Large-scale Acoustic Scene Analysis with Deep Residual Networks

by

Logan H. Ford

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author .....
Department of Electrical Engineering and Computer Science
June 7, 2019

Certified by .....
James Glass
Senior Research Scientist
Thesis Supervisor

Certified by .....
Hao Tang
Postdoctoral Associate
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Large-scale Acoustic Scene Analysis

with Deep Residual Networks

by

Logan H. Ford

Submitted to the Department of Electrical Engineering and Computer Science
on June 7, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Many of the recent advances in audio event detection, particularly on the AudioSet dataset, have focused on improving performance using the released embeddings produced by a pre-trained model. In this work, we instead study the task of training a multi-label event classifier directly from the audio recordings of AudioSet. Using the audio recordings, not only are we able to reproduce results from prior work, we have also confirmed improvements of other proposed additions, such as an attention module. Moreover, by training the embedding network jointly with the additions, we achieve a mean Average Precision (mAP) of 0.392 and an area under ROC curve (AUC) of 0.971, surpassing the state-of-the-art without transfer learning from a large dataset. We also analyze the output activations of the network and find that the models are able to localize audio events when a finer time resolution is needed. In addition, we use this model in exploring multimodal learning, transfer learning, and realtime sound event detection tasks.

Thesis Supervisor: James Glass
Title: Senior Research Scientist

Thesis Supervisor: Hao Tang
Title: Postdoctoral Associate

Acknowledgments

I would first like to thank my supervisor, Jim Glass, for his time and consistent support of my work over the past year. Working with Jim and the lab members of the Spoken Language Systems Group has been deeply enjoyable, educational, rewarding, and inspiring. I would also like to thank Hao Tang and François Grondin for their close help and assistance throughout the work that went into this thesis. They always made themselves available to collaborate, share and discuss ideas, and to help come up with workable solutions to difficult problems. Their experience and expertise gave me perspective that would've been incredibly difficult to develop on my own.

Next, a big thank you goes to my parents, Anne and Michael, for their unending love and support that was so valuable in making the most of my time at MIT. They've been extremely supportive of my decisions and efforts, and have truly been there for me through all the highs and lows.

Lastly, I would like to thank Jin Yu and the rest of the Signify team; it was a pleasure working with them all, and sharing in both their ideas and results.

This research was sponsored in part by Signify.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we will be discussing acoustic scene analysis; the use of audio data to understand the surrounding environment. Humans use audio to make deductions of their surroundings and the world. We have the ability to detect many different sounds and to associate them with objects and their characteristics. This is used, along with our past experience and our current understanding of the environment, to infer the most likely of possible scenarios. We can detect a vehicle coming down the road which is outside our visual field from the hum of the engine and the sound of the wheels meeting the pavement; predicting whether it is a car, truck, RV, or motorcycle while also estimating its speed, direction, and distance. We can infer the emotional state of a person or animal from the volume, tone, and speed of their voice. We can even gain an understanding of the type of space that we're in, like whether we're in a small conference room or large orchestra hall, from acoustic trends. The ultimate goal of acoustic scene analysis is to build systems that can understand the environment as well as humans through analyzing the surrounding audio.

All of these capabilities could be extremely useful in building systems that assist us. They could be used to keep an ear out when no one is present, like in a home or business security system, or for assisting those who are hard of hearing in being aware of and understanding important pieces of the auditory world. Such technology has already been used for wildlife conservation by detecting the calls of humpback whales

in underwater recordings[1]. These types of capabilities could also be used to build more robust autonomous agents, that can perceive the world more like we actually do—cross referencing between audio and visual input streams to not only understand the world better at face-value, but to be able to assist in finding and taking actions that will best resolve uncertainty. Hearing an odd, faint sound from down the hall may not tell us much in that moment, but may provoke us to explore what caused it and learn what is actually going on.

In introducing acoustic scene analysis and audio processing in general, we'll present some examples and their visualizations. One very common way to visually represent audio signals in the audio processing community is as a spectrogram, which is a two-dimensional plot with frequency on the y-axis, and time on the x-axis. Spectrograms are computed with a short-time Fourier transform, which computes the Fourier transform of several short and overlapping windows to track how the present frequencies change over time. For the figures shown in this chapter, we used a window size of 25ms, and shifted our window by 10ms for each frame of calculation. The y-axis of our examples here is expressed in kHz, and the x-axis is the time in seconds, with 100 frames per second. An example spectrogram is shown in Figure 1-2.

The way most people have seen sound signals visually represented is as a waveform, like the one seen in Figure 1-1. A waveform is a one-dimensional vector of amplitude values recorded several thousands of times per second. When plotted, it is visually easy to see where there is activity over the length of recording, but it is difficult to discern much more than that. In a spectrogram representation, such as that in Figure 1-2, it is possible to visually discern patterns such as formants in speech, the resonance frequencies of the vocal tract. Such patterns can be used to distinguish different types of sounds from others, and ultimately predict what may be the source of the signal. In Figures 1-2, 1-3, 1-4, 1-5, and 1-6, you may be able to distinguish some of the characteristics for each of these different sound types. Each sound is governed by the physics behind the production process. The very large variability of sound makes it impossible to, say, keep a dictionary of sounds of the physical world.

[1] `https://ai.googleblog.com/2018/10/acoustic-detection-of-humpback-whales.html`

This, along with the fact that sounds are often overlapping and present at the same time, makes classification a particularly difficult task and is why we need to utilize machine learning. In our work, we extract log Mel features, which are very common for audio tasks, as the representation of our data to be fed into models. Log Mel features are inspired by the human auditory system, and similarly have more discriminatory power for lower frequencies. We assume since these features have been successful in the field of Automatic Speech Recognition, are based on human perception, and since humans are quite good at discerning audio sounds, that these features are sufficient for acoustic scene analysis.

Within acoustic scene analysis, we are interested in two tasks, i.e., audio classification and sound event detection. Audio classification is the task of detecting whether an event occurred given an audio clip. Sound event detection is a more difficult task than audio classification, involving not only detecting whether an event occurred in a clip, but finding exactly *where* in the clip over time that the event took place. Examples of these could include recognizing car horns in driving footage or detecting glass breaking in a home security recording system. These tasks serve as the first step not only for understanding the environment but also for many downstream tasks, such as voice activity detection before speech recognition (Cho and Kim, 2011), noise detection before speech enhancement (Ravindran and Anderson, 2005), and localizing speakers before speaker identification (Liu et al., 2007). Any improvement in these tasks can potentially help improve the downstream tasks.

Human performance on audio classification has only been lightly tested, mainly in areas such as acoustic scene classification and music genre classification, with no apparent benchmark for sound event detection tasks. In (Krijnders and ten Holt, 2013) a survey was made to establish a baseline for human performance on task 1 of DCASE 2013, which was to classify a balanced set of 100 soundscapes into one of the ten categories of 'bus', 'busy-street', 'office', 'openairmarket', 'park', 'quiet-street', 'restaurant', 'supermarket', 'tube', and 'tubestation'. For the survey, 37 participants were each given 50 of the soundscapes, required to listen to them in their entirety, and sort them into the 10 categories. The mean accuracy of participants on this task was

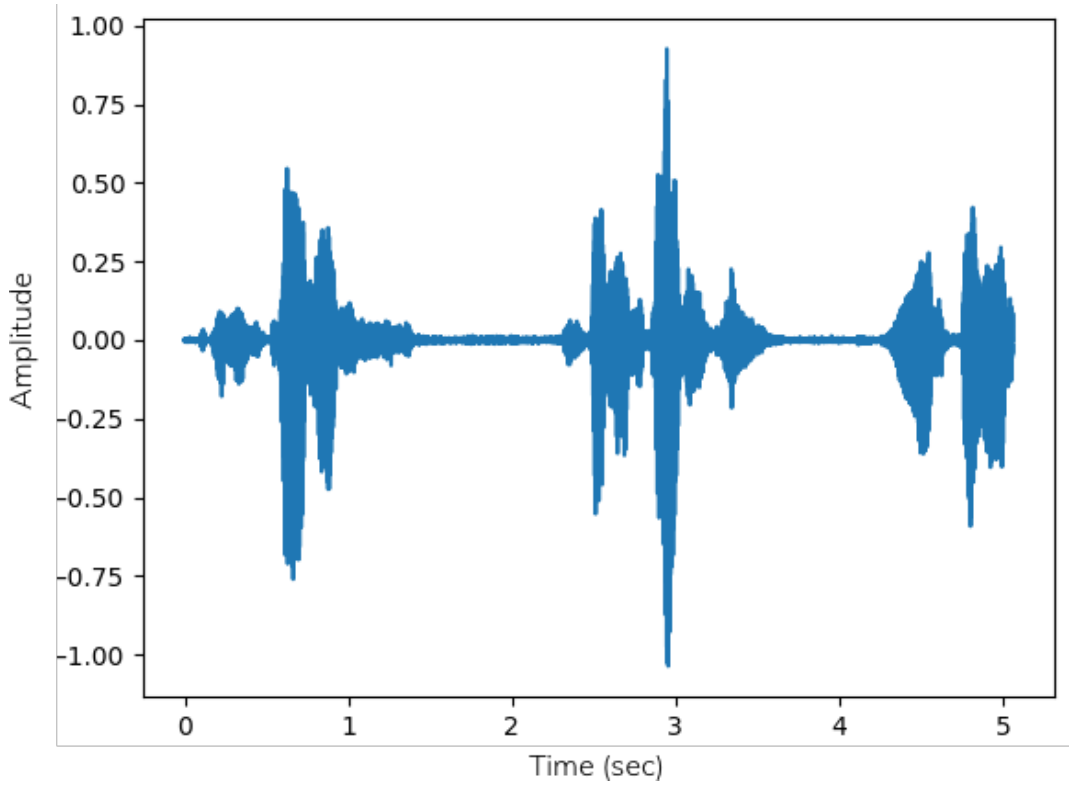


Figure 1-1: A 5-second waveform of speech sampled at 16kHz. The amplitudes are normalized to be within the range $[-1, 1]$.

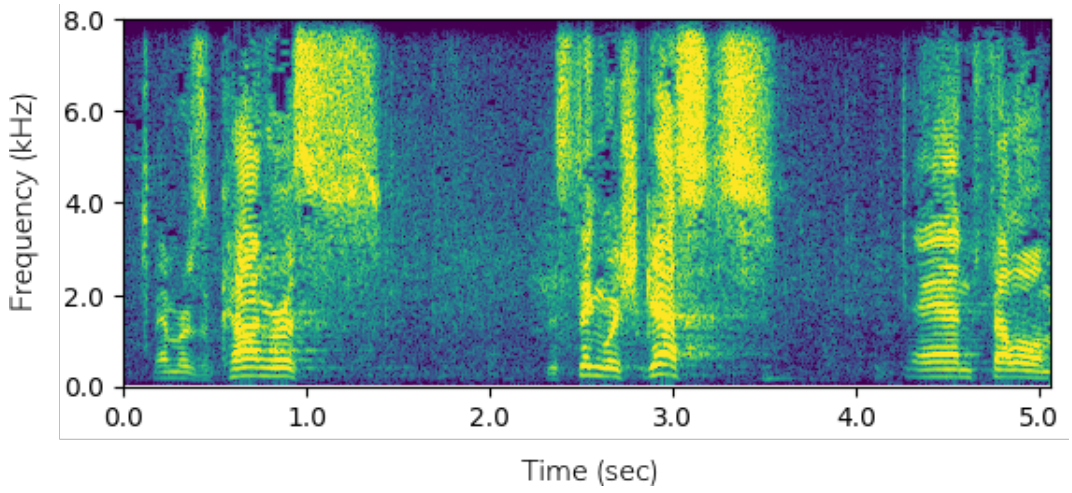


Figure 1-2: The corresponding spectrogram of the the above waveform. The analysis window is 25ms, i.e., 400 wave samples in 16kHz.

