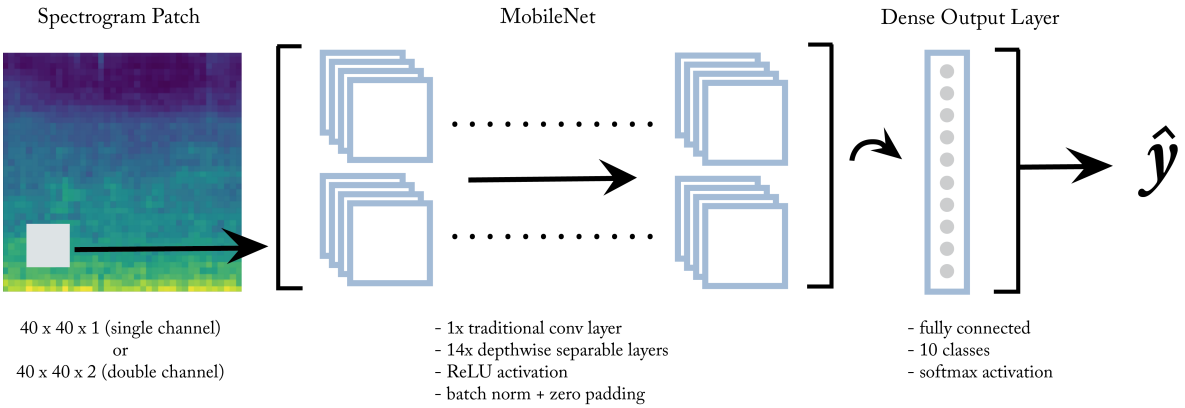


Figure 10: MobileNet Diagram

For the *deep* CNN, we'll reconfigure Google's MobileNet[13] to be our acoustic scene classifier. MobileNet is a neural network built with size and speed in mind, that is to say, its intended use is directed towards platforms with limited computing power. It differs from traditional CNNs in its use of depthwise separable layers[14] that effectively perform convolutions at a reduced computation time[15]. MobileNet relies on a traditional convolutional layer as its input layer, after which it leverages the power of depthwise separable convolutions for all subsequent layers. Its architects offer two hyper-parameters that can tune the network for virtually any number of classification projects: the width multiplier and the resolution multiplier. The width multiplier serves to thin the network

[13] As of this writing there are 3 iterations of MobileNet [31, 32, 33]. To identify which model would be optimal for our purposes, MobileNet v1 was tested against v2, with the latter performing surprisingly worse when provided the same dataset and a matching number of learnable parameters. MobileNet v3 was released during the writing of this work and was omitted from these tests. Henceforth when discussing MobileNet, we are referencing MobileNet v1.

[14] Depthwise separable convolution [34] is a process that combines depthwise convolutions (which filter the input feature map) with pointwise convolution (1x1 convolutions) to generate features.

[15] approximately 8-9x faster than traditional convolution

uniformly, reducing its overall count of learnable parameters. The resolution multiplier works to reduce the overall computation cost of the network. Tooling these parameters offer to reduce the standard number of 2.3 million parameters down to 500k or below. Our custom configuration of MobileNet is 15-layers deep[16]. The input is a traditional convolutional layer featuring zero-padding, batch normalization, and a ReLU activation function. All subsequent hidden layers are of the depthwise separable variety, also featuring zero-padding, batch normalization, and ReLU activation. Lastly, the final hidden activations undergo global average pooling and are passed through a densely connected layer with softmax activation for classification. Instead of designing a deep network from scratch, MobileNet affords us the opportunity to use a network specifically for the class of devices relevant to this work.

Shared Hyper-Parameters

For consistency, each model is trained with identical hyper-parameters where possible. Each convolutional layer features random weight initialization and is outfitted with a ReLU activation function for its output. Since neural networks are susceptible to overfitting, an essential step in designing a classifier is to account for regularization. To keep our model from grossly overfitting each convolutional activation is regularized using batch-normalization and dropout[17], to be then reduced using a subsequent max-pooling layer. Mini-batch gradient descent is used to keep training cycles expedient [37]. Each model trains on a mini-batch of 128 samples per epoch, a subset substantially smaller than the whole training data. The adaptive moment estimation (ADAM) optimization algorithm is used to facilitate efficient training. It's implemented with the learning rate ($\alpha = 0.001$) prescribed when originally published [38]. Because each acoustic scene corresponds to one class and one class only, we can use categorical cross-entropy as

[16] counting the combination of depthwise and pointwise convolutions as one distinct "depthwise separable" layer

[17] Until recently utilizing dropout in collaboration with batch normalization was an unpopular architectural choice as their disharmony had been well noted [35]. Recent work has shown that the effective placement of a dropout layer immediately following batch normalization, but before the succeeding weight layer, offered classification improvements across several deep neural networks presented with a variety of classification tasks [36].

the network's loss function. Training occurs for 50 epochs. While a form of early stopping would be nice to curtail the training time for each experiment, it makes fine-tuning a neural network a near-impossible task. By keeping the number of epochs consistent, we allow each network the same amount of training time to reach their optima during gradient descent. Their primary metric for evaluation is classification accuracy, averaged across each class. Once trained, the models can be used to make predictions on our test sets, from which we can gather empirical data regarding the scale at which varying modes of audio feature representation affects a CNN during acoustic scene classification.

Chapter 4

Experiment

4.1 Surveying the Problem Set

Each test involved asking one of the three models to classify audio using the dataset preprocessed in one of six different manners. The audio data used is offered as either downmixed to a single-channel or kept in its original two-channel format. This audio information is then provided with unfiltered frequencies or filtered down either to 1 kHz or 10 kHz. This combination of variables presents 18 different simulations to compare and evaluate. The explicit intention of the experiment is to identify the effect of reduced audio information on an acoustic scene classifier. Therefore, *classification accuracy* will be our primary performance metric for comparative evaluation. Each test was simulated in Python[1] using the Keras machine learning library within a Google Colaboratory[2] environment atop NVIDIA Tesla T4 and K80 GPUs. Since computations on a GPU are non-deterministic and often exhibit variance, each test was run five times with their results averaged to account for variability.

[1] version 3.6.8
[2] https://colab.research.google.com

4.2 Model Tuning

For their performance to be comparable, the models need to share similar amounts of learnable parameters. Identifying this magic number involved a fair amount of trial and error. Each model was tuned by training with audio downmixed to mono and its full frequency range represented. Their resulting classification accuracies offered a baseline performance of sorts to tweak hyperparameters as needed. The goal was to be mindful that no model should be over-engineered for the problem task[3]. It was also incumbent to ensure that no model was poorly tuned such that it might be prone to overfitting. The tuning process started with setting lower and higher bounds for desired learnable parameters: 400k - 500k on the low end and 2.6m - 2.7m on the high end. From there specific hyper-parameters[4] were tweaked to raise and lower the learnable parameter count until the three models observed classification accuracies that were within ±4 percentage points away from one another

Input: (40 x 40 x 1 *or* 2)
Conv1: 128x (3 x 3) + ReLU + BN *Max-Pooling*: (5 x 5) $s = 2$ *Dropout*: 25%
Conv2: 256x (3 x 3) + ReLU + BN *Max-Pooling*: (5 x 5) $s = 2$ *Dropout*: 25%
Conv3: 384x (3 x 3) + ReLU + BN *Max-Pooling*: (5 x 5) $s = 1$ *Dropout*: 25%
Dense: 10 units + softmax
$\sim$ 1.2 million parameters

Table 1: Proposed Generic CNN Architecture

After careful consideration, the number of parameters chosen for each model fell between 1.2m - 1.3m. Leaning into the higher end of learnable parameters could improve

[3]such that the experimental findings would generalize poorly to other acoustic scene classifiers
[4]number of filters, the addition/subtraction of zero-padding, max-pooling stride distance

classification accuracy, but would do so at the cost of training time[5]. While 2m+ parameters might offer an extra 1% accuracy, they push the problem set in a direction away from the limited reality of commodity hardware computing. Alternatively relying on too few parameters can trend towards severe overfitting, leaving no room to derive actionable insights from the simulations. Manual hyper-parameter selection is an innately imperfect process, especially so when generalizing for a variety of CNN architectures, in the end, our selected values fit the needs and requirements for this experiment well (*see: Tables 1, 2, & 3*).

