

# **Audio Event Classification Using Deep Learning Methods**

**Zhicun Xu**

**School of Electrical Engineering**

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 26.11.2018

**Thesis supervisor:**

Prof. Mikko Kurimo

**Thesis advisor:**

M.Sc. Peter Smit



Author: Zhicun Xu		
Title: Audio Event Classification Using Deep Learning Methods		
Date: 26.11.2018	Language: English	Number of pages: 6+67
Department of Signal Processing and Acoustics		
Professorship: Acoustics and Audio Technology		
Supervisor: Prof. Mikko Kurimo		
Advisor: M.Sc. Peter Smit		
<p>Whether crossing the road or enjoying a concert, sound carries important information about the world around us. Audio event classification refers to recognition tasks involving the assignment of one or several labels, such as ‘dog bark’ or ‘doorbell’, to a particular audio signal. Thus, teaching machines to conduct this classification task can help humans in many fields. Since deep learning has shown its great potential and usefulness in many AI applications, this thesis focuses on studying deep learning methods and building suitable neural networks for this audio event classification task. In order to evaluate the performance of different neural networks, we tested them on both Google AudioSet and the dataset for DCASE 2018 Task 2. Instead of providing original audio files, AudioSet offers compact 128-dimensional embeddings outputted by a modified VGG model for audio with a frame length of 960ms. For DCASE 2018 Task 2, we firstly preprocessed the soundtracks and then fine-tuned the VGG model that AudioSet used as a feature extractor. Thus, each soundtrack from both tasks is represented as a series of 128-dimensional features. We then compared the DNN, LSTM, and multi-level attention models with different hyper parameters. The results show that fine-tuning the feature generation model for the DCASE task greatly improved the evaluation score. In addition, the attention models were found to perform the best in our settings for both tasks. The results indicate that utilizing a CNN-like model as a feature extractor for the log-mel spectrograms and modeling the dynamics information using an attention model can achieve state-of-the-art results in the task of audio event classification. For future research, the thesis suggests training a better CNN model for feature extraction, utilizing multi-scale and multi-level features for better classification, and combining the audio features with other multimodal information for audiovisual data analysis.</p>		
Keywords: audio event classification, AudioSet, multi-level attention model, VGG		

## Preface

The perception of sound is important for human beings. Having always been fascinated by sound and the theories behind it, I am grateful that I get this opportunity from Prof. Mikko Kurimo to write thesis related to machine listening.

I would like to thank the great guidance and patience from both my supervisor Mikko Kurimo and advisor Peter Smit. It is their help that makes me take on the right track and progress well. The thesis is funded by the Kone foundation, and the European Union's Horizon 2020 research and innovation programme via the project MeMAD (GA780069). This project gives me great opportunity to experience the research studies in such a diverse group with people from different organizations and research fields. I am also really thankful for that.

It was quite an experience making my thesis, hopefully my work can somewhat be useful in this research field.

Otaniemi, 26.11.2018

Zhicun Xu

# Contents

<b>Abstract</b>	ii
<b>Preface</b>	iii
<b>Contents</b>	iv
<b>Symbols and abbreviations</b>	vi
<b>1 Introduction</b>	1
<b>2 Background</b>	3
2.1 Applications and Motivations . . . . .	3
2.2 Previous Work . . . . .	4
2.3 Challenges . . . . .	7
2.4 Development of Deep Learning . . . . .	8
<b>3 Research Theory and Methods</b>	14
3.1 Machine Learning Basics . . . . .	14
3.2 The Theory of Deep Learning . . . . .	17
3.2.1 Feedforward Neural Network . . . . .	17
3.2.2 Activation Function . . . . .	18
3.2.3 Loss Functions and Backpropagation . . . . .	19
3.2.4 Regularization . . . . .	21
3.2.5 Optimization . . . . .	25
3.2.6 Batch Normalization . . . . .	27
3.2.7 Convolutional Neural Networks . . . . .	29
3.2.8 Recurrent Neural Networks and LSTM . . . . .	30
3.2.9 Multi-level Attention Model . . . . .	32
3.3 Log-Mel Spectrogram . . . . .	35
<b>4 Research Tasks</b>	37
4.1 Google AudioSet . . . . .	37
4.2 DCASE 2018 Task2 . . . . .	40
4.3 Evaluation Methods . . . . .	41
4.3.1 Mean Average Precision . . . . .	41
4.3.2 Area Under Curve . . . . .	42
<b>5 Experiments</b>	44
5.1 Google AudioSet . . . . .	44
5.1.1 VGGish Generated Features . . . . .	44
5.1.2 Mini-batch Balancing . . . . .	45
5.1.3 Different Model Structures . . . . .	45
5.2 DCASE 2018 Task2 . . . . .	46
5.2.1 Preprocessing . . . . .	46
5.2.2 Fine-tuning the VGGish Model . . . . .	47