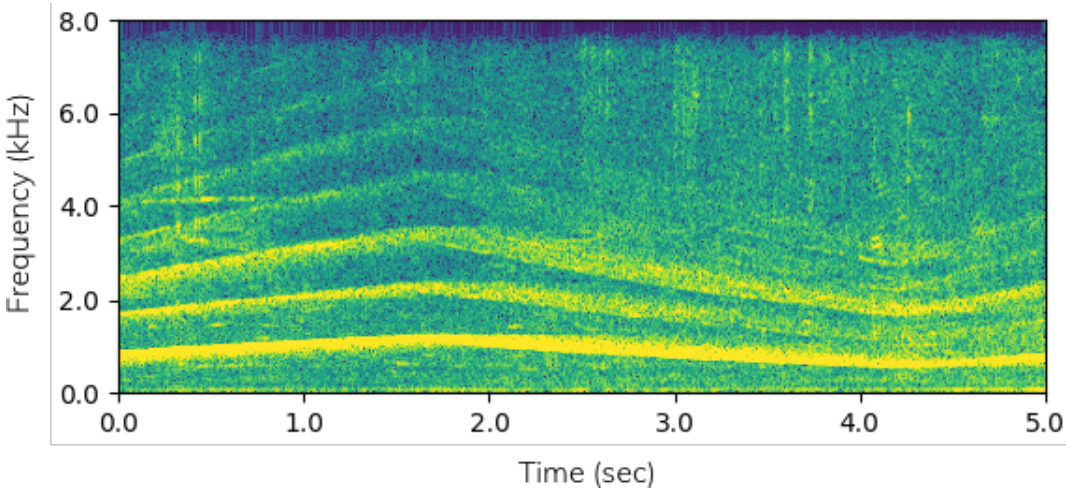


Figure 1-3: A spectrogram of a siren. Notice the strong harmonic bands moving up and down with the frequency of the siren.

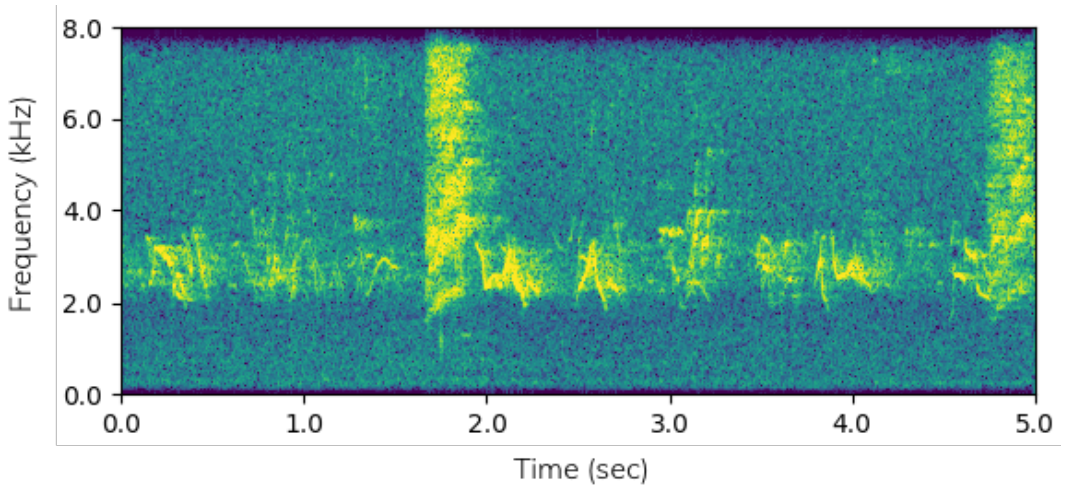


Figure 1-4: A spectrogram of birds chirping. Notice the very precise but fluttering frequency bands, with little harmonic structure present in most chirps.

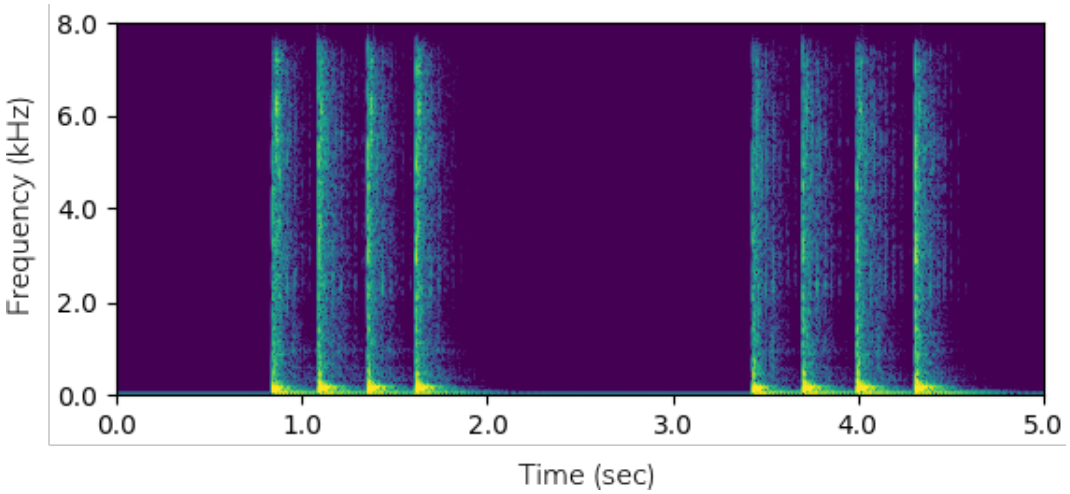


Figure 1-5: A spectrogram of knocking on wood. Notice the vertical band at impact, followed by quick fading.

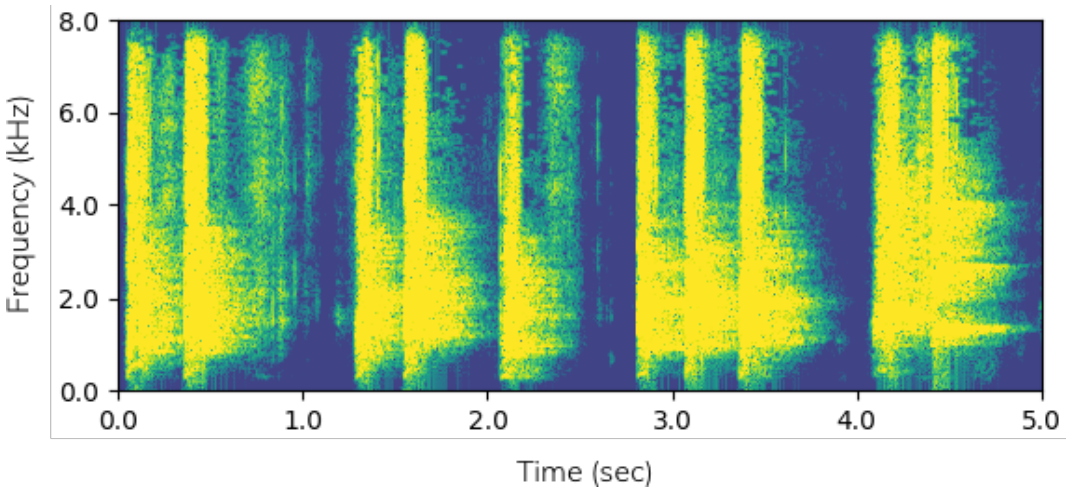


Figure 1-6: A spectrogram of a dog barking. Notice the strong vertical bands and trailing tails of specific frequency ranges.

79%, while the best performing individual model for this challenge did not surpass 77% as reported in (Stowell et al., 2015).

In (Mesaros et al., 2016a), human performance was tested on the acoustic scene classification task of DCASE 2016, which is to categorize a balanced set of 390 thirty-second segments from the TUT Acoustic Scenes 2016 dataset into the categories of 'lakeside beach', 'bus', 'cafe/restaurant', 'car', 'city center', 'forest path', 'grocery store', 'home', 'library', 'metro station', 'office', 'urban park', 'residential area', 'train', and 'tram'. For testing human performance, subsets of 30 segments with 2 segments per class were distributed among 13 participants. They found that human accuracy on this task was about 60% for those from Finland (All data was recorded in Finland), and 53% for everyone else, which is significantly lower than baseline machine performance of 77%. Expert listeners achieved performance of 77%, while the average submitted model achieves 80.9%, with the best model achieving 89.7% accuracy.

In (Seyerlehner et al., 2011), human performance was tested for the task of music genre classification on a random subset of the "1517-Artists" dataset into 19 genres. They found that human performance varied with musical knowledge, with a minimum accuracy of 26% and maximum of 71%. Nevertheless, 20 of the 24 participants beat all machine methods, which had a max of 45% accuracy. All automated methods were classical, using either Nearest Neighbor (Jean-Julien Aucouturier and Pachet, 2007; Pohle et al., 2009) or Support Vector Machine (Cortes and Vapnik, 1995; Seyerlehner et al., 2010) methods on several different types of audio features.

From these experiments, we can see that humans perform well on audio classification with broader categories, but it requires much more mental effort to perform well on fine-grained categories. In addition, finding the exact boundaries of each sound event is time-consuming, and we seek to have an automated solution for this problem.

Capturing the large variety of every sound class is impossible. Analysis can be especially challenging in cases of many overlapping sounds, or in underrepresented acoustic conditions or environments. Many very semantically different sounds, such as the class types of 'Blender' and 'Vacuum Cleaner', can actually be quite close in

acoustic characteristics, adding to the challenge. More data still needs to be collected both to quantify human performance for comparison and to better understand the main failure points of audio classification methods.

1.1 Thesis Overview

This thesis will proceed as follows: In Chapter 2, we discuss and define the tasks of audio classification and segmentation and outline previous work. In Chapter 3, we describe the model architectures used in our experiments, outline our training and evaluation procedures for classification, describe the datasets used, and share and discuss our results. This includes the DAVEnet audio-branch from (Harwath et al., 2016), a model called VGGish released alongside AudioSet from (Gemmeke et al., 2017), ResNet from (He et al., 2016), and several model extensions explored in (Kong et al., 2018; Yu et al., 2018). In Chapter 4, we describe how our model can be used for sound event detection and share our results. In Chapter 5, we explore audio-visual scene understanding, transfer learning, and a system for real-time sound event detection. And finally, we conclude in Chapter 6.

Chapter 2

Background

Our two tasks of focus are audio classification and sound event detection. In audio classification, the goal is to simply recognize when a particular type of sound occurs somewhere in the input. Sound event detection, sometimes simply referred to as audio segmentation, is a separate but similar task to audio classification. In segmentation, the goal is to be able to detect exactly where in the sample each label occurred by more finely labeling the events within the audio scene over time.

In this chapter we will look at the problems of audio classification and segmentation in detail, formally define them, review some of the earlier studies in the field, and describe some of the datasets used for enabling and evaluating progress on these tasks.

2.1 Task Definitions

In this section, we will formally define the tasks of audio classification and audio segmentation. Let $\mathcal{X}$ be the space of input frames. For example, $\mathcal{X} = \mathbb{R}^{64}$ if the input feature is 64-dimensional log Mel spectrograms. Let $\mathcal{Y}$ be the space of labels, and in this case, $\mathcal{Y} = \{0, 1\}^K$, where K is the number of audio events that we are interested in detecting. In the case of soft prediction, we can use the more general form $\mathcal{Y} = [0, 1]^K$. The task of audio classification is to find a function that maps T input frames $x_1, \ldots, x_T$ to a vector $y \in \mathcal{Y}$ where $x_t \in \mathcal{X}$ for $t = 1, \ldots, T$.

Audio classification was initially addressed as a single-label task (Nielsen et al., 2006; Sasaki et al., 2009; Grondin and Michaud, 2016; Lu et al., 2001, 2002; Dutta and Ghosal, 2017). As a single-label task, $\mathcal{Y}$ contains only vectors that have all zeros except for the value of a single index. However, by definition, audio classification is a multi-label classification task, because multiple audio events can occur in the same audio clip. This means that the value at each index of $y \in \mathcal{Y}$ can be chosen independently.

For audio segmentation, the goal is to find a function that maps T input frames $x_1, \ldots, x_T$ to a sequence $(c_1, s_1, e_1), \ldots, (c_m, s_m, e_m)$ of m events where s_i and e_i are the start and end time and $c_i \in \{1, \ldots, K\}$ is the class of the i-th event for $i = 1, \ldots, m$. Note that in general the number of frames T varies from clip to clip, and the number of events m can be zero. This means for every class, we have 0 or more start and end time pairs of when that class was present in the audio clip.

Audio classification is a weaker task in the sense that if there is a solver for audio segmentation, we can use that solver to solve audio classification. On the other hand, while collecting fine-grained labeling for audio segmentation requires expert knowledge and is labor-intensive, collecting labels for audio classification does not require expert knowledge and can be easily crowdsourced. If an audio segmentation output vector y has $y_{c_i} = 1$ for $i = 1, \ldots, m$ and zero everywhere else, then this vector y is a correct audio classification so long as the audio segmentation is correct. In other words, if a class has a start and end time pair in the audio segmentation output, then it is present in the audio clip and should be classified as "present" for an audio classification output of this audio sample.