Input: (40 x 40 x 1 *or* 2)
Conv1: 16x (3 x 3) + ReLU + BN
Conv2: 32x (3 x 3) + ReLU + BN
Conv3-4: 64x (3 x 3) + ReLU + BN
Conv5-6: 128x (3 x 3) + ReLU + BN
Conv7-12: 256x (3 x 3) + ReLU + BN
Conv13: 512x (3 x 3) + ReLU + BN
Conv14: 512x (3 x 3) + ReLU + BN
Average-Pooling: (7 x 7) $s = 1$
Dense: 10 units + softmax
$\sim$ 1.3 million parameters

Table 2: Proposed Custom MobileNet Architecture

4.3 Expectations

I believe there are auditory clues to be found in higher frequencies[6] that provide additional contextual information about an acoustic scene, that may be lost otherwise when filtered to lower ranges. It is also reasonable to suspect that there may be spatial cues specific to particular acoustic scenes. Some of these cues may provide added context that could be lost when the audio is downmixed to a single channel. I theorize that when presented with information obtained from full-range audio, an acoustic scene classifier will perform better than when presented with filtered audio. Similarly, I propose

[5] as it would require more computing power than afforded

[6] 10 kHz - 22 kHz frequency range

Input: (40 x 40 x 1 *or* 2)
Conv1: 80 (3 x 5) | 64 (3 x 9) | 32 (3 x 15) | 16 (3 x 21) + ReLU + BN *Max-Pooling*: (5 x 5) $s = 2$ *Dropout*: 20%
Conv2: 224 (3 x 3) + ReLU + BN *Max-Pooling*: (5 x 5) $s = 1$ *Dropout*: 20%
Dense: 10 units + softmax
$\sim$ 1.3 million parameters

Table 3: Proposed Wide CNN Architecture

that an acoustic scene classifier will perform better when presented with data obtained from multi-channel audio as opposed to a single channel. As it pertains to each model paradigm, my educated guess is that the more complex the model, the better it will be able to account for the substandard audio. That is to say, the expected reduction in classification performance will be mitigated by the model's depth *and* width. I expect the deep model, with its 15 layers, to hold up to substandard audio information more so than the wide and generic models (in that order).

4.4 Results

As previously defined, *classification accuracy* is the primary metric by which we'll compare classifier performance. It is, however, not the only metric worth accounting for during the experiment. To better understand the implications of degraded audio information on CNN-based acoustic scene classifiers, we can take a more granular look at model performance. By capturing each model's training progression, we can map its performance over time. These logs keep track of the model's classification accuracy and loss performance on the training and validation data subsets over each epoch. They help identify commonalities and nuances across the various models in the training process.

Another great insight regarding classification performance comes in the form of a "con-

fusion matrix"[7]. Confusion matrices are tables detailing a model's performance along class lines for each data point in the evaluation data subset. One axis represents the predicted class, juxtaposed against a second axis representing the actual class. Each classified data point is tallied up as either a "true positive," "true negative," "false positive," or "false negative," and tabulated accordingly. These values directly inform the precision[8] and recall[9] metrics of a model, which can be used to determine their harmonic average, or "F_1-score"[10].

$$\textbf{Precision} = \sum \text{True positive} \;/\; \sum \text{Predicted condition positive}$$

$$\textbf{Recall} = \sum \text{True positive} \;/\; \sum \text{Condition positive}$$

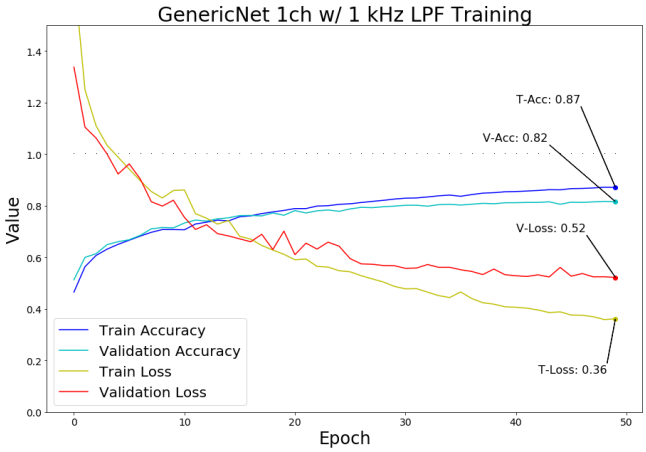
$$\textbf{F1-Score} = 2 \text{ x (Precision x Recall) / (Precision + Recall)}$$

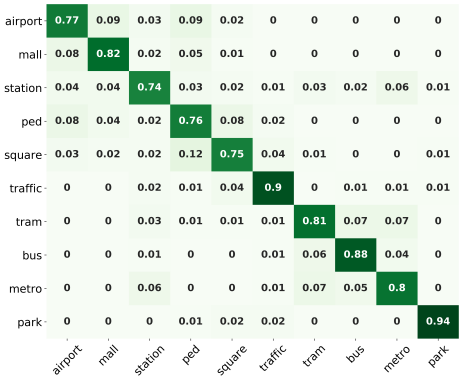
Outcomes from the experiment for the primary metric:

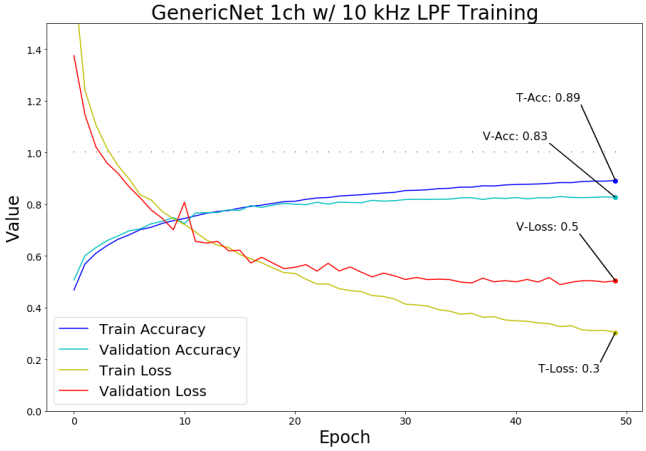
	Generic		**Wide**		**Deep**	
Hz Range	*1ch*	*2ch*	*1ch*	*2ch*	*1ch*	*2ch*
0 - 01 kHz	81.6%	91.5%	76.9%	86.3%	86.3%	91.7%
0 - 10 kHz	82.7%	92.5%	79.4%	87.5%	87.7%	92.7%
0 - 22 kHz	82.6%	92.6%	79.5%	86.2%	88.3%	93.0%

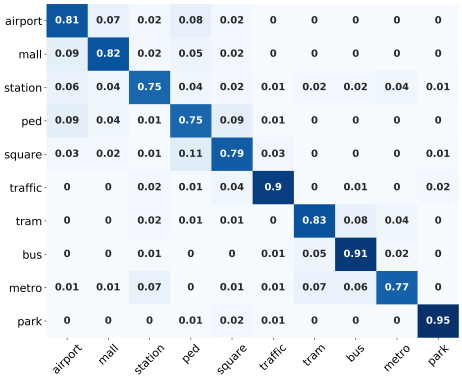
Table 4: Classification Accuracy by CNN Model

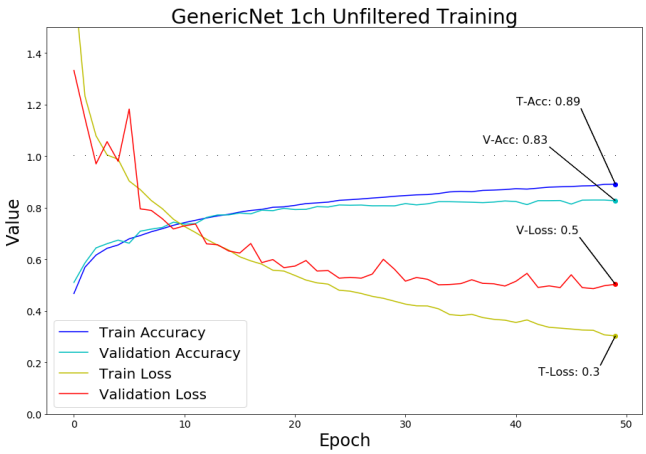
[7] also known as an error matrix
[8] also known as positive predicted value
[9] also known as sensitivity
[10] also known as F-measure


GenericNet 1ch w/ 1 kHz LPF Training
T-Acc: 0.87
V-Acc: 0.82
V-Loss: 0.52
T-Loss: 0.36
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

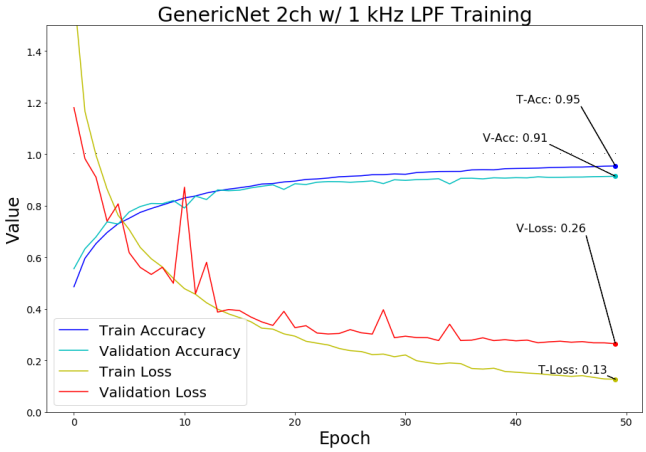

airport 0.77 0.09 0.03 0.09 0.02 0 0 0 0 0
mall 0.08 0.82 0.02 0.05 0.01 0 0 0 0 0
station 0.04 0.04 0.74 0.03 0.02 0.01 0.03 0.02 0.06 0.01
ped 0.08 0.04 0.02 0.76 0.08 0.02 0 0 0 0
square 0.03 0.02 0.02 0.12 0.75 0.04 0.01 0 0 0.01
traffic 0 0 0.02 0.01 0.04 0.9 0 0.01 0.01 0.01
tram 0 0 0.03 0.01 0.01 0.01 0.81 0.07 0.07 0
bus 0 0 0.01 0 0 0.01 0.06 0.88 0.04 0
metro 0 0 0.06 0 0 0.01 0.07 0.05 0.8 0
park 0 0 0 0.01 0.02 0.02 0 0 0 0.94
airport mall station ped square traffic tram bus metro park