2.2 Early Approaches

Acoustic scene analysis was a branch of research categorized as computational auditory scene analysis in cognitive science, and mostly focuses on the ability of humans to separate sounds sources (Patrick Whittlesey Ellis, 1996). A major goal of this branch is to improve speech quality for people with hearing impairment. In (Büchler

et al., 2005), they focus on classifying sound into four categories: clean speech, noisy speech, noise, and music, for hearing aids. It explores features, such as modulation frequency, variance of energies, and harmonicity, and different models, such as frame classifiers and hidden Markov models. (Larsen et al., 2003) focuses on modeling room acoustics to improve speech enhancement for hearing aids using estimated impulse responses. These studies aim to solve acoustic scene analysis in a relatively narrow domain. The study of (Liu et al., 1998) is one of the few to explore acoustic scene analysis in a more realistic setting. In (Liu et al., 1998), low-level features such as energies, pitch contours, and volume contours are used for identifying scene breaks in videos and for classifying video clips into one of five categories of TV programs.

2.3 Machine Learning Approaches

Previous works on acoustic scene analysis have used non-neural approaches, such as Support Vector Machine (SVM) and Nearest Neighbor (NN) approaches. In (Guo and Li, 2003), SVM, NN, k-NN, and nearest center (NC) approaches were compared based on their performance on an audio database of 409 sounds from MuscleFish, classified into 16 categories. Several different types of features were tested as inputs to these models, including MFCCs and perceptual features such as spectrum power, brightness, bandwidth, and pitch frequency. The SVM approach was the most successful, achieving error rates of about 10%. In (Seyerlehner et al., 2011), SVM and nearest neighbor approaches were used for a music genre classification task, and all machine methods were outperformed by most of the evaluated human labelers. Work in (Geiger et al., 2013) made use of cepstral, spectral, energies, and fundamental frequencies as features, making per frame predictions with an SVM and dynamic programming smoothing. (Jiang et al., 2005) uses an SVM approach to classify audio clips into one of the five classes, 'pure speech', 'non-pure speech', 'music', 'environment sound', and 'silence'. A Hidden Markov Model (HMM) approach for acoustic event recognition over 15 classes is shown in (Geiger et al., 2011) that can effectively learn new class types using MAP adaptation. (Temko et al., 2006) uses both SVM

and an HMM based approaches for acoustic event detection and classification on the UPC, CMU, and ITC datasets, which include a list of less than twenty office or seminar sound event categories. They tested different features such as MFCCs, frequency-filtered log filter-bank energies, zero-crossing rate, short time energy, 4 subband energies, and spectral flux. Several different approaches are outlined as submissions for the DCASE 2013 tasks in (Stowell et al., 2015), nearly all of which were HMM based.

While classical methods had some success in the task of audio classification, neural based approaches are much better suited for handling massive amounts of data, like the roughly 4,000 hours of AudioSet data, due to their much larger capacity and expressibility. Even when little in-domain data is available, transfer learning with neural architectures allows us to still take advantage of the large amounts of related data available.

2.4 Datasets

There has been a lot of progress in acoustic scene analysis thanks to many publicly available datasets, such as ESC, those in the DCASE challenges, and AudioSet. ESC-50 contains a total of 2,000 5-second recordings, with 40 examples per each of the 50 semantic classes, totaling 167 hours of recordings (Piczak, 2015b). The classes are split into the 5 categories of 'Animals', 'Natural soundscapes & water sounds', 'Human, non-speech sounds', 'interior/domestic sounds', and 'exterior/urban noises'. The TUT Acoustic Scenes 2017 dataset includes a total of 1,560 30-second recording segments, each belonging to one of fifteen acoustic scene classes, totaling 13 hours of audio data (Mesaros et al., 2016c). The acoustic scenes are indoor and outdoor locations such as 'Library', 'Office', 'Cafe/Restaurant', 'City center', and 'Metro station'. The SINS dataset, which is the focus of task 5 of DCASE 2018, used a collection of 13 microphone arrays across a home to record domestic activities, with the arrays recording for about 200 hours (Dekkers et al., 2017). These are all in contrast to AudioSet, which contains 2 million 10-second clips, labeled across over 500 class types,

totaling nearly 4,000 hours of data (Gemmeke et al., 2017). This recent push in the amount of labeled audio data has bolstered the use of deep neural approaches for acoustic scene analysis. We describe AudioSet in finer detail in Chapter 3.

Chapter 3

Audio Classification

The audio classification task is to detect whether or not a specific class occurs in an audio clip. Here we will be dealing with many classes for each audio clip, but we are not concerned with where in the clip the class occurs. This is in contrast to sound event detection, where we would be predicting exactly where in the audio each event is occurring. In this work, we focus on deep convolutional neural networks (CNNs) for audio classification due to the success in many prior studies (Serizel et al., 2018; Piczak, 2015a; Salamon and Bello, 2017; Cakır et al., 2017; Zhang et al., 2015; Bae et al., 2016). CNNs perform especially well when there is access to a large amount of data. In particular, it has been shown in (Hershey et al., 2017) that a CNN trained on 70 million audio clips on YouTube is able to generalize well to other audio clips collected in the wild, such as AudioSet (Gemmeke et al., 2017).

A set of features extracted from a model in (Hershey et al., 2017) has been released in the public domain, and a significant amount of work has been done based on these features. The study in (Kong et al., 2018) applies an attention module for each class over time after a few additional transformations on top of the released features. This approach was able to surpass the results on AudioSet in (Hershey et al., 2017). This work is extended by (Yu et al., 2018), applying a multi-level attention module. In particular, the output of attentions applied at different layers are concatenated before the final classification. This approach has the current state-of-the-art AUC result on AudioSet, significantly surpassing their previous work. Most recently, this work has

once again been extended by (Kong et al., 2019), where they applied an attention module in the hidden layers before a final classification layer, holding the current state-of-the-art for mAP.

Though the released features certainly help make progress in this field, the results are not satisfying for the following reasons. First, we ignore if pre-training on 8 million audio clips is necessary to perform well on AudioSet. Second, we are unable to state that improvement from the additions, such as the attention module, would transfer without pre-training. Finally, having to work on released features without access to the actual working model hinders the possibility of updating the base model based on the error signal from the additions, i.e., training the model end to end.

In this work, we explore these questions, by having a clean setting and clean comparisons for the task and considering some of the additions proposed in the past. Specifically, we find that training on AudioSet itself is sufficient to perform well on its evaluation set. Moreover, we do see performance improvement in the clean setting when the additions are used and when the models are trained end to end. We also explore several different architectures, and achieve a new state-of-the-art without pre-training on 8 million audio clips or any other outside data.

Here we will outline the model architectures we tested, the datasets we experimented on, the training procedures we utilized, and the metrics we used to evaluate our results.

3.1 Model Architectures

In this section, we will describe the several model architectures used in our experiments. These include our base models DAVEnet audio, VGGish, and ResNet variants, and what we call additions, which are used to extend our base models.

3.1.1 DAVEnet audio

In this subsection, we describe the DAVE network, short for Deep Audio-Visual Encoding Network. DAVEnet audio refers to the architecture used in the audio branch

of (Harwath et al., 2016), with an additional fully-connected layer and sigmoid output to enable class prediction for an audio classification task. It contains 5 convolutional layers with max pooling after all but the first. An interesting thing to note in this architecture is that the frequency dimension is flattened from the first convolutional layer, and all other convolutions are only across time. The architecture is outlined below, where "F" is the number of filters, "W" is width, "H" is height, "S" is stride, and "U" is the number of node units.

1. Convolution: F=128, W=1, H=40, ReLU
2. Convolution: F=256, W=11, H=1, ReLU
3. Maxpool: W=3, H=1, S=2
4. Convolution: F=512, W=17, H=1, ReLU
5. Maxpool: W=3, H=1, S=2
6. Convolution: F=512, W=17, H=1, ReLU
7. Maxpool: W=3, H=1, S=2
8. Convolution: F=1024, W=17, H=1, ReLU
9. Avgpool across time
10. Fully-connected: U = num_classes, Sigmoid

We later also refer to a "DAVE-ish" model, which uses a slightly tuned architecture to be able to fit neatly with additional attention based extensions, outlined in Section 3.1.4. To use the attention based extensions, we want the "DAVE-ish" model to output a vector for every 960ms of audio. To achieve this, we use a more aggressive pooling strategy with mixed max-average pooling from (Lee et al., 2016). The "DAVE-ish" architecture is outlined below.

1. Convolution: F=128, W=1, H=40, ReLU
2. Convolution: F=256, W=11, H=1, ReLU
3. Maxpool: W=3, H=1, S=2
4. Convolution: F=512, W=17, H=1, ReLU
5. Avgpool: W=2, H=1, S=2
6. Maxpool: W=2, H=1, S=2

7. Convolution: F=512, W=17, H=1, ReLU
8. Avgpool: W=2, H=1, S=2
9. Maxpool: W=2, H=1, S=2
10. Convolution: F=1024, W=17, H=1, ReLU
11. Avgpool: W=2, H=1, S=2
12. Maxpool: W=2, H=1, S=2

3.1.2 VGGish

In this subsection, we describe the VGGish architecture. The VGGish architecture and pre-trained weights were released with the AudioSet dataset to be able to regenerate the released embedding vectors[1]. While the AudioSet user community has had issues in using this model to generate similar embeddings compared to those released, the architecture is good to benchmark and compare to in our testing and iteration of models. The VGGish design is similar to that of the original VGG architecture except with fewer layers, with only 6 convolutional layers and 9 in total. The architecture is outlined below.

1. Convolution: F=64, W=3, H=3, ReLU
2. Maxpool: W=2, H=2, S=2
3. Convolution: F=128, W=3, H=3, ReLU
4. Maxpool: W=2, H=2, S=2
5. Convolution: F=256, W=3, H=3, ReLU
6. Convolution: F=256, W=3, H=3, ReLU
7. Maxpool: W=2, H=2, S=2
8. Convolution: F=512, W=3, H=3, ReLU
9. Convolution: F=512, W=3, H=3, ReLU
10. Maxpool: W=2, H=2, S=2
11. Fully-connected: U = 4096, ReLU
12. Fully-connected: U = 4096, ReLU

[1] `https://github.com/tensorflow/models/blob/7d2da5cbc9d364ca597575f61c8edff781058ac4/research/audioset/vggish_slim.py`

13. Fully-connected: U = 128, ReLU

3.1.3 ResNet Variants

In this subsection, we describe the proposed and evaluated ResNet variants. ResNet (He et al., 2016), short for residual network, was the first model to introduce skip connections (sometimes also known as residual connections), additional layers that pass values around future layers, allowing blocks of layers to easily learn the identity function and avoid transformation of features. Skip connections also enable improved error propagation and significantly help against the vanishing gradient problem. This allows for the extending of a network with many more layers with little worry of degradation in performance. The ResNet architecture has been used in audio tasks such as speaker spoof detection (Chen et al., 2017b) and unsupervised audio representation learning (Jansen et al., 2018).

We start with the ResNet50 audio variant proposed by (Hershey et al., 2017), termed here **ResNet A**. It consists of 50 layers with a skip connection almost every 3 layers, with an average pooling layer before prediction. Following (Hershey et al., 2017), we divide each 10s audio clip into 960ms independent segments, feed each segment into the network to make independent predictions, then average the individual predictions over all segments.

The first variation we explored eliminated the segmentation step used by ResNet A, and processed the entire 10s audio clip, advancing one 10ms frame at a time. As shown in Figure 3-1, this effectively produced a 3D tensor consisting of time (T), Channels (filterbanks), and CNN filters (F). The **ResNet B** model performed Global pooling over this tensor at the penultimate layer, producing a single F-dimensional vector, which was passed to the final classification layer. Global pooling thus averages over time and frequency channels.

In order to be able to attend over time, we then considered a different kind of pooling that we call Channel pooling, which average pooled only over the Channel

Model	Input	Base Model	Pooling	Extension
ResNet A	11x960ms	ResNet50	Global	1FC + Time Avg
ResNet B	10s	ResNet50	Global	1FC
ResNet B + att	10s	ResNet50	Channel	1FC + Attention
ResNet C + att	10s	ResNet34	Channel	1FC + Attention
ResNet D + att	10s	ResNet101	Channel	1FC + Attention
ResNet B + emb + att	10s	ResNet50	Channel	3FC + Multi-Attention
ResNet B + emb + 2att	10s	ResNet50	Channel	3FC + 2 Multi-Attention

Table 3.1: Tested ResNet Architectures. Note that the Base Models are the ResNet architecture after applying changes described by (Hershey et al., 2017), and removing all layers after the final convolution layer.

dimension. Where global pooling is defined as

$$y = \frac{1}{CT} \sum_{i \leq C} \sum_{j \leq T} \mathbf{X}_{i,j}$$

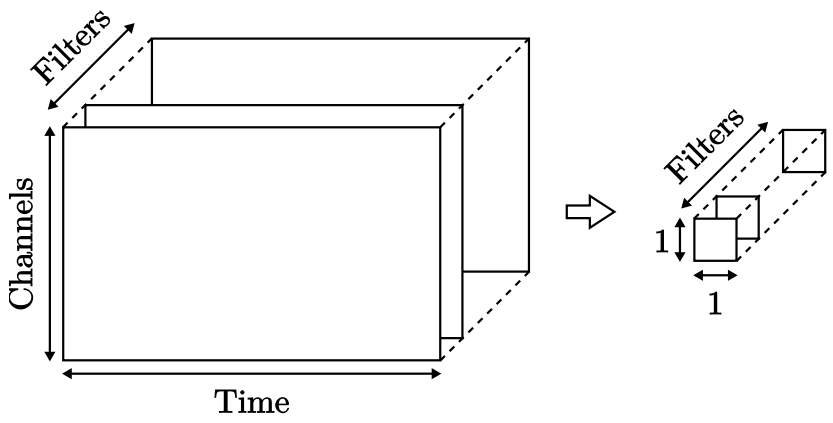
for 2D input matrix X, channel pooling is defined as

$$\mathbf{y} = \frac{1}{C} \sum_{i \leq C} \mathbf{X}_{i}$$

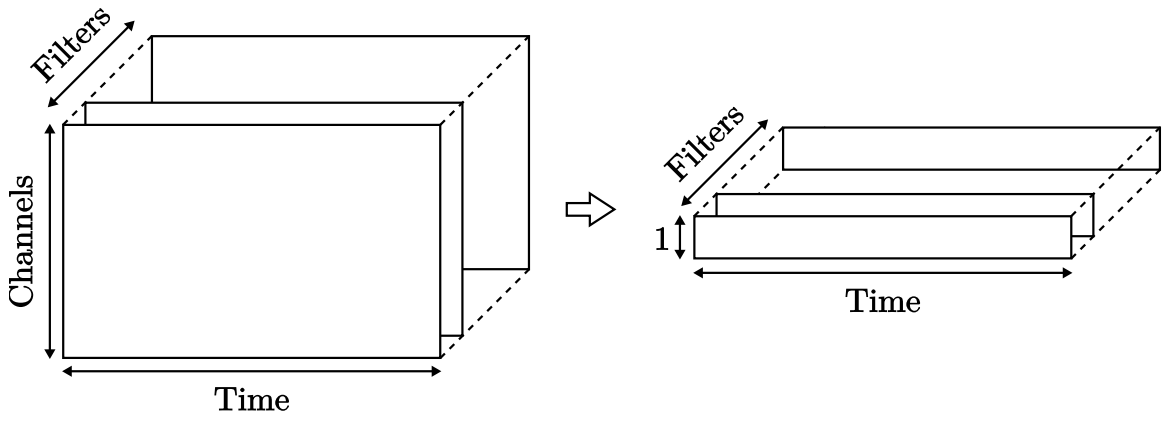
where our returned $\mathbf{y}$ is a vector, maintaining its length in the time dimension.

Model **ResNet B + att** incorporated an attention mechanism with Channel pooling. Work in (Yu et al., 2018) utilizes an "embedded mapping" and multi-level attention, which are described in more detail in the next section. Model **ResNet B + emb + att** and Model **ResNet B + emb + 2att** utilize both the embedded mapping ("emb") and multi-level attention. **ResNet B + emb + 2att** has a separate attention module for each level of features, while **ResNet B + emb + att** shares the same attention module for both levels as seen in the original paper.

Finally, we also considered two other ResNet models using different residual architectures. **ResNet C + att** and **ResNet D + att** are the same as **ResNet B + att** but are based on ResNet34 and ResNet101, respectively. Table 3.1 describes these model architectures in detail.



(a) Global pooling



(b) Channel pooling

Figure 3-1: Global vs Channel pooling. In global pooling, all elements in the time and channel dimensions are averaged to generate a $1 \times 1 \times F$ tensor, with F being the number of filters. Alternatively, channel pooling averages over channels to generate a $1 \times T \times F$ tensor, where T is the number of frames, which allows for using an attention mechanism over time.

3.1.4 Additions

There are several additions that have been used to improve classification performance on the released features of AudioSet. Two of these, used in (Kong et al., 2018), are an "embedded mapping" and an attention module. The embedded mapping consists of three fully-connected layers with 1,024 units each, taking as input a feature vector for one time step and outputting a transformed vector. These layers are applied as a transformation for each time step's feature vector. The attention module learns to predict an output per class for each time step, and a class-specific weighting over time steps used for each class's final output. In the follow-on work in (Yu et al., 2018), a multi-level attention module is used to achieve higher performance. The multi-level attention module uses the same attention module as described previously, but it is applied to the outputs of intermediate layers of the embedded mapping in addition to the final transformed vector. These outputs are concatenated and then fed into a fully-connected layer for a final output.

3.2 Dataset

Our work focused on utilizing AudioSet, a collection of over 2 million 10-second clips of YouTube videos released by Google, weakly labeled with the sounds that the clip contains from a set of 527 labels. These 527 are a part of an ontology of sounds[2]. Weakly labeled, as opposed to strongly labeled, means that labels are given to a clip with no indication of where in the clip the associated sound occurred. AudioSet is also a multi-label dataset so every clip can, and most often does, have multiple labels associated with it, with an average of 5 labels per sample. The dataset is split into three groups: `balanced_train`, `unbalanced_train`, and `evaluation`. The `balanced_train` dataset is a set of 22,000 examples, where each label has 49 samples, while the `unbalanced_train` set contains the rest of the complete training dataset. The `evaluation` set consists of 22,000 examples. AudioSet indexes video

[2] `https://github.com/audioset/ontology/blob/1e85b88a094054c66096a18073b76cb71f5b1436/ontology.json`

IDs, timestamps, and labels for each video segment from YouTube. It also provides bottleneck features, which consist of 128-D vectors for each second of audio, which were obtained using VGGish trained on an early version of the YouTube-8M dataset.

The labeling procedure for AudioSet outlined in (Gemmeke et al., 2017) involved majority voting of 3 labelers per sample. Labelers were presented with both video and audio of a 10 second sample and did not have access to either the title or any metadata. It was reported that 76.2% of sample voting was unanimous, 23.6% were 2:1 majority, and only 0.5% were "unsure". Note that interpretation of these numbers is nuanced, as they state that if the first two labelers were in agreement, they do not have another labeler vote and thus these cases fall under the unanimous set, even if in reality a third voter would have put them at a 2:1 majority vote.

We extracted the dataset from YouTube, but due the constant change in video availability (videos being removed, taken down, etc.) there is a natural shrinkage (about 5%) from the original dataset. This noted, we do draw fair comparisons between the previous state-of-the-art architecture and our models by evaluating on the same subset of the evaluation dataset. The audio is stored at a 16kHz sampling rate in the FLAC format, the Free Lossless Audio Codec. We strictly use log Mel filterbank outputs as the features for our models, with the following parameters: a window size of 25ms, window stride of 10ms, hamming window, and 64 log Mel bins.

3.3 Evaluation

We train our model to be able to predict the classes that occur at some-point in an approximately 10 second long audio clip. We have our model predict each label independently, as multiple labels can occur in a single sample. Our model predicts softly, returning some value between 0 and 1. It is also worth noting that while the labels of AudioSet form an ontology, each label is still predicted independently, partly due to the fact that the appearance of a ground truth label does not imply the presence of parent and ancestor labels.

For evaluation, we measure how our model performs on the AudioSet evaluation

set by three metrics:

1) mean Average Precision (mAP), which is the mean across all class's Average Precision, which is an approximation of the area under a class's precision-recall curve. This is defined as

$$\sum_n (R_n - R_{n-1}) P_n$$

where R_n and P_n are the Recall and Precision for the nth threshold. We calculate the AP of each class using the Python package sklearn implementation[3].

2) Area under the curve (AUC) of the receiver operating characteristic (ROC) curve of all classes. A receiver operating characteristic is the curve of True Positive Rate versus False Positive Rate as the discrimination threshold for binary classification is changed. This acts as a step function, and therefore the area can be calculated exactly. We again use an sklearn implementation for this metric[4].

3) Sensitivity Index (d-prime), is deterministically calculated from AUC as

$$d' = \sqrt{2} Z(AUC)$$

where $Z(p)$ is the inverse of the Cumulative Distribution Function, otherwise known as the Percent Point Function. The main reasoning for its use is that AUC values are quite high on AudioSet performance, and d-prime more clearly distinguishes differences even for small changes in AUC.

We use these metrics rather than say F1 score because our models predict softly, giving some value between zero and one. This allows us to characterize performance across all threshold choices. These metrics will reward a model that is closer to the correct answer even if only slightly and while still in absolute terms, quite far off. It's for this same reason that we use a binary cross entropy loss function, as a zero-one loss would increase sparsity in signal.

Our evaluation subset consists of 19,185 samples; 5.9% fewer than the original 20,371. However, all the numbers reported are computed against the same evaluation

[3] `https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html`

Model	mAP	AUC	d-prime
DAVE audio-branch	0.144	0.911	1.905
DAVE-ish + emb + att	0.156	0.920	1.990
DAVE-ish + feature_level_model	0.157	0.924	2.022
VGGish	0.130	0.913	1.918
VGGish + emb + att	0.159	0.927	2.051
VGGish + feature_level_model	0.153	0.926	2.043
ResNet A	0.184	0.927	2.056
ResNet B	0.222	0.943	2.237

Table 3.2: Model Performance of AudioSet-100k trained models on AudioSet Evaluation set.

set. We also found that the results on our evaluation set are consistent with the published numbers.

3.4 Experiments

3.4.1 Preliminary Results on AudioSet-100k

Each model was trained on a subset of AudioSet we call AudioSet-100k, a subset of 100,000 training samples, comprised of the entire available `balanced_train` set and 80k samples randomly taken from the `unbalanced_train` set. We trained each model for up to 100 epochs, with most all models peaking in performance by the 50th epoch. All networks were trained with the Pytorch framework (Ketkar, 2017), using the Adam optimizer (Kingma and Ba, 2015), with a learning rate of 0.001, learning rate reduction by an order of magnitude every 40 epochs, weight decay of 5×10^{-7}, and beta values of 0.95 and 0.999. We utilize the Binary Cross Entropy loss function.

In our evaluation of the performance of models on the AudioSet-100k data, we see in Table 3.2 that the ResNet architectures outperformed both DAVEnet audio and VGGish variants. Both DAVE audio and VGGish architectures saw significant improvements in performance with the addition of embedded mapping and multi-level attention layers. ResNet B shows much improved performance of ResNet A, likely

Model	mAP	AUC	d-prime
Hershey et al. (2017)	0.314	0.959	2.452
Yu et al. (2018)	0.360	0.970	2.660
Kong et al. (2019)	0.369	0.969	2.640
VGGish	0.274	0.952	2.349
ResNet A	0.347	0.966	2.582
ResNet B	0.329	0.966	2.584
ResNet A + att	0.352	0.966	2.581
ResNet B + att	0.379	0.970	2.657
ResNet B + att (Avg)	**0.392**	**0.971**	**2.682**
ResNet C + att	0.360	0.966	2.587
ResNet D + att	0.380	0.970	2.655
VGGish + emb + att	0.286	0.946	2.270
ResNet B + emb + att	0.345	0.958	2.446
ResNet B + emb + 2att	0.334	0.955	2.396

Table 3.3: Model Performance of fully trained models on AudioSet Evaluation set. Note that (Avg) indicates checkpoint averaged results, where outputs of all 50 epoch versions are averaged and evaluated.

due in part to the large residual field of this architecture.