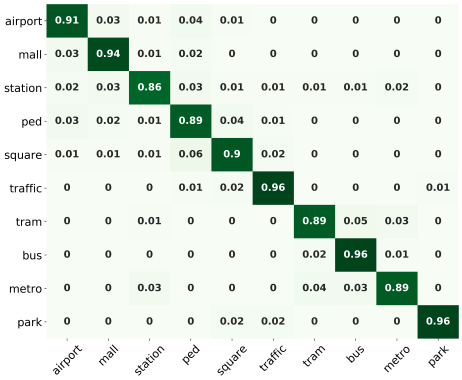

GenericNet 1ch w/ 10 kHz LPF Training
T-Acc: 0.89
V-Acc: 0.83
V-Loss: 0.5
T-Loss: 0.3
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

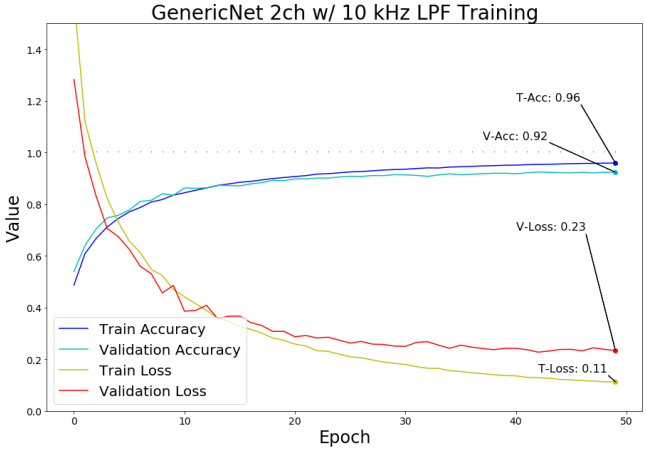

airport 0.81 0.07 0.02 0.08 0.02 0 0 0 0 0
mall 0.09 0.82 0.02 0.05 0.02 0 0 0 0 0
station 0.06 0.04 0.75 0.04 0.02 0.01 0.02 0.02 0.04 0.01
ped 0.09 0.04 0.01 0.75 0.09 0.01 0 0 0 0
square 0.03 0.02 0.01 0.11 0.79 0.03 0 0 0 0.01
traffic 0 0 0.02 0.01 0.04 0.9 0 0.01 0 0.02
tram 0 0 0.02 0.01 0.01 0 0.83 0.08 0.04 0
bus 0 0 0.01 0 0 0.01 0.05 0.91 0.02 0
metro 0.01 0.01 0.07 0 0.01 0.01 0.07 0.06 0.77 0
park 0 0 0 0.01 0.02 0.01 0 0 0 0.95
airport mall station ped square traffic tram bus metro park

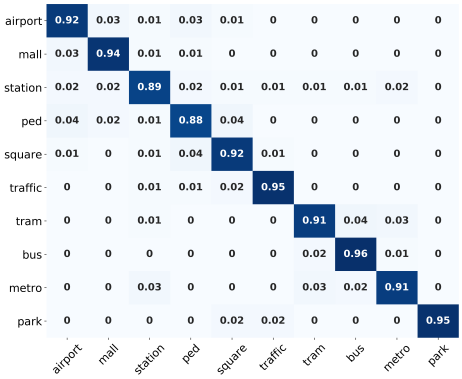

GenericNet 1ch Unfiltered Training
T-Acc: 0.89
V-Acc: 0.83
V-Loss: 0.5
T-Loss: 0.3
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

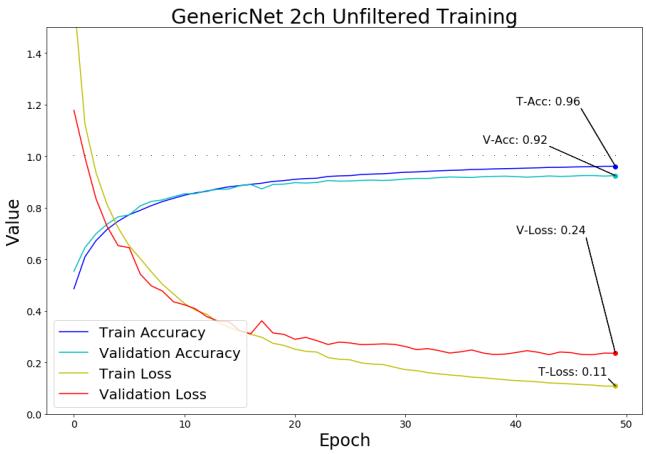

airport 0.77 0.08 0.02 0.1 0.03 0 0 0 0 0
mall 0.07 0.83 0.02 0.06 0.02 0 0 0 0 0
station 0.05 0.04 0.75 0.05 0.02 0.01 0.02 0.01 0.04 0.01
ped 0.07 0.03 0.01 0.78 0.08 0.02 0 0 0 0
square 0.03 0.02 0.01 0.1 0.78 0.04 0 0 0 0.01
traffic 0 0 0.02 0.01 0.03 0.9 0 0.01 0.01 0.02
tram 0 0 0.02 0.01 0.01 0 0.82 0.08 0.06 0
bus 0 0 0.01 0 0 0.01 0.04 0.9 0.03 0
metro 0 0.01 0.08 0 0.01 0.01 0.06 0.05 0.79 0
park 0 0 0 0.01 0.02 0.02 0 0 0 0.94
airport mall station ped square traffic tram bus metro park


GenericNet 2ch w/ 1 kHz LPF Training
T-Acc: 0.95
V-Acc: 0.91
V-Loss: 0.26
T-Loss: 0.13
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

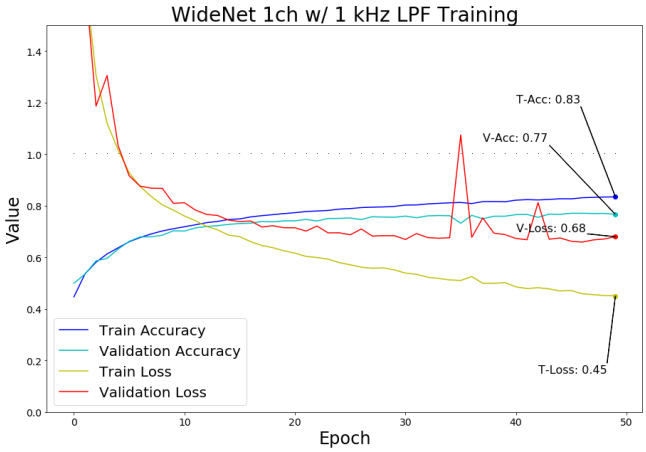

airport 0.91 0.03 0.01 0.04 0.01 0 0 0 0 0
mall 0.03 0.94 0.01 0.02 0 0 0 0 0 0
station 0.02 0.03 0.86 0.03 0.01 0.01 0.01 0.01 0.02 0
ped 0.03 0.02 0.01 0.89 0.04 0.01 0 0 0 0
square 0.01 0.01 0.01 0.06 0.9 0.02 0 0 0 0
traffic 0 0 0 0.01 0.02 0.96 0 0 0 0.01
tram 0 0 0.01 0 0 0 0.89 0.05 0.03 0
bus 0 0 0 0 0 0 0.02 0.96 0.01 0
metro 0 0 0.03 0 0 0 0.04 0.03 0.89 0
park 0 0 0 0 0.02 0.02 0 0 0 0.96
airport mall station ped square traffic tram bus metro park

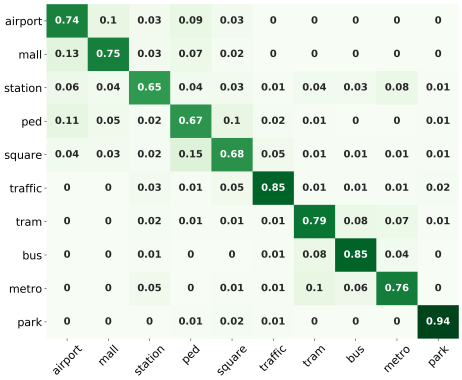

GenericNet 2ch w/ 10 kHz LPF Training
T-Acc: 0.96
V-Acc: 0.92
V-Loss: 0.23
T-Loss: 0.11
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

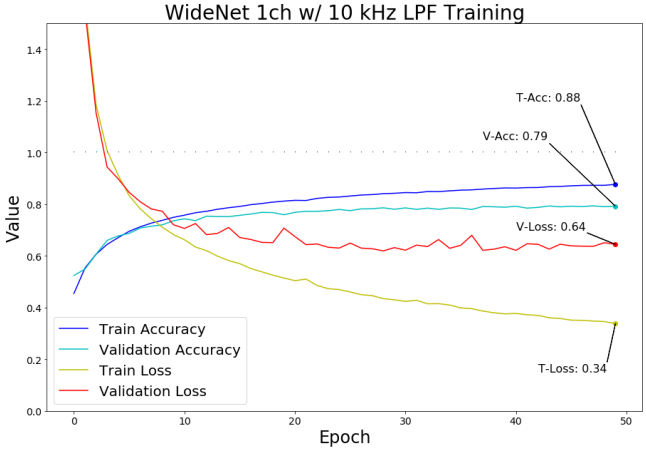

airport 0.92 0.03 0.01 0.03 0.01 0 0 0 0 0
mall 0.03 0.94 0.01 0.01 0 0 0 0 0 0
station 0.02 0.02 0.89 0.02 0.01 0.01 0.01 0.01 0.02 0
ped 0.04 0.02 0.01 0.88 0.04 0 0 0 0 0
square 0.01 0 0.01 0.04 0.92 0.01 0 0 0 0
traffic 0 0 0.01 0.01 0.02 0.95 0 0 0 0
tram 0 0 0.01 0 0 0 0.91 0.04 0.03 0
bus 0 0 0 0 0 0 0.02 0.96 0.01 0
metro 0 0 0.03 0 0 0 0.03 0.02 0.91 0
park 0 0 0 0 0.02 0.02 0 0 0 0.95
airport mall station ped square traffic tram bus metro park