3.4.2 Results on Full AudioSet

Each model was trained on the whole subset of the AudioSet training set that was still available on YouTube at the time of extraction. Our training subset consists of 1,953,082 samples of the total 2,063,949, a 5.3% loss from the original dataset. We trained each model for up to 50 epochs, with most all models peaking in performance by the 40th epoch. All networks were trained with the Pytorch framework (Ketkar, 2017), using the Adam optimizer (Kingma and Ba, 2015), with a learning rate of 0.0001, learning rate reduction by an order of magnitude every 40 epochs, weight decay of 5×10^{-7}, and beta values of 0.95 and 0.999. We utilize the Binary Cross Entropy loss function.

To ensure we could compare to the state-of-the-art method, we re-trained the

Multi-level attention model in (Yu et al., 2018) on the released features and evaluated the final model-averaged results on our evaluation subset. As the results were slightly worse overall (mAP: 0.3586, AUC: 0.9678), we believe that the published numbers are a fair comparison.

As seen in Table 3.3, there's an interesting pattern in our transition from ResNet A, to ResNet B, to ResNet A + att, to ResNet B + att. Performance worsened when feeding the whole audio clip into the ResNet variant model (B). The addition of an attention module improved our performance for both our ResNet A and ResNet B models. Performance improved significantly for the ResNet B model in particular, leading to our checkpoint averaged result achieving state-of-the-art performance (ResNet B + att (Avg)), and even our peak individual model (ResNet B + att) still outperforms previous state-of-the-art results. It is also clear in the progression from our shallower to deeper models (C, B, D) that there is clear performance gain in moving from the ResNet34 to the ResNet50 based model, but insignificant gains moving deeper from ResNet50 to ResNet101 based models.

The VGGish model does not perform particularly well, and does not surpass even the task's baseline performance.

It is worth pointing out that, in contrast to the published results, the added "embedded mapping" does not improve peak performance here, as seen by the results of Models ResNet B + emb + att and ResNet B + emb + 2att. It's also interesting to see that these models implementing multi-level attention have a better increase in performance when the attention module is shared between levels (ResNet B + emb + att), as opposed to a separate module per level (ResNet B + emb + 2att).

We see a significant gain in the performance of our ResNet B + att with model averaging. Model averaging refers to the use of several model checkpoints in an ensemble classifier, averaging each model's output probability vectors for a final output vector, as outlined by (Chen et al., 2017a). The general idea is that each model will have some bias and variance in its outputs, and by averaging our output vectors we are "averaging out" the noise and therefore likely to converge to a more correct output. This is in contrast to averaging the model parameters, which they call Checkpoint

Class	AP	AUC	# training samples	Quality estimate
Bagpipes	0.9169	0.9987	1715	90
Emergency Vehicle	0.9117	0.9951	5586	100
Crowing, cock-a-doodle-doo	0.8933	0.9990	2033	100
Hoot	0.8792	0.9831	109	56
Civil Defense Siren	0.8703	0.9972	1903	100
Change ringing (campanology)	0.8689	0.9995	605	90
Siren	0.8635	0.9941	8286	100
Battle Cry	0.8575	0.9992	387	89
Didgeridoo	0.8395	0.9983	1180	90
Accordion	0.8382	0.9990	2833	90

Table 3.4: Top 10 classes in AP for the checkpoint averaged model, ResNet B + att (Avg), in results on the AudioSet evaluation set. # training samples refers to the number of samples labeled with this class in the AudioSet training set, and quality estimate refers to a quality estimate of the class labeling in AudioSet, given with the release of the dataset.

Class	AP	AUC	# training samples	Quality estimate
Bagpipes	0.9185	0.9985	1715	90
Emergency Vehicle	0.9006	0.9968	5586	100
Change ringing (campanology)	0.8809	0.9995	605	90
Civil Defense Siren	0.8723	0.9979	1903	100
Crowing, cock-a-doodle-doo	0.8628	0.9984	2033	100
Battle cry	0.8568	0.9990	387	89
Siren	0.8504	0.9940	8286	100
Hoot	0.8385	0.9824	109	56
Accordion	0.8248	0.9979	2833	90
Music	0.8202	0.9397	1005610	100

Table 3.5: Top 10 classes in AP for the best individual model, ResNet B + att, in results on the AudioSet evaluation set. # training samples refers to the number of samples labeled with this class in the AudioSet training set, and quality estimate refers to a quality estimate of the class labeling in AudioSet, given with the release of the dataset.

Smoothing, but this they claim could be problematic due to different local minima of the new model, and because equivalent models can be represented differently by permuting the hidden nodes.

In Tables 3.4 and 3.5, we have the top performing classes in the AP metric for both ResNet B + att, and ResNet B + att (Avg), along with the number of samples in the dataset for each. Most of the classes have high label quality estimations, and are very loud noises like Bagpipes, Emergency Vehicle, Change ringing, Civil Defense Siren, Crowing, Battle cry, and Siren. This makes sense as they are more likely to have a good signal to noise ratio and unlikely to be crowded out or strongly conflicted by other sounds that are present in the clip. In Tables 3.6 and 3.7, we have the bottom performing classes in the AP metric for ResNet B + att and ResNet B + att (Avg). All of these labels seem to be less object-specific and fairly vague. They also have mostly quite poor label quality estimations, except for the acoustic environment labels "Inside, large room or hall" and "Outside, rural or natural" which may be particularly difficult to learn to predict. In Table 3.8, we have the 20 classes that most improved in performance from using a checkpoint averaged model. There seems to be little relation between the quality estimate or number of training samples and the amount of improvement that a class will see.

The per-class performance of these two models gives some qualitative insight into which classes and class-types are easier or more difficult to perform well on, and which benefit the most from checkpoint averaging. Not all class performances benefit from checkpoint averaging equally and some even decrease, as seen in the comparisons between Tables 3.5 and 3.7 and Tables 3.4 and 3.6.

Class	AP	AUC	# training samples	Quality estimate
Squish	0.0139	0.8300	316	20
Scrape	0.0196	0.8293	357	20
Buzz	0.0235	0.8122	390	50
Burst, pop	0.0255	0.8850	2217	40
Rattle	0.0375	0.8154	1012	50
Inside, large room or hall	0.0386	0.8286	27813	78
Creak	0.0400	0.8875	89	11
Harmonic	0.0442	0.9111	687	17
Bang	0.0455	0.9187	300	50
Outside, rural or natural	0.0455	0.8379	35546	100

Table 3.6: Bottom 10 classes in AP for the checkpoint averaged model, ResNet B + att (Avg), in results on the AudioSet evaluation set. # training samples refers to the number of samples labeled with this class in the AudioSet training set, and quality estimate refers to a quality estimate of the class labeling in AudioSet, given with the release of the dataset.

Class	AP	AUC	# training samples	Quality estimate
Squish	0.0143	0.8462	316	20
Scrape	0.0182	0.8264	357	20
Buzz	0.0268	0.8220	390	50
Burst, pop	0.0287	0.8822	2217	40
Rattle	0.0347	0.8218	1012	50
Inside, large room or hall	0.0378	0.8210	27813	78
Mouse	0.0411	0.8005	353	44
Outside, rural or natural	0.0425	0.8373	35546	100
Pulse	0.0446	0.9096	145	63
Bang	0.0447	0.9212	300	50

Table 3.7: Bottom 10 classes in AP for the best individual model, ResNet B + att, in results on the AudioSet evaluation set. # training samples refers to the number of samples labeled with this class in the AudioSet training set, and quality estimate refers to a quality estimate of the class labeling in AudioSet, given with the release of the dataset.

Class	AP individual	AP averaged	Net change	# training samples	Quality estimate
Drum kit	0.4574	0.5452	0.0877	15087	100
Rimshot	0.4412	0.5217	0.0805	4452	40
String section	0.3630	0.4421	0.0791	1132	40
Gobble	0.6880	0.7648	0.0768	849	50
Gargling	0.4256	0.4979	0.0723	77	90
Turkey	0.6995	0.7714	0.0719	1083	67
Acoustic guitar	0.3888	0.4585	0.0697	14476	100
Goose	0.5853	0.6524	0.0671	1730	67
Bass drum	0.4928	0.5584	0.0656	9203	100
Toot	0.4880	0.5534	0.0654	693	80
Keyboard (musical)	0.3817	0.4467	0.0650	10325	90
Bow-wow	0.4833	0.5467	0.0634	3810	50
Clickety-clack	0.3783	0.4417	0.0634	1968	70
Coo	0.6359	0.6968	0.0609	2858	80
Piano	0.2307	0.2904	0.0597	11432	70
Honk	0.5602	0.6196	0.0594	1769	89
Insect	0.6667	0.7260	0.0593	2996	80
Chicken, rooster	0.7064	0.7643	0.0578	6266	90
Guitar	0.5791	0.6359	0.0568	51291	80
Banjo	0.6945	0.7506	0.0561	2396	88

Table 3.8: Top 20 improved classes in AP between the best individual model, ResNet B + att, and the checkpoint averaged model, ResNet B + att (Avg), in results on the AudioSet Evaluation set. # training samples refers to the number of samples labeled with this class in the AudioSet training set, and quality estimate refers to a quality estimate of the class labeling in AudioSet, given with the release of the dataset.

Chapter 4

Sound Event Detection

With the addition of the attention module, we are curious to see whether our strongest model is not only able to classify sound events but also locate when the events actually occur in time. This particular task has many different names in the literature. In this thesis, we define sound event detection as the task of predicting the start time and end time of a sound event. Training a model for this particular task requires the start times and end times of all the sound events in a dataset. Collecting these labels is time-consuming and expensive. It would significantly reduce the cost of data collection if a model trained only for acoustic scene classification can be used for sound event detection, as weakly annotating data is a much easier task than strongly annotating. This has been the basis of work such as (Kumar and Raj, 2016). Even if strong labels are still needed for the development of particular systems, systems built from weakly labeled data could be used to assist and make the annotation process more efficient. In this chapter, we will study this problem in detail.

In general, we say that a dataset is weakly labeled when every audio clip has labels indicating whether an event occurs in the audio clip, and a dataset is strongly labeled when every event is labeled with its start time and end time. Though we focus on sound events in this thesis, the events can be other types, such as phonemes or words in speech recognition or human actions in action recognition from videos. Though the task is called sound event detection, we refer to the general task as segmentation, and will use the terms interchangeably. The general approach developed here can

hopefully be applied to other domains as well.

To study whether our model trained for acoustic scene classification is able to perform sound event detection, we will extract the pre-attention results after the model reads an audio clip and evaluate them on a strongly labeled dataset. Since the pre-attention results for each frame lie between 0 and 1, we explore a few approaches to convert them to hard boundaries based on thresholding.

4.1 Dataset

To evaluate our model, we made use of data from task 4 of DCASE 2018, "Sound event detection in domestic environments" (Serizel et al., 2018). This dataset is a subset of AudioSet, but only uses 10 labels from the ontology. The data consists of a training set which contains a weakly labeled portion of about 1,600 clips and 2,200 class occurrences, an unlabeled in-domain portion of about 14,400 clips that truly contain the relevant labels, and an unlabeled out-of-domain portion of about 40,000 clips which mostly do not truly contain the relevant labels. There is also a dev set, which contains 288 strongly labeled clips. At the end of the DCASE challenge an evaluation set was released for final submission evaluation, which is a similarly sized set of strongly labeled audio clips.

4.2 Methods and Results

We used the strongly labeled portions of the data which were the dev set released at the start of the challenge, and the final evaluation set. To perform sound event detection with our model architecture, we took the output right before attention is applied over time and applied a class-dependent threshold to determine activation. We show results for a threshold of 0.5 for all classes, as well as fine-tuned thresholds which were found by sweeping over values and evaluating on the dev set by each class's F1 score. We then also swept over kernel sizes of 3, 5, 7, 9, or 11 for class-specific median filters, allowing for smoothing over activation values before thresholds

Model	Dev	Eval
DCASE Baseline	14.06	-
JiaKai (2018)	25.9	32.4
0.5 Thresholds	9.83	6.70
Swept Thresholds (ST)	12.41	8.70
median filtering and ST	17.87	11.50

Table 4.1: Event Detection Performance of our tested approaches on task 4 of DCASE 2018 (F1 Macro %). Using our ResNet B + att model, we tested making hard labeling decisions using 0.5 thresholds, sweeping over thresholds, and sweeping over median filter sizes and thresholds for best F1 scores per class.

are applied. A median filter kernel size of 5 means that for each value in our array, we replace it with the median of the set of the current value, the 2 previous values, and the next 2 values in the array.

Figures 4-1, 4-2, 4-3, and 4-4 display the spectrograms and activation levels for different sound events, showing some of the successes and failures of the model in event detection. For each figure, on the bottom we can see the spectrogram of the audio clip, with frequency on the y-axis and the frame number on the x-axis. On top we have the activations for the top-2 classes on the y-axis over the frame number on the x-axis, represented as red and blue curves.

Performance on this task is evaluated with the macro-averaged F1 score. This means finding the F1 score per label and taking the unweighted mean of these values. To compute an F1 score, there needs to be a way to determine a hit or miss for each strong label. For this particular challenge, a labeling is determined to be a hit if both the starting and end times are within 200ms of the ground-truth (a 200ms "collar"), and the event duration is no less than 80% of the true event duration (Mesaros et al., 2016b). A labeling that does not fit these criteria is a false-positive, and any ground-truth label that does not have a corresponding "hit" labeling is considered a false-negative.

Our results are shown in Table 4.1. There are a few things to note in reviewing these results. For one, our model has a time resolution of about 300ms but the

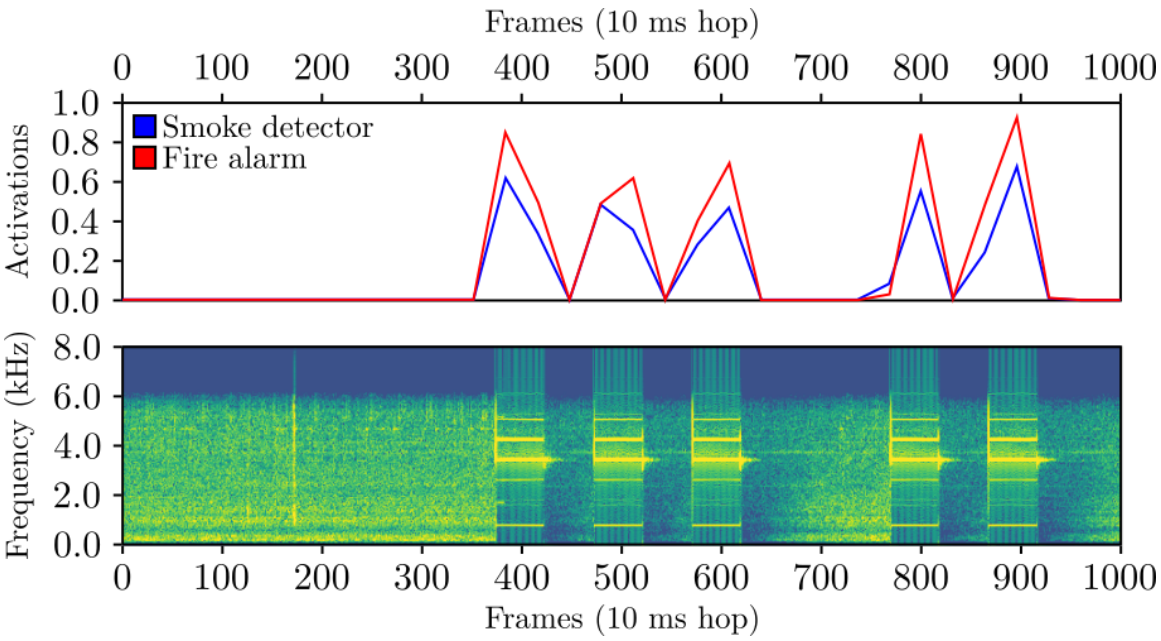


Figure 4-1: Here we see clear detection of a fire alarm by our model. We can see that there are 5 blocks of green in the bottom spectrogram starting at frame 375 — this is the fire alarm going off in the audio. Our model accurately activates during these times and deactivates everywhere else.

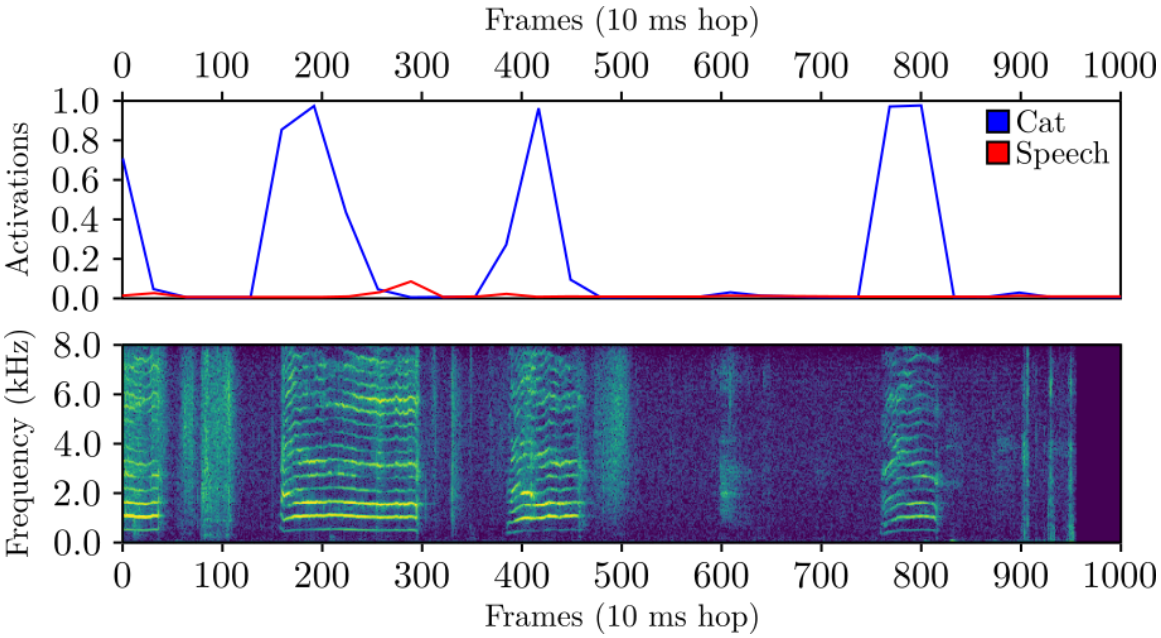


Figure 4-2: Here we see clear detection of a cat by our model. We can see that there are 4 stretches of green harmonic structure in the bottom spectrogram — this is the cat making noises in the audio clip. Our model accurately activates during these times and deactivates everywhere else.

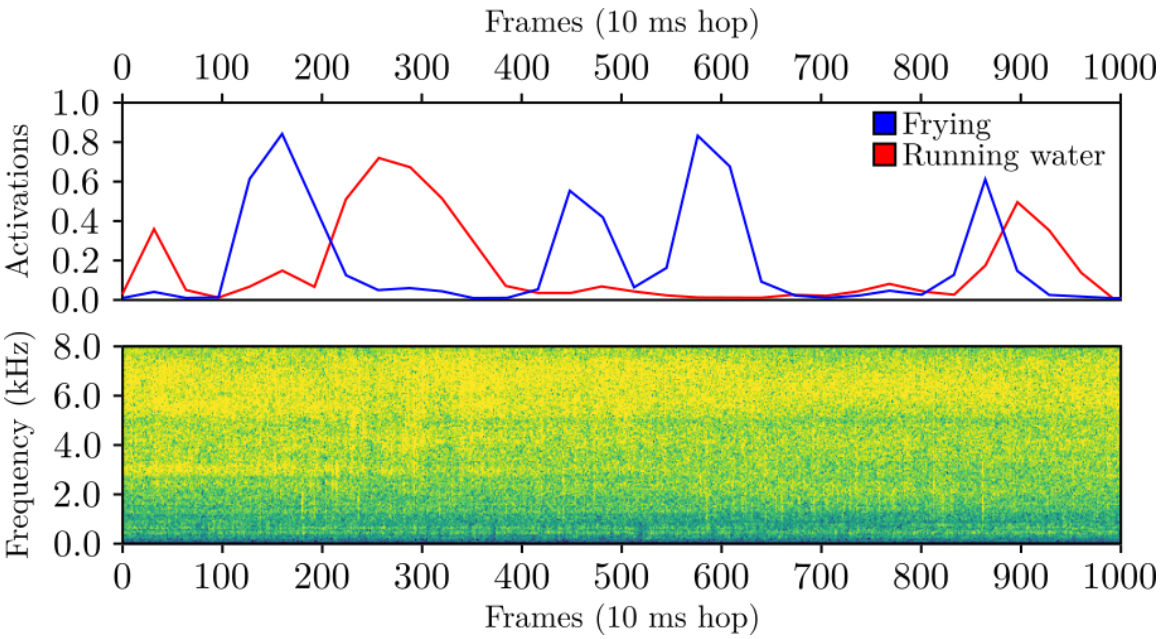


Figure 4-3: The continuous sound of frying is seen in the spectrogram as bright green and yellow. The model fails to consistently detect frying despite its presence throughout the clip, and it misfires on the 'Running water' class several times throughout the duration of the clip.

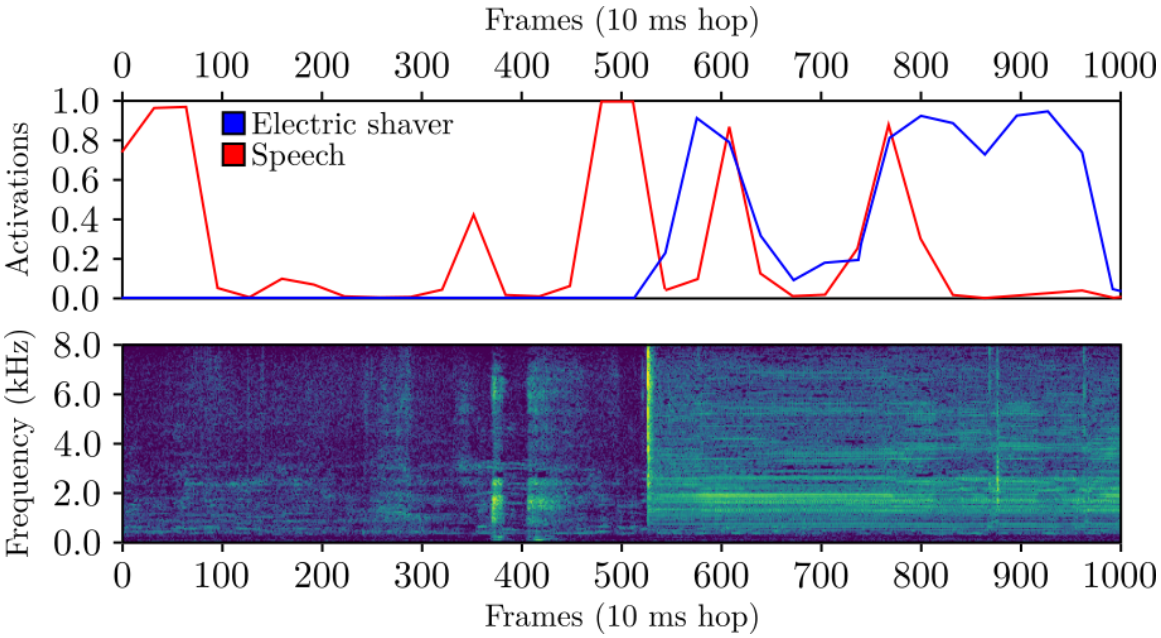


Figure 4-4: In this figure, we can see a clear shift in the spectrogram around frame 520 when there is more spectral activity. This is the point at which the Electric shaver is turned on, but its detection by our model is inconsistent throughout this activity. While it's difficult to see in the spectrogram, there is light background mumbling that seems to be triggering Speech detection.