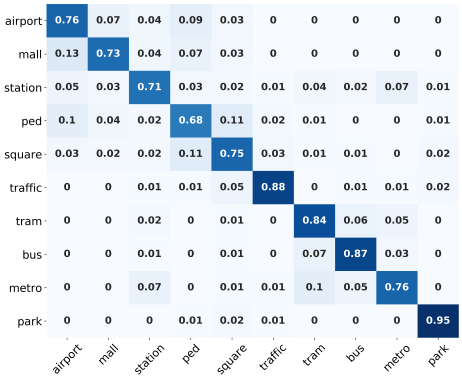

GenericNet 2ch Unfiltered Training
T-Acc: 0.96
V-Acc: 0.92
V-Loss: 0.24
T-Loss: 0.11
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

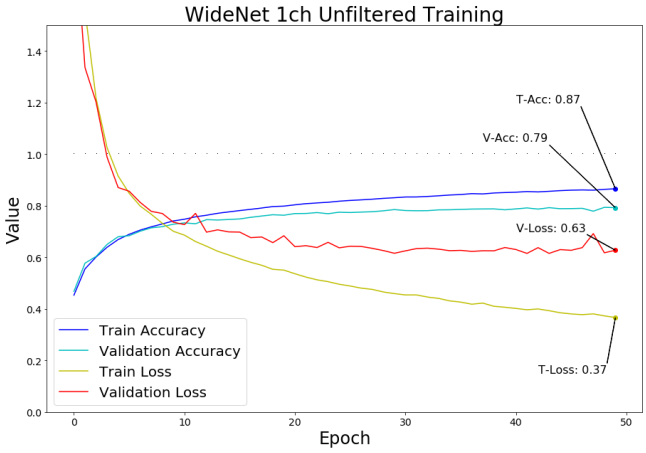

airport 0.92 0.02 0.01 0.03 0.01 0 0 0 0 0
mall 0.03 0.93 0.01 0.02 0 0 0 0 0 0
station 0.02 0.02 0.9 0.02 0.01 0.01 0.01 0 0.02 0
ped 0.03 0.02 0.01 0.88 0.04 0 0 0 0 0
square 0.01 0 0.01 0.04 0.92 0.01 0 0 0 0
traffic 0 0 0.01 0.01 0.02 0.95 0 0 0 0.01
tram 0 0 0.01 0 0 0 0.94 0.03 0.02 0
bus 0 0 0 0 0 0 0.03 0.94 0.02 0
metro 0 0 0.03 0 0 0 0.04 0.02 0.9 0
park 0 0 0 0 0.01 0.01 0 0 0 0.97
airport mall station ped square traffic tram bus metro park


WideNet 1ch w/ 1 kHz LPF Training
T-Acc: 0.83
V-Acc: 0.77
V-Loss: 0.68
T-Loss: 0.45
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

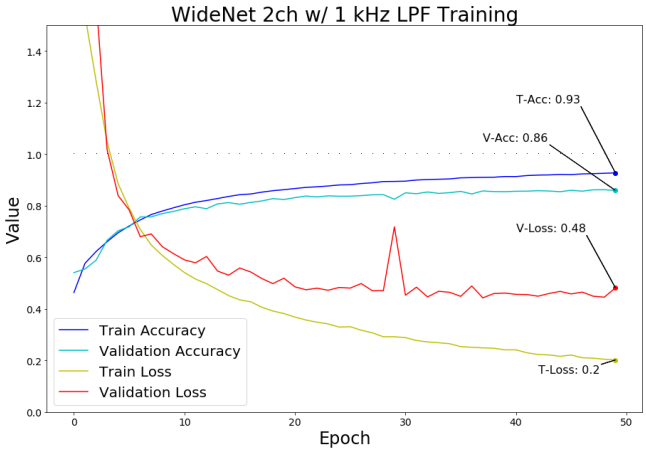

airport 0.74 0.1 0.03 0.09 0.03 0 0 0 0 0
mall 0.13 0.75 0.03 0.07 0.02 0 0 0 0 0
station 0.06 0.04 0.65 0.04 0.03 0.01 0.04 0.03 0.08 0.01
ped 0.11 0.05 0.02 0.67 0.1 0.02 0.01 0 0 0.01
square 0.04 0.03 0.02 0.15 0.68 0.05 0.01 0.01 0.01 0.01
traffic 0 0 0.03 0.01 0.05 0.85 0.01 0.01 0.01 0.02
tram 0 0 0.02 0.01 0.01 0.01 0.79 0.08 0.07 0.01
bus 0 0 0.01 0 0 0.01 0.08 0.85 0.04 0
metro 0 0 0.05 0 0.01 0.01 0.1 0.06 0.76 0
park 0 0 0 0.01 0.02 0.01 0 0 0 0.94
airport mall station ped square traffic tram bus metro park

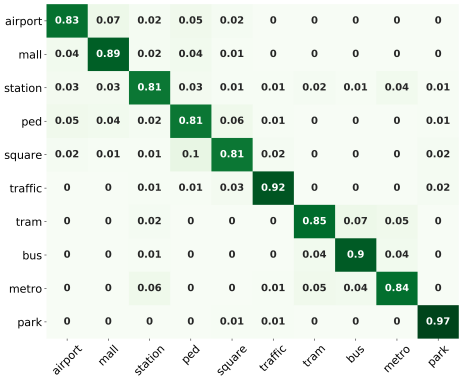

WideNet 1ch w/ 10 kHz LPF Training
T-Acc: 0.88
V-Acc: 0.79
V-Loss: 0.64
T-Loss: 0.34
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

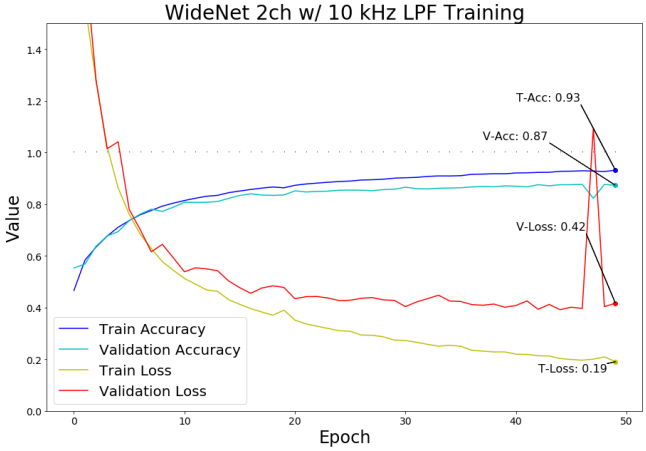

airport 0.76 0.07 0.04 0.09 0.03 0 0 0 0 0
mall 0.13 0.73 0.04 0.07 0.03 0 0 0 0 0
station 0.05 0.03 0.71 0.03 0.02 0.01 0.04 0.02 0.07 0.01
ped 0.1 0.04 0.02 0.68 0.11 0.02 0.01 0 0 0.01
square 0.03 0.02 0.02 0.11 0.75 0.03 0.01 0.01 0 0.02
traffic 0 0 0.01 0.01 0.05 0.88 0 0.01 0.01 0.02
tram 0 0 0.02 0 0.01 0 0.84 0.06 0.05 0
bus 0 0 0.01 0 0.01 0 0.07 0.87 0.03 0
metro 0 0 0.07 0 0.01 0.01 0.1 0.05 0.76 0
park 0 0 0 0.01 0.02 0.01 0 0 0 0.95
airport mall station ped square traffic tram bus metro park

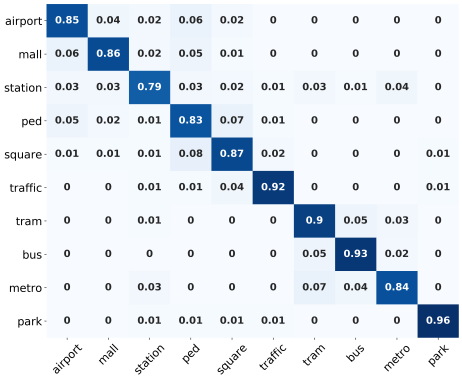

WideNet 1ch Unfiltered Training
T-Acc: 0.87
V-Acc: 0.79
V-Loss: 0.63
T-Loss: 0.37
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