Class	Our model	Baseline
Alarm/bell/ringing	**9.8**	3.9
Blender	3.4	**15.4**
Cat	**40.0**	0.0
Dishes	**20.5**	0.0
Dog	**21.1**	0.0
Electric Shaver/Toothbrush	24.5	**32.4**
Frying	12.5	**31.0**
Running Water	1.9	**11.4**
Speech	**27.3**	0.0
Vacuum Cleaner	17.9	**46.5**

Table 4.2: Event Detection Performance by Class of our best approach, median filtering and swept thresholds, applied to the ResNet B + att model on task 4 of DCASE 2018 (F1 %).

evaluation is done with a 200ms collar. This means that there is naturally going to be some loss in performance purely due to this discretization. When the time collar is increased to 400ms, Macro F1 performance increases to 31.25% on the dev set and 22.26% on the final eval set, indicating that the model is detecting many of the events but with less precision. In the release of the baseline, they claim that their model doesn't seem to be really learning how to perform sound event detection as it's only succeeding with long-duration class types, which in many clips is essentially equivalent to a clip-level classification. In contrast, our model performs particularly well with short-duration events, and worse with longer-duration events, as seen in Table 4.2. This shows that our model is in fact learning how to segment given the success with short-duration event types. A possible reason for failure in long-duration events is that the model found some features in time to be more strongly indicative of a classification result, leading to inconsistent detection throughout the event's entirety. There is also some belief in (JiaKai, 2018) that our resampling of 16kHz may be too aggressively low, and that it could make distinguishing between classes like vacuum cleaner, electric shaver, and electric toothbrush more difficult.

Chapter 5

Audio-Visual Scene Understanding and Transfer Learning

Given the strong performance of our proposed model, in this Chapter, we explore three potential extensions, one that includes vision as a second modality for acoustic scene understanding, another that studies how well the trained model transfers across datasets, and the other that turns our best model into a real-time detection system.

5.1 Multimodal Learning

Multimodal learning uses two or more modalities of information, such as the combination of auditory and visual recordings, to enable or enhance learning. Work in (Harwath et al., 2016) demonstrated the ability of two distinct models for audio and visual processing to assign certain spoken words from audio captioning to related areas of interest in an image. This was done with a triplet loss function, which ensures that points of connected or similar data are closer together and points of dissimilar data are farther apart in an embedding space. DAVEnet, a network architecture short for deep audio-visual encoding network proposed in (Harwath et al., 2016), has both the visual and auditory models map to a 1024 dimensional embedding space, and makes sure that matching image/caption pairs are close together, while mismatched pairs are farther apart.

5.1.1 Original DAVEnet

Instead of using pairs of images and spoken language captions for training, we used pairs of images and audio clips from the scene. We used the audio clips from our AudioSet-100k set, which comprises more than 250 hours, and we extracted the median video frame from each original YouTube clip. We first trained the original DAVEnet on this dataset, and qualitatively searched for patterns in the top scoring example images in several of the embedding dimensions. Examples are shown in Figures 5-1, 5-2, and 5-3. There was some loose patterning: dimension 1 contained images of text on blank backgrounds or people in light tan or pink coloring, dimension 2 contained lots of dark blue/purple dominant images and high static images, and dimension 3 contained many pink/red images, 2 of which also appeared in our dimension 1 set. The associated audio did not seem to be semantically interpretable.

5.1.2 DAVEnet with additional classification loss

We next explored whether the addition of the visual branch and triplet loss can assist in training the audio branch for classification on the AudioSet task compared to training the audio branch alone. This was done by adding binary cross entropy loss function on audio samples to our original DAVEnet loss equation. We then added a hyperparameter to tune the weighting between this BCE loss and the original triplet loss in training, effectively tuning the amount of influence of visual information on the audio model. Our results in Table 5.1 show that performance increased with use of the pre-trained visual model (only applicable when the visual model is utilized), but that overall the use of the triplet loss strictly decreased overall classification performance.

In looking through many of these example images and others in the dataset, it's clear that many of the visual components of the AudioSet videos are not of real world scenes or at all connected to the audio being played. This could be cause for the lack of obvious semantic patterns in each dimension. It also could be why addition of the visual branch and triplet loss does not enhance performance on the AudioSet classification task, as these examples may be more distracting than helpful to the model in

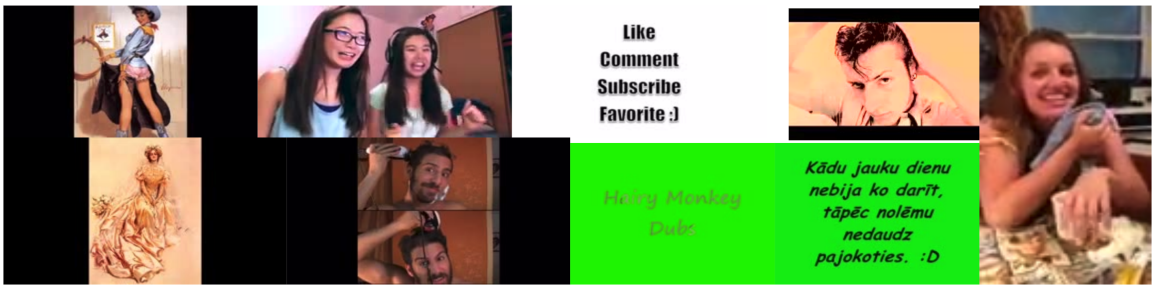


Figure 5-1: Examples from dimension 1 of the visual embedding space of DAVEnet. We see text on plain backgrounds, and people with tan coloring.

Figure 5-2: Examples from dimension 2 of the visual embedding space of DAVEnet. Examples contain shades of purple and many have high variance patterns, such as static.

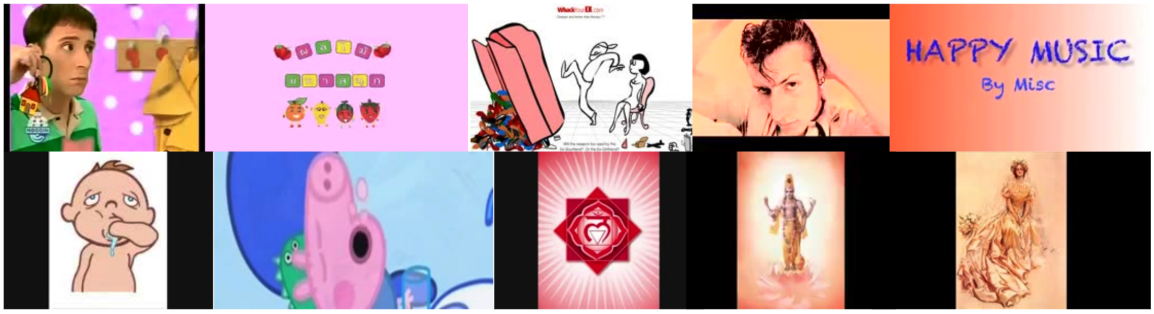


Figure 5-3: Examples from dimension 3 of the visual embedding space of DAVEnet. Most examples are pink and red in color.

Model	mAP	AUC	d-prime
DAVE audio-visual (0.5/0.5)	0.0488	0.835	1.378
DAVE audio-visual (0.5/0.5) (**PT**)	0.0815	0.864	1.555
DAVE audio-visual (0.65/0.35) (**PT**)	0.0844	0.866	1.567
DAVE audio-visual (0.8/0.2) (**PT**)	0.0931	0.879	1.651
DAVE audio-visual (0.95/0.05) (**PT**)	0.1123	0.899	1.801
DAVE audio-visual (1.0/0.0)	0.1444	0.911	1.905

Table 5.1: Model Performance of differently weighted DAVE audio-visual models on AudioSet Evaluation set. Note that (0.8/0.2) indicates a weighting of 0.8 for the BCE loss and 0.2 for the original DAVEnet triplet loss, and (**PT**) indicates the visual model being pre-trained on ImageNet

suggesting semantically useful representations. There may be promise in audio-visual learning for AudioSet task performance, potentially with a more performant audio model, expansion beyond AudioSet-100k, or with the sampling of more frames from each YouTube video.

5.2 Transfer Learning

Transfer learning is the use of a model that is pre-trained on outside data before 'fine-tuning' through training for a different domain or dataset. It is particularly useful when little data is available for a specific use-case, but there are large datasets of related tasks available. Using a model that is pre-trained on a larger amount of similar domain data can increase performance and increase training speed on new data and decrease the need for more labor involved data labeling. For our model trained on AudioSet, we wanted to explore how this model would perform on a separate dataset without fine-tuning to the specific domain.

5.2.1 Dataset

For testing out-of-domain performance, we utilized data from task 2 of DCASE 2018. This dataset was made up of audio clips from FreeSound, uploaded by the FreeSound

community, labeled with 41 labels from the AudioSet Ontology. The data is weakly labeled, but there is only one label per clip. The dataset is made up of a training set of about 9,500 samples, with each label being represented by at least 94 and no more than 300 samples. One difference in this domain is that the audio samples have a range of duration from 300ms to 30 seconds. About 3,700 samples of the training set has been manually verified and have been marked as such. The test set is made up of about 1,600 manually-verified samples and similar class weightings.

5.2.2 Results

We tested our best non-ensemble model (ResNet B + att) directly on the test set by restricting the output classes to those outlined in the task and taking the highest output of those. This achieved an mAP@3 result of 0.4307, which is significantly lower than the mAP@3 results of the baseline model of 0.7, despite being exposed to a much larger corpus of data. A possible explanation for this is the discrepancy in typical input length, the increased discriminative power of tuning the model to a constrained number of classes, or simply a lack of exposure to the new domain.

5.3 Real-time Sound Event Detection System

An area of interest for us was to apply the audio classification model in real-time, both for demo purposes and to qualitatively observe the utility of our model for such a real world application. We created a Python program that continuously streams audio input from the computer microphone, calculates a feature matrix, and runs it through our model for per class sigmoid outputs. We found the model to be quite good at tasks such as speech detection, distinguishing between different instruments or types of music, and recognizing animals, all while running in a new environment and with a new recording device. We used our best epoch ResNet B + att model for good classification accuracy at a usable latency and update rate of 800ms. The output window of the system is shown in Figure 5-4. To be able to run this model at this fast of an update rate on a laptop CPU machine, we needed to use a smaller

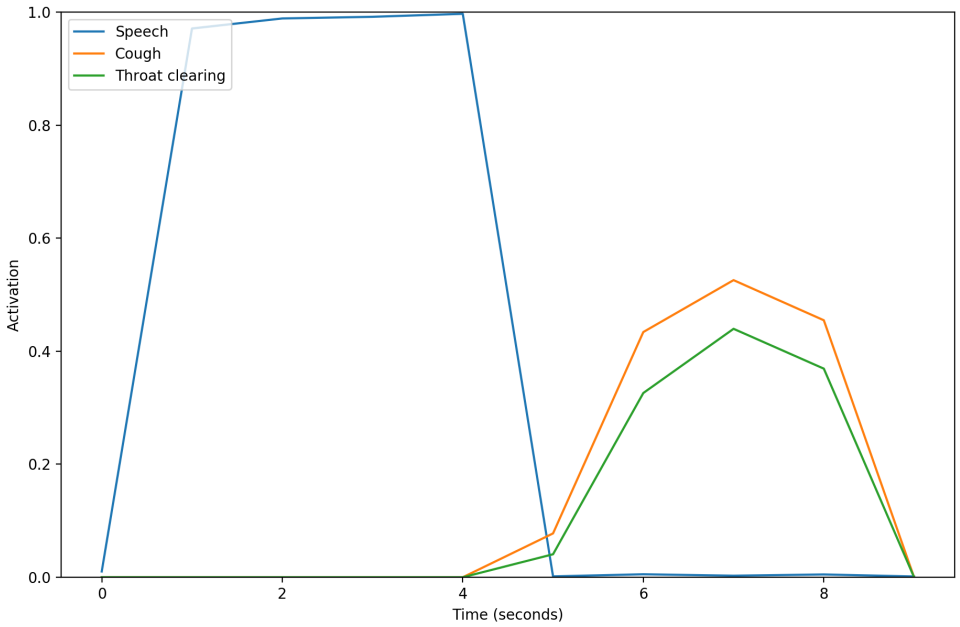


Figure 5-4: A screenshot of the real-time sound event detection system. It displays the top 3 detected classes (color-coded, see legend) and their output values as lines over the previous 10 seconds, updated every 800ms. The model takes as input the previous 2 seconds of audio for every update.

input window of 2 seconds. This is in contrast to the typical training sample's input length of 10 seconds, and the receptive field size of slightly over 4 seconds. Despite this cut, the model is still able to recognize many sounds quite robustly.

Here, we optimized for speed and accuracy, with no real restriction in hardware usage beyond the manipulation of input size to allow for timely computation without an onboard GPU. Our model can run inference with a 2 second input in 700ms on a 2.5 GHz Intel Core i7 CPU machine, with single-threading. For real world real-time use cases, there are many considerations to balance. The speed, accuracy, and output requirements must be considered against the computational resources and power usage available. Such systems have already been deployed for applications like gunshot detection[1] or glassbreak detection[2].