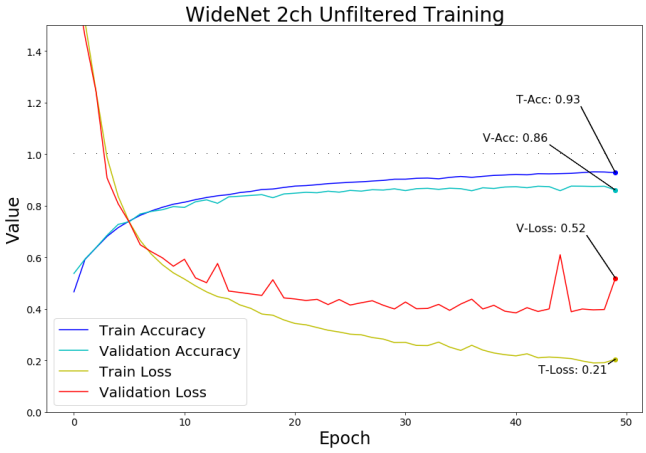

airport 0.73 0.1 0.03 0.1 0.03 0 0 0 0 0
mall 0.09 0.76 0.04 0.08 0.02 0 0 0 0 0
station 0.04 0.04 0.7 0.04 0.02 0.01 0.04 0.02 0.07 0.01
ped 0.08 0.05 0.02 0.72 0.09 0.02 0.01 0 0 0.01
square 0.03 0.02 0.02 0.14 0.73 0.03 0.01 0.01 0 0.01
traffic 0 0 0.02 0.01 0.05 0.87 0.01 0.01 0.01 0.03
tram 0 0 0.02 0 0.01 0 0.83 0.07 0.06 0
bus 0 0 0.01 0 0 0 0.07 0.87 0.04 0
metro 0 0 0.05 0 0 0.01 0.09 0.06 0.79 0
park 0 0 0 0.01 0.02 0.01 0 0 0 0.96
airport mall station ped square traffic tram bus metro park


WideNet 2ch w/ 1 kHz LPF Training
T-Acc: 0.93
V-Acc: 0.86
V-Loss: 0.48
T-Loss: 0.2
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

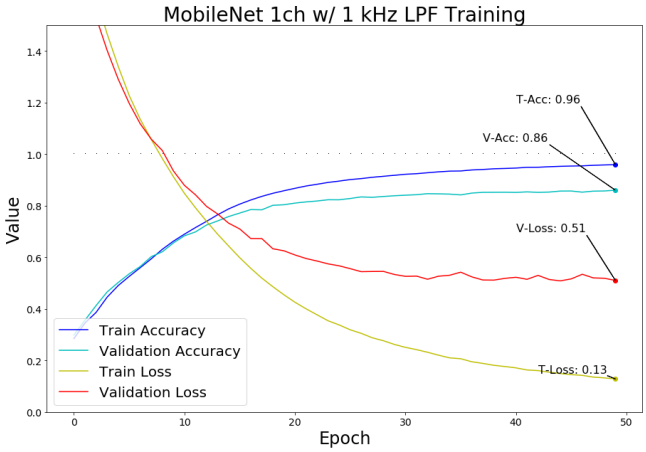

airport 0.83 0.07 0.02 0.05 0.02 0 0 0 0 0
mall 0.04 0.89 0.02 0.04 0.01 0 0 0 0 0
station 0.03 0.03 0.81 0.03 0.01 0.01 0.02 0.01 0.04 0.01
ped 0.05 0.04 0.02 0.81 0.06 0.01 0 0 0 0.01
square 0.02 0.01 0.01 0.1 0.81 0.02 0 0 0 0.02
traffic 0 0 0.01 0.01 0.03 0.92 0 0 0 0.02
tram 0 0 0.02 0 0 0 0.85 0.07 0.05 0
bus 0 0 0.01 0 0 0 0.04 0.9 0.04 0
metro 0 0 0.06 0 0 0.01 0.05 0.04 0.84 0
park 0 0 0 0 0.01 0.01 0 0 0 0.97
airport mall station ped square traffic tram bus metro park

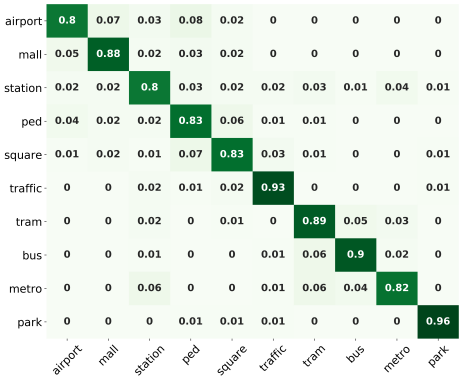

WideNet 2ch w/ 10 kHz LPF Training
T-Acc: 0.93
V-Acc: 0.87
V-Loss: 0.42
T-Loss: 0.19
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

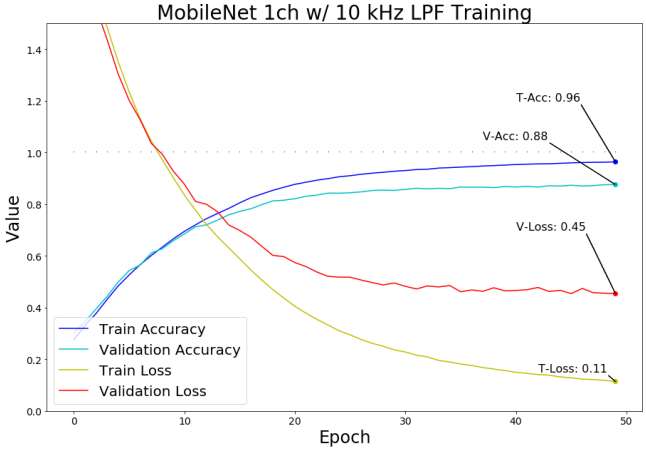

airport 0.85 0.04 0.02 0.06 0.02 0 0 0 0 0
mall 0.06 0.86 0.02 0.05 0.01 0 0 0 0 0
station 0.03 0.03 0.79 0.03 0.02 0.01 0.03 0.01 0.04 0
ped 0.05 0.02 0.01 0.83 0.07 0.01 0 0 0 0
square 0.01 0.01 0.01 0.08 0.87 0.02 0 0 0 0.01
traffic 0 0 0.01 0.01 0.04 0.92 0 0 0 0.01
tram 0 0 0.01 0 0 0 0.9 0.05 0.03 0
bus 0 0 0 0 0 0 0.05 0.93 0.02 0
metro 0 0 0.03 0 0 0 0.07 0.04 0.84 0
park 0 0 0.01 0.01 0.01 0.01 0 0 0 0.96
airport mall station ped square traffic tram bus metro park

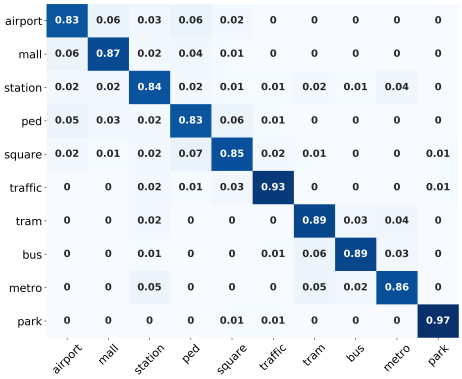

WideNet 2ch Unfiltered Training
T-Acc: 0.93
V-Acc: 0.86
V-Loss: 0.52
T-Loss: 0.21
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

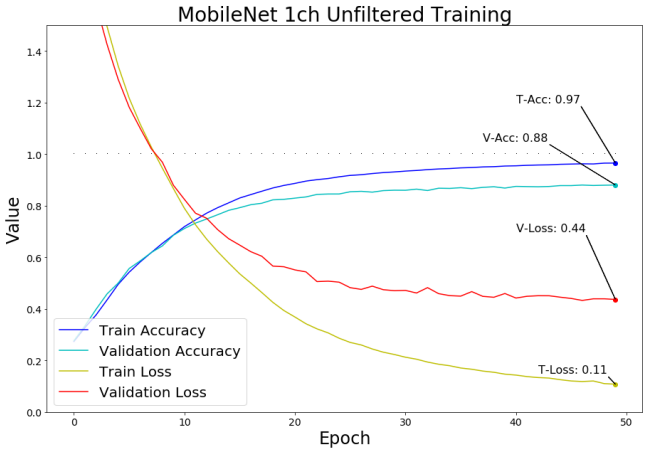

airport 0.84 0.06 0.03 0.05 0.01 0 0 0 0 0
mall 0.05 0.87 0.02 0.04 0.01 0 0 0 0 0
station 0.03 0.03 0.81 0.03 0.01 0.01 0.02 0.01 0.04 0.01
ped 0.06 0.03 0.02 0.82 0.05 0.01 0 0 0 0.01
square 0.02 0.01 0.02 0.1 0.81 0.02 0 0 0 0.01
traffic 0 0 0.02 0.01 0.04 0.89 0 0.01 0.01 0.02
tram 0 0 0.01 0 0 0 0.86 0.06 0.05 0.01
bus 0 0 0 0 0 0 0.04 0.92 0.03 0
metro 0 0 0.04 0 0 0 0.05 0.04 0.85 0
park 0 0 0 0 0.01 0.01 0.02 0 0 0.94
airport mall station ped square traffic tram bus metro park


MobileNet 1ch w/ 1 kHz LPF Training
T-Acc: 0.96
V-Acc: 0.86
V-Loss: 0.51
T-Loss: 0.13
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