[1] `https://www.shotspotter.com`

[2] `https://www.security.honeywell.com/All-Categories/intrusion/sensors/glassbreak-detector`

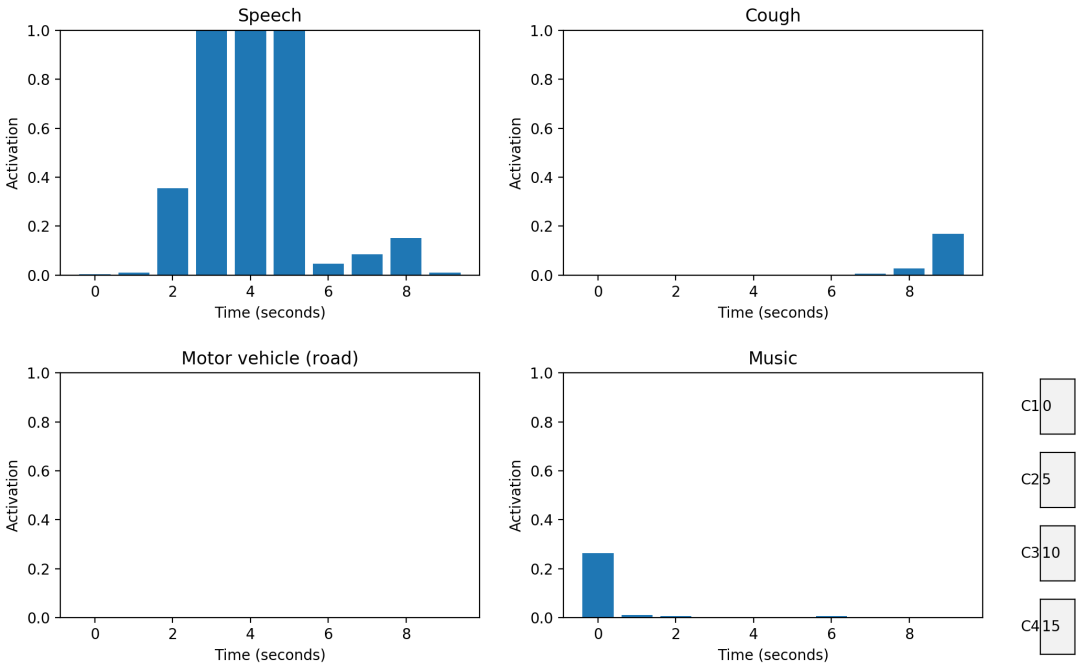


Figure 5-5: A screenshot of an earlier iteration of the real-time sound event detection system. It displays 4 chosen classes and their output values in the form of a bar graph over the previous 10 seconds, updated every 800ms. The model takes as input the previous 2 seconds of audio for every update.

Chapter 6

Conclusion

6.1 Summary of Findings

This thesis presents a model that outperforms previous state-of-the-art work on the AudioSet dataset labeling task, surpassing results from models built upon the released bottleneck features, achieving an mAP of 0.392 and AUC of 0.971. This indicates that there is promise in further exploring model types that learn directly from log-Mel features of the audio samples. Our work aligned with that of (Hershey et al., 2017) in that ResNet variants are indeed particularly performant models for audio classification. We also validated that the addition of an attention module helps significantly in improving the performance of CNN architectures for audio classification. The attention module has also shown to perform segmentation in order to achieve a final classification and can be used to extract strong labels for acoustic events. We've also shown how these models can be used for multimodal learning, transfer learning, and real-time applications.

6.2 Future Work

One area of potential focus is to improve the segmentation ability of models trained for the AudioSet task. Work here has indicated that there is a potential greater need for capturing long-duration acoustic characteristics. Model architectures employing

some form of recurrent structure could be particularly useful in more accurately performing segmentation and increasing confidence of detecting a particular sound given surrounding key features. Use of intermediate layers could be helpful for more fine-grained time labeling to increase performance on segmentation tasks.

Given the difficulty of labeling audio data and the noise present in datasets like AudioSet, Golden Label Correction from (Hendrycks et al., 2018) could be particularly useful in combatting noisy labels and understanding the ways that humans tend to misannotate audio samples. Better understanding how humans tend to misannotate audio samples can inform future architectures and training frameworks, how we frame the problem, and the tools we give to labelers for annotating data.

Bibliography

S. H. Bae, I. Choi, and N. S. Kim. Acoustic scene classification using parallel combination of LSTM and CNN. In *Proc. DCASE*, pages 11–15, 2016.

M. Büchler, A. Silvia, S. Launer, and N. Dillier. Sound classification in hearing aids inspired by auditory scene analysis. *EURASIP Journal on Advances in Signal Processing*, 18, 01 2005. doi: 10.1155/ASP.2005.2991.

E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM IEEE Audio, Speech, Language Process.*, 25(6):1291–1303, 2017.

H. Chen, S. Lundberg, and S.-I. Lee. Checkpoint ensembles: ensemble methods from a single training process. In *arXiv:1710.03282*, 2017a.

Z. Chen, Z. Xie, W. Zhang, and X. Xu. ResNet and model fusion for automatic spoofing detection. In *INTERSPEECH*, 2017b.

N. Cho and E.-K. Kim. Enhanced voice activity detection using acoustic event detection and classification. *IEEE Trans. Consum. Electron.*, 57(1):196–202, 2011.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

G. Dekkers, S. Lauwereins, B. Thoen, M. W. Adhana, H. Brouckxon, B. Van den Bergh, T. van Waterschoot, B. Vanrumste, M. Verhelst, and P. Karsmakers. The SINS database for detection of daily activities in a home environment using an acoustic sensor network. *Detection and Classification of Acoustic Scenes and Events 2017*, 2017.

S. Dutta and A. Ghosal. A hierarchical approach for silence/speech/music classification. In *Proc. IEEE ICPCSI*, pages 3001–3005, 2017.

J. T. Geiger, M. A. Lakhal, B. W. Schuller, and G. Rigoll. Learning new acoustic events in an HMM-based system using MAP adaptation. In *INTERSPEECH*, 2011.

J. T. Geiger, B. Schuller, and G. Rigoll. Large-scale audio feature extraction and SVM for acoustic scene classification. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, Oct 2013. doi: 10.1109/WASPAA.2013.6701857.

J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *IEEE ICASSP*, pages 776–780, 2017.

F. Grondin and F. Michaud. Robust speech/non-speech discrimination based on pitch estimation for mobile robots. In *Proc. IEEE ICRA*, pages 1650–1655, 2016.

G. Guo and S. Z. Li. Content-based audio classification and retrieval by support vector machines. *IEEE trans. Neural Networks*, 14(1):209–215, 2003.

D. Harwath, A. Torralba, and J. Glass. Unsupervised learning of spoken language with visual context. In *Proc. NeurIPS*, pages 1858–1866, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.

D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*, 2018.

S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson. CNN architectures for large-scale audio classification. In *Proc. IEEE ICASSP*, pages 131–135, 2017.

A. Jansen, M. Plakal, R. Pandya, D. P. W. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous. Unsupervised learning of semantic audio representations. In *IEEE ICASSP*, 2018.

B. D. Jean-Julien Aucouturier and F. Pachet. The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. In *The Journal of the Acoustical Society of America*, 2007.

L. JiaKai. Mean teacher convolution system for DCASE 2018 task 4. In *Proc. DCASE*, 2018.

H. Jiang, J. Bai, S. Zhang, and B. Xu. SVM-based audio scene classification. In *2005 International Conference on Natural Language Processing and Knowledge Engineering*, pages 131–136, Oct 2005.

N. Ketkar. Introduction to PyTorch. In *Deep learning with Python*, pages 195–208. Springer, 2017.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley. Audio Set classification with attention model: a probabilistic perspective. In *Proc. IEEE ICASSP*, pages 316–320, 2018.

Q. Kong, C. Yu, Y. Xu, T. Iqbal, W. Wang, and M. D. Plumbley. Weakly labelled AudioSet tagging with attention neural networks. In *arXiv:1903.00765*, 2019.

J. D. Krijnders and G. A. ten Holt. Tone-fit and MFCC scene classification compared to human recognition. In *PersonalCommunication*, 2013.

A. Kumar and B. Raj. Audio event detection using weakly labeled data. *ACM Multimedia*, pages 1038–1047, 2016.

E. Larsen, C. D. Schmitz, C. R. Lansing, W. D. O'Brien, B. C. Wheeler, and A. S. Feng. Acoustic scene analysis using estimated impulse responses. In *The Thirty-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 1, pages 725–729 Vol.1, Nov 2003. doi: 10.1109/ACSSC.2003.1292009.

C.-Y. Lee, P. W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pages 464–472, 2016.

S.-C. Liu, J. Bi, Z.-Q. Jia, R. Chen, J. Chen, and M.-M. Zhou. Automatic audio classification and speaker identification for video content analysis. In *Proc. IEEE/ACIS SNPD*, volume 2, pages 91–96, 2007.

Z. Liu, Y. Wang, and T. Chen. Audio feature extraction and analysis for scene segmentation and classification. *VLSI Signal Processing*, 20:61–79, 1998.

L. Lu, H. Jiang, and H. Zhang. A robust audio classification and segmentation method. In *Proc. ACM Multimedia*, pages 203–211, 2001.

L. Lu, H.-J. Zhang, and H. Jiang. Content analysis for audio classification and segmentation. *IEEE Trans. Speech & Audio Process.*, 10(7):504–516, 2002.

A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley. Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge. *IEEE/ACM TASLP*, pages 379–393, 2016a.

A. Mesaros, T. Heittola, and T. Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162, 2016b.

A. Mesaros, T. Heittola, and T. Virtanen. TUT database for acoustic scene classification and sound event detection. In *Proc. EUSIPCO*, pages 1128–1132, 2016c.

A. B. Nielsen, L. K. Hansen, and U. Kjems. Pitch based sound classification. In *Proc. IEEE ICASSP*, volume 3, pages 788–791, 2006.

D. Patrick Whittlesey Ellis. *Prediction-driven computational auditory scene analysis*. PhD thesis, Massachusetts Institute of Technology, 06 1996.

K. J. Piczak. Environmental sound classification with convolutional neural networks. In *Proc. IEEE MLSP*, pages 1–6, 2015a.

K. J. Piczak. ESC: Dataset for environmental sound classification. In *Proc. ACM Multimedia*, pages 1015–1018, 2015b.

T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer. On rhythm and general music similarity. In *Proc. of the 10th Int. Society for Music Information Retrieval Conf.*, 2009.

S. Ravindran and D. V. Anderson. Audio classification and scene recognition and for hearing aids. In *Proc. IEEE ISCAS*, pages 860–863, 2005.

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

Y. Sasaki, M. Kaneyoshi, S. Kagami, H. Mizoguchi, and T. Enomoto. Daily sound recognition using pitch-cluster-maps for mobile robot audition. In *Proc. IEEE/RSJ IROS*, pages 2724–2729, 2009.

R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. P. Shah. Large-scale weakly labeled semi-supervised sound event detection. In *arXiv:1807.10501*, 2018.

K. Seyerlehner, G. Widmer, and P. Knees. Fusing block-level features for music similarity estimation. In *Proc. of the 13th Int. Conference on Digital Audio Effects*. DAFx, 2010.

K. Seyerlehner, G. Widmer, and P. Knees. A comparison of human, automatic and collaborative music genre classification and user centric evaluation of genre classification systems. In M. Detyniecki, P. Knees, A. Nürnberger, M. Schedl, and S. Stober, editors, *Adaptive Multimedia Retrieval. Context, Exploration, and Fusion*, pages 118–131. Springer Berlin Heidelberg, 2011.

D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley. Detection and classification of acoustic scenes and events. *IEEE Trans. Multimedia*, 17(10): 1733–1746, 2015. ISSN 1520-9210. doi: 10.1109/TMM.2015.2428998.

A. Temko, R. G. Malkin, C. Zieger, D. Macho, C. Nadeu, and M. Omologo. CLEAR evaluation of acoustic event detection and classification systems. In *CLEAR*, pages 311–322, 2006.

C. Yu, K. S. Barsim, Q. Kong, and B. Yang. Multi-level attention model for weakly supervised audio classification. In *arXiv:1803.02353*, 2018.

H. Zhang, I. McLoughlin, and Y. Song. Robust sound event recognition using convolutional neural networks. In *IEEE ICASSP*, pages 559–563, 2015.