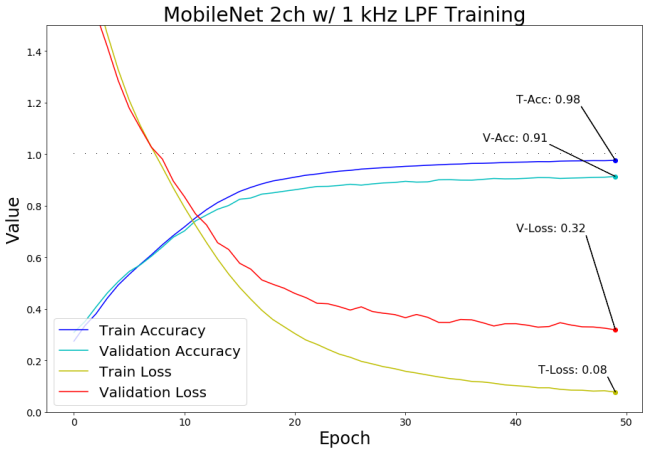

airport 0.8 0.07 0.03 0.08 0.02 0 0 0 0 0
mall 0.05 0.88 0.02 0.03 0.02 0 0 0 0 0
station 0.02 0.02 0.8 0.03 0.02 0.02 0.03 0.01 0.04 0.01
ped 0.04 0.02 0.02 0.83 0.06 0.01 0.01 0 0 0
square 0.01 0.02 0.01 0.07 0.83 0.03 0.01 0 0 0.01
traffic 0 0 0.02 0.01 0.02 0.93 0 0 0 0.01
tram 0 0 0.02 0 0.01 0 0.89 0.05 0.03 0
bus 0 0 0.01 0 0 0.01 0.06 0.9 0.02 0
metro 0 0 0.06 0 0 0.01 0.06 0.04 0.82 0
park 0 0 0 0.01 0.01 0.01 0 0 0 0.96
airport mall station ped square traffic tram bus metro park

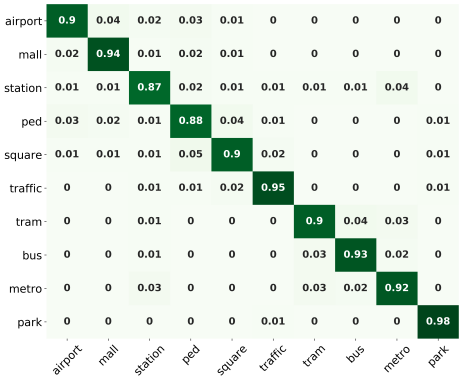

MobileNet 1ch w/ 10 kHz LPF Training
T-Acc: 0.96
V-Acc: 0.88
V-Loss: 0.45
T-Loss: 0.11
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

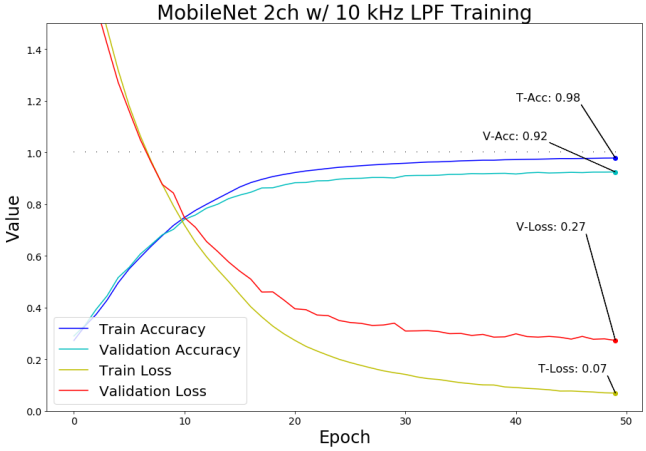

airport 0.83 0.06 0.03 0.06 0.02 0 0 0 0 0
mall 0.06 0.87 0.02 0.04 0.01 0 0 0 0 0
station 0.02 0.02 0.84 0.02 0.01 0.01 0.02 0.01 0.04 0
ped 0.05 0.03 0.02 0.83 0.06 0.01 0 0 0 0
square 0.02 0.01 0.02 0.07 0.85 0.02 0.01 0 0 0.01
traffic 0 0 0.02 0.01 0.03 0.93 0 0 0 0.01
tram 0 0 0.02 0 0 0 0.89 0.03 0.04 0
bus 0 0 0.01 0 0 0.01 0.06 0.89 0.03 0
metro 0 0 0.05 0 0 0 0.05 0.02 0.86 0
park 0 0 0 0 0.01 0.01 0 0 0 0.97
airport mall station ped square traffic tram bus metro park

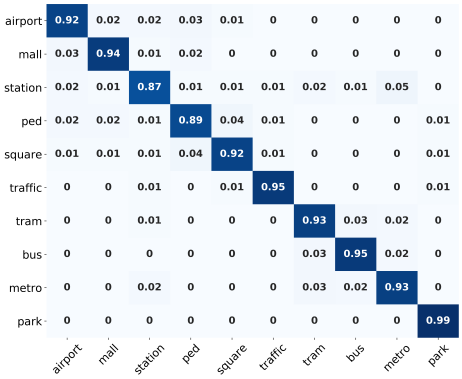

MobileNet 1ch Unfiltered Training
T-Acc: 0.97
V-Acc: 0.88
V-Loss: 0.44
T-Loss: 0.11
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch

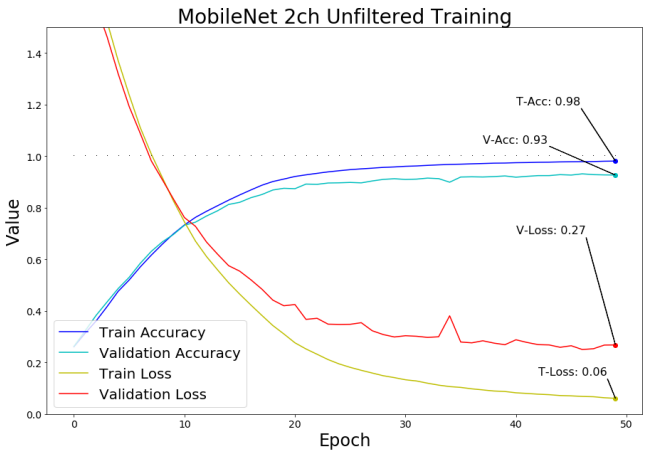

airport 0.84 0.07 0.02 0.05 0.02 0 0 0 0 0
mall 0.04 0.9 0.02 0.03 0.01 0 0 0 0 0
station 0.02 0.02 0.83 0.02 0.01 0.02 0.02 0.01 0.04 0
ped 0.05 0.03 0.02 0.82 0.06 0.01 0 0 0 0
square 0.02 0.01 0.01 0.06 0.85 0.02 0 0 0 0.01
traffic 0 0 0.01 0.01 0.02 0.94 0 0 0 0.01
tram 0 0 0.02 0 0 0 0.88 0.05 0.04 0
bus 0 0 0.01 0 0 0 0.03 0.93 0.02 0
metro 0 0 0.04 0 0 0.01 0.04 0.04 0.87 0
park 0 0 0 0 0.01 0.01 0 0 0 0.98
airport mall station ped square traffic tram bus metro park


MobileNet 2ch w/ 1 kHz LPF Training
T-Acc: 0.98
V-Acc: 0.91
V-Loss: 0.32
T-Loss: 0.08
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch


airport 0.9 0.04 0.02 0.03 0.01 0 0 0 0 0
mall 0.02 0.94 0.01 0.02 0.01 0 0 0 0 0
station 0.01 0.01 0.87 0.02 0.01 0.01 0.01 0.01 0.04 0
ped 0.03 0.02 0.01 0.88 0.04 0.01 0 0 0 0.01
square 0.01 0.01 0.01 0.05 0.9 0.02 0 0 0 0.01
traffic 0 0 0.01 0.01 0.02 0.95 0 0 0 0.01
tram 0 0 0.01 0 0 0 0.9 0.04 0.03 0
bus 0 0 0.01 0 0 0 0.03 0.93 0.02 0
metro 0 0 0.03 0 0 0 0.03 0.02 0.92 0
park 0 0 0 0 0 0.01 0 0 0 0.98
airport mall station ped square traffic tram bus metro park


MobileNet 2ch w/ 10 kHz LPF Training
T-Acc: 0.98
V-Acc: 0.92
V-Loss: 0.27
T-Loss: 0.07
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch


airport 0.92 0.02 0.02 0.03 0.01 0 0 0 0 0
mall 0.03 0.94 0.01 0.02 0 0 0 0 0 0
station 0.02 0.01 0.87 0.01 0.01 0.01 0.02 0.01 0.05 0
ped 0.02 0.02 0.01 0.89 0.04 0.01 0 0 0 0.01
square 0.01 0.01 0.01 0.04 0.92 0.01 0 0 0 0.01
traffic 0 0 0.01 0 0.01 0.95 0 0 0 0.01
tram 0 0 0.01 0 0 0 0.93 0.03 0.02 0
bus 0 0 0 0 0 0 0.03 0.95 0.02 0
metro 0 0 0.02 0 0 0 0.03 0.02 0.93 0
park 0 0 0 0 0 0 0 0 0 0.99
airport mall station ped square traffic tram bus metro park


MobileNet 2ch Unfiltered Training
T-Acc: 0.98
V-Acc: 0.93
V-Loss: 0.27
T-Loss: 0.06
Train Accuracy
Validation Accuracy
Train Loss
Validation Loss
Value
Epoch


airport 0.92 0.02 0.02 0.03 0 0 0 0 0 0
mall 0.02 0.94 0.01 0.02 0 0 0 0 0 0
station 0.01 0.01 0.9 0.01 0.01 0.01 0.01 0 0.03 0
ped 0.03 0.01 0.01 0.91 0.02 0.01 0 0 0 0
square 0.01 0 0.01 0.05 0.91 0.01 0 0 0 0
traffic 0 0 0.02 0.01 0.01 0.94 0 0 0.01 0.01
tram 0 0 0.01 0 0 0 0.93 0.03 0.02 0
bus 0 0 0 0 0 0 0.03 0.95 0.01 0
metro 0 0 0.04 0 0 0 0.03 0.02 0.9 0
park 0 0 0 0 0 0 0 0 0 0.99
airport mall station ped square traffic tram bus metro park

Chapter 5

Discussion

5.1 Effect of Channel Depth

	Generic	**Wide**	**Deep**
Hz Range	*1-2ch* $\vec{\Delta}$	*1-2ch* $\vec{\Delta}$	*1-2ch* $\vec{\Delta}$
0 - 01 kHz	-9.9%	-9.4%	-5.4%
0 - 10 kHz	-9.8%	-8.1%	-5.0%
0 - 22 kHz	-10.0%	-6.7%	-4.7%

Table 5: Difference of Accuracy Between Channel Depths

Channel depth appears to have made the most significant difference in all classification accuracy outcomes. All simulations performed markedly worse when offered single-channel audio information as opposed to the multi-channel alternative. The generic network appears to have been the greatest affected, averaging a loss of 9.9% ± .082 irrespective of the frequency band. The wide network did not fare much better, averaging a loss of 8.07% ± 1.1. Ultimately MobileNet held up best to the loss of multi-channel information with an average difference of 5.03% ± 0.286. There is also an observable

relationship with the number of layers in a network and the degree to which channel depth impacts classification accuracy. MobileNet, with its many depthwise separable layers, offered the greatest buffer to the observed poorer performance. The wide network, while shallower than the generic network, features two more convolutional layers overall due to its ensembled first layer. It appropriately withstood the information loss better than its three-layered counterpart.

The most considerable insight from these results is in determining *why* channel depth made such a difference. While preparing the dataset, we scaled everything[1] down to zero mean and unit variance[2] to help expedite the training process. On the single channel audio data, this operation occurs only once; however, on the multi-channel alternative, we perform this operation individually on each channel. The resulting spectra include two similar yet uniquely scaled portraits for every training example. It is thus reasonable to assume that in a multi-channel classification environment, the uniquely scaled spectrogram provided by the additional channel only serves to bolster the training process.

5.2 Effect of Frequency Range

	Generic		**Wide**		**Deep**	
Upper Bound	*1ch* $\vec{\Delta}$	*2ch* $\vec{\Delta}$	*1ch* $\vec{\Delta}$	*2ch* $\vec{\Delta}$	*1ch* $\vec{\Delta}$	*2ch* $\vec{\Delta}$
01 - 10 kHz	-1.1%	-1.0%	-2.5%	-1.2%	-1.4%	-1.0%
10 - 22 kHz	0.1%	-0.1%	-0.1%	1.3%	-0.6%	-0.3%

Table 6: Difference of Accuracy Across Frequency Bands

When introduced with substandard frequency information, most simulations observed a negatively impacted classification accuracy. However, the influence of frequency in-

[1] relative to the training data
[2] using SciKit Learn's Standard Scalar [39]

formation is not nearly as substantial as what we've found with channel depth. All models averaged a loss of 1.1% ± .525 in classification accuracy when dropping from the middle-tier (0 - 10 kHz) to the lower (0 - 1 kHz) frequency bands. Alternatively, there seemed to be a negligible effect on classification accuracy between the middle-tier and unfiltered bands with an average loss of .05% ± .599. However, upon further inspection, there is an outlier with the wide network performing 1.3% worse when presented with standard[3] audio information. While all other simulations either stagnated or observed marginal losses, the wide model did better. Removing this outlier brings our mean to -0.2% ± 0.236. Based on the range of loss for each model, we can see that the generic network stood firm against the reduced frequency information as it featured the least variability. The wide network, however, observed the most variability when presented with differences in frequency information.

In trying to understand why the imposed frequency constraints were virtually insignificant (debunking one of my expectations), we can look to the frequencies present in the typical stimuli of our day to day lives. When you consider that a piccolo, one of the highest-pitched instruments, caps out around 5 kHz [40], or that the range of most bird vocalization occurs between 1 kHz and 8 kHz [41], it's easier to acknowledge that the vast majority of perceptible audio information occurs in the lower half of the human audible frequency range. There may not be that much going on in those higher frequency ranges that are salient to the human listening experience. Therefore, in the context of an acoustic scene, the various excitations that contribute to its soundscape are likely to fall in this reduced frequency range. Furthermore, because our hearing experience is defined at a logarithmic scale, we can disregard a large amount of this information in both natural and machine perception.

[3]two-channels and full frequency range as originally recorded

5.3 Implications by Class

The *park* and *street traffic* acoustic scenes were featured in the top three performing categories across all simulations. The *traveling by metro*, *metro station*, and *pedestrian street* scenes were routinely in the bottom three performing classes across all simulations. Each of the three networks struggled with the latter two categories while the deep network handled the *airport* acoustic scene a bit worse than the *traveling by metro* scene.

With consideration of the impact channel depth had on classification accuracy, it merits looking into how this particular form of audio degradation affected performance across class lines.

	Generic	**Wide**	**Deep**
Scene	*1-2ch* $\vec{\Delta}$	*1-2ch* $\vec{\Delta}$	*1-2ch* $\vec{\Delta}$
airport	-13.49% ±1.27	-9.24% ±1.03	-9.02% ±0.71
shopping mall	-11.58% ±0.72	-12.52% ±1.51	-6.05% ±1.25
metro station	-13.4% ±1.5	-11.71% ±3.19	-5.13% ±2.07
pedestrian street	-12.5% ±1.22	-12.76% ±2.11	-6.68% ±1.71
public square	-13.97% ±0.35	-11.04% ±1.68	-6.53% ±0.34
street traffic	-5.56% ±0.28	-4.3% ±1.99	-1.46% ±0.79
travel by tram	-9.65% ±1.6	-4.99% ±1.33	-3.3% ±1.66
travel by bus	-5.95% ±1.81	-5.81% ±0.28	-3.86% ±1.61
travel by metro	-11.53% ±1.88	-7.73% ±0.66	-6.38% ±2.37
urban park	-1.42% ±0.92	-0.71% ±1.88	-1.65% ±0.43

Table 7: Class Specific Loss Between Channel Depths

The top-performing *urban park* scene remained mostly unaffected by the change in channel depth, observing an average difference of 1.25% in classification accuracy. Meanwhile, half of the acoustic scenes - *pedestrian street*, *airport*, *public square*, *metro station*, and *shopping mall* - averaged a double-digit increase in classification accuracy when using multi-channel audio data. The *pedestrian street*, *airport*, and *public square*, scenes were the most affected of the group with increases of 10.65%, 10.58%, and 10.51% respectively.

Scratching just under the surface of those numbers, we can see the impact of reduced channel information on classes relative to network architecture. Of the scenes with double-digit differences, *shopping mall* and *pedestrian street* saw the greatest change in the wide network, while *airport*, *public square*, and *metro station*, were most affected in the generic network.

5.4 General Observations

In terms of overall classification accuracy, the deepest network performed the best under all circumstances, followed by the generic network, with the wide network coming in last. These results indicate that network depth may have played the largest factor in classification accuracy, irrespective any modifications made to the dataset. Our simulations also showed that increased network width helped compensate for substandard audio. Therefore, raising the overall number of layers, be it in terms of depth *and* width, offers us the best opportunity to withstand degraded audio information. While MobileNet performed the best, the generic network provided comparable classification accuracy when provided the two-channel dataset. That boost from the extra channel of information served as an equalizer of sorts between the top two performing simulations. Mitigating the need for deeper networks in ASC with preprocessing techniques that preserve multi-channel information is certainly worth investigating further.

Chapter 6

Conclusions

6.1 Future Work

The scope of this project was forcibly limited to two variables of audio representation and three "styles" of CNNs. While it would have been a fool's errand to account for every network architecture or variable in these 50-odd pages, the opportunity remains to investigate further. Compared to the networks considered here, many other architectures are deserving of this type of analysis. From well-documented classifiers such as ResNet [42] or VGG [43], to the recently published SubSpectralNet [44] and plenty of its yet-to-be-named contemporaries, examining each could shed more light on the relationship of model design to handling substandard audio. Research is also needed to determine if ensembling wide layers into deep networks is a viable solution for buffering accuracy loss. Of the variables associated with audio representation and organization, there remain several elements worth exploring in a similar analysis. It remains to be seen if there is a threshold of channels after which the benefits of increased channel depth is lost, or if the findings related to channel depth hold true when compared to different ways of downmixing multiple inputs. It would also be nice to see if the accuracy loss caused by downmixing can be restored using data augmentation. Lastly, ASC

preprocessing research has shown that increased spectrogram resolution aids in classifier performance [45]. Whether the classification difference introduced by substandard conditions can be accounted for by increased frequency resolution merits a look as well.

6.2 Applicability

The findings of this work have a direct impact on future academic endeavors in ASC, as well as industrial applications of machine listening technology. In academia, it's worth investigating whether the preprocessing techniques demonstrated here transfer well to state-of-the-art solutions for other CASA tasks[1]. It was also certainly encouraging to see the classification performance of MobileNet applied to machine listening. This "plug-and-play" style of a neural network offers an existing solution that is both deployable and extensible to machine perception, particularly with commodity hardware in mind. Given the small degree by which frequency loss made a difference in classification accuracy, it is feasible for manufacturers of machine listening applications to work with substandard microphones in their projects. Furthermore, based on the significant influence of channel depth, microphone arrays that provide multiple input channels might be the best path forward for technologies that incorporate ASC.

6.3 Meaning

With each inquiry into acoustic scene classification, another set of contributions become procedural references towards the canon of optimal machine listening. This work offers a perspective of ASC that puts the focal point on preprocessing rather than the classification models themselves. While the classification model is an essential factor, obsessing over a hyper engineered architecture might lead to overlooking other equally important steps in the ASC process. By examining the representation of our data, we can unlock higher potentials for existing ASC solutions. My simulations showed that

[1] such as sound event detection, localization, and audio tagging

the reduction of frequency information on audio data does not have a substantially detrimental impact on a CNNs ability to classify acoustic scenes. They also revealed that each tested model architecture performed significantly better when offered multiple channels of uniquely scaled spectrograms for each training example. These findings open a door of possibilities to both industrial and academic pursuits as we collectively navigate the novel problem spaces of artificial perception.

List of Figures

List of Tables

Bibliography

[1] Wang, D. L. & Brown, G. J. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications* (Wiley-IEEE Press, 2006).

[2] Imoto, K. Introduction to acoustic event and scene analysis. *Acoustical Science and Technology* **39**, 182–188 (2018).

[3] Gemmeke, J. F. *et al.* Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017* (New Orleans, LA, 2017).

[4] Hayashi, T. *et al.* Bidirectional LSTM-HMM hybrid system for polyphonic sound event detection. Tech. Rep., DCASE2016 Challenge (2016).

[5] Papayiannis, C., Amoh, J., Rozgic, V., Sundaram, S. & Wang, C. Detecting media sound presence in acoustic scenes. In *Proc. Interspeech 2018*, 1363–1367 (2018). URL `http://dx.doi.org/10.21437/Interspeech.2018-2559`.

[6] Mesaros, A., Heittola, T. & Virtanen, T. Acoustic scene classification: An overview of dcase 2017 challenge entries. In *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 411–415 (2018).

[7] Deng, J. *et al.* ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).

[8] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision* **88**, 303–338 (2010).

[9] Lin, T. *et al.* Microsoft COCO: common objects in context. *CoRR* **abs/1405.0312** (2014). URL `http://arxiv.org/abs/1405.0312`. `1405.0312`.

[10] Heittola, T. & Mesaros, A. DCASE 2017 challenge setup: Tasks, datasets and baseline system. Tech. Rep., DCASE2017 Challenge (2017).

[11] Giannoulis, D. *et al.* Detection and classification of acoustic scenes and events: An ieee aasp challenge. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1–4 (2013).

[12] Barchiesi, D., Giannoulis, D., Stowell, D. & Plumbley, M. D. Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine* **32**, 16–34 (2015).

[13] MacQueen, J. *et al.* Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, 281–297 (Oakland, CA, USA, 1967).

[14] Stowell, D., Giannoulis, D., Benetos, E., Lagrange, M. & Plumbley, M. D. Detection and classification of acoustic scenes and events. *IEEE Transactions on Multimedia* **17**, 1733–1746 (2015).

[15] Mesaros, A. *et al.* Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **26**, 379–393 (2018).

[16] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986). URL `https://doi.org/10.1038/323533a0`.

[17] Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural networks* **2**, 359–366 (1989).

[18] Commons, W. File:gradient ascent (surface).png — wikimedia commons, the free media repository (2014). URL `https://commons.wikimedia.org/w/index.php?title=File:Gradient_ascent_(surface).png&oldid=143887840`. [Online; accessed 24-June-2019].

[19] Lu, Z., Pu, H., Wang, F., Hu, Z. & Wang, L. The expressive power of neural networks: A view from the width. *CoRR* **abs/1709.02540** (2017). URL `http://arxiv.org/abs/1709.02540`. `1709.02540`.

[20] Fonseca, E. *et al.* Acoustic scene classification by ensembling gradient boosting machine and convolutional neural networks. In *Workshop on Detection and Classification of Acoustic Scenes and Events* (Munich, Germany, 2017). URL `http://hdl.handle.net/10230/33454`.

[21] Porter, A., Bogdanov, D., Kaye, R., Tsukanov, R. & Serra, X. Acousticbrainz: a community platform for gathering music information obtained from audio. In *16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, 786–792 (Malaga, Spain, 2015). URL `http://dblp.org/rec/html/conf/ismir/PorterBKTS15`.

[22] Choi, I., Kwon, K., Bae, S. H. & Kim, N. S. DNN-based sound event detection with exemplar-based approach for noise reduction. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 16–19 (2016).

[23] Eghbal-zadeh, H., Lehner, B., Dorfer, M. & Widmer, G. Cp-jku submissions for dcase-2016: a hybrid approach using binaural i-vectors and deep convolutional neural networks (2016).

[24] Kim, H.-G. & Young Kim, J. Acoustic event detection in multichannel audio using gated recurrent neural networks with high-resolution spectral features. *ETRI Journal* **39**, 832–840 (2017).

[25] Cutnell, J. D. *Physics, 9th Edition* (John Wiley and Sons, 2012). URL `https://www.xarg.org/ref/a/0470879521/`.

[26] Heittola, T., Mesaros, A. & Virtanen, T. TAU Urban Acoustic Scenes 2019, Development dataset (2019). URL `https://doi.org/10.5281/zenodo.2589280`.

[27] Gelfand, S. A. *Essentials of Audiology* (Thieme, 2011). URL `https://www.xarg.org/ref/a/B005TJONK0/`.

[28] Bogdanov, D. *et al.* Essentia: An open-source library for sound and music analysis. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, 855–858 (ACM, New York, NY, USA, 2013). URL `http://doi.acm.org/10.1145/2502081.2502229`.

[29] Hastie, T. *The elements of statistical learning : data mining, inference, and prediction* (Springer, New York, 2001).

[30] Pons, J., Slizovskaia, O., Gong, R., Gómez, E. & Serra, X. Timbre analysis of music audio signals with convolutional neural networks. *CoRR* **abs/1703.06697** (2017). URL `http://arxiv.org/abs/1703.06697. 1703.06697`.

[31] Howard, A. G. *et al.* Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR* **abs/1704.04861** (2017). URL `http://arxiv.org/abs/1704.04861. 1704.04861`.

[32] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A. & Chen, L. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR* **abs/1801.04381** (2018). URL `http://arxiv.org/abs/1801.04381. 1801.04381`.

[33] Howard, A. *et al.* Searching for mobilenetv3. *CoRR* **abs/1905.02244** (2019). URL `http://arxiv.org/abs/1905.02244`. `1905.02244`.

[34] Sifre, L. & Mallat, S. Rigid-motion scattering for texture classification. *CoRR* **abs/1403.1687** (2014). URL `http://arxiv.org/abs/1403.1687`. `1403.1687`.

[35] Li, X., Chen, S., Hu, X. & Yang, J. Understanding the disharmony between dropout and batch normalization by variance shift. *CoRR* **abs/1801.05134** (2018). URL `http://arxiv.org/abs/1801.05134`. `1801.05134`.

[36] Chen, G. *et al.* Rethinking the usage of batch normalization and dropout in the training of deep neural networks. *CoRR* **abs/1905.05928** (2019). URL `http://arxiv.org/abs/1905.05928`. `1905.05928`.

[37] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. & Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR* **abs/1609.04836** (2016). URL `http://arxiv.org/abs/1609.04836`. `1609.04836`.

[38] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *CoRR* **abs/1412.6980** (2015).

[39] Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011).

[40] Goddard, S. *Attention, balance and coordination : the A.B.C. of learning success* (John Wiley & Sons, Chichester, UK Hoboken, NJ, 2017).

[41] Vundavalli, S. K. & Danthuluri, S. R. S. V. *Bird Chirps Annotation Using Time-Frequency Domain Analysis.* Master's thesis, Blekinge Institute of Technology (2016).

[42] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *CoRR* **abs/1512.03385** (2015). URL `http://arxiv.org/abs/1512.03385`. `1512.03385`.

[43] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR* **abs/1409.1556** (2014).

[44] Phaye, S. S. R., Benetos, E. & Wang, Y. Subspectralnet - using sub-spectrogram based convolutional neural networks for acoustic scene classification. *CoRR* **abs/1810.12642** (2018). URL `http://arxiv.org/abs/1810.12642`. `1810.12642`.

[45] Piczak, K. J. The details that matter : Frequency resolution of spectrograms in acoustic scene classification (2017).