5.2.3	Data Augmentation . . . . .	48
5.2.4	Mixup . . . . .	48
5.2.5	Different Model Structures . . . . .	49
<b>6</b>	<b>Results</b>	<b>51</b>
6.1	Google AudioSet . . . . .	51
6.2	DCASE 2018 TASK2 . . . . .	53
<b>7</b>	<b>Conclusion</b>	<b>58</b>
	<b>References</b>	<b>60</b>

# Symbols and abbreviations

## Abbreviations

Adam	adaptive moments
AI	artificial intelligence
AP	average precision
ASC	acoustic scene classification
ASR	automated speech recognition
AUC	area under curve
BLEU	bilingual evaluation understudy
CNN	convolutional neural networks
CPU	central processing unit
DCASE	detection and classification of acoustic scenes and events
DCGANs	deep convolutional generative adversarial networks
DNN	deep neural network
GAN	generative adversarial network
GMM	Gaussian mixture model
GPU	graphics processing unit
HMM	hidden Markov models
HPSS	harmonic-percussive sound separation
LSTM	long short-term memory
MAP	mean average precision
METEOR	metric for evaluation of translation with explicit ordering
MFCCs	Mel-frequency cepstral coefficients
MIR	music information retrieval
MIREX	music information retrieval evolution exchange
MLP	multilayer perceptron
NMF	non-negative matrix factorization
PCA	principal component analysis
PCM	pulse-code modulation
ReLU	rectified linear unit
RNN	recurrent neural network
ROC	receiver operating characteristic
RQA	recurrence quantification analysis
SAI	stabilized auditory image
SVM	support vector machine
TDNN	time-delay neural network
TRECVid	TREC video retrieval evaluation
VGG	visual geometry group

# 1 Introduction

Sound is important in our daily communication with the world, and recognizing sound events can provide us with a wide variety of information, such as localizations and identifications of sound sources, indications of possible dangers, genres of music or even emotions from speakers. Because of the importance and wide usage of sound, machines are needed to help and cope with hearing tasks for humans. One of the most important hearing tasks is audio event classification. This task involves the assignment of one or more descriptive labels, such as ‘male speech’ or ‘plucked string instruments’, to a particular audio signal. This task could provide many possibilities for the artificial intelligence (AI) industries in applications involving the perception of sound, such as sound print, public surveillance, smart homes, medical information monitoring, speech recognition, music information retrieval and multimedia content analysis.

Historically, many challenges related to audio event classification have been organized for research purposes, especially the challenge of Detection and Classification of Acoustic Scenes and Events (DCASE). DCASE is a challenge specifically intended for audio, and it summarizes the state-of-the-art in the machine listening field. Other related challenges include the genre and mood classification in Music Information Retrieval Evolution eXchange(MIREX), the CHiMe speech separation and recognition challenge, signal separation in Signal Separation Evaluation Campaign (SiSEC), and multimedia content analysis in TREC Video Retrieval Evaluation (TRECVid). The submissions for these challenges indicate the main research methods and trends for the task of audio event classification in recent years.

Most of the challenges only focus on the classification of a limited number of classes or certain clusters of classes, such as bird sounds or genres of music. Fortunately, Google has created a dataset called AudioSet [18], which has a carefully designed ontology where 527 classes are organized in a hierarchical tree structure. The AudioSet ontology covers a wide range of classes, including human, animal, nature, musical and miscellaneous sounds. Thus, AudioSet is a suitable research dataset for general-purpose audio event classification. Task 2 from DCASE 2018 is also an audio event classification task using the AudioSet ontology, even though it only uses 41 classes. Because of its narrower scope, this thesis will firstly perform experiments on Google AudioSet to obtain a general solution and then adapt this solution to DCASE 2018 Task 2. Both tasks are supervised learning tasks with labeled soundtracks. The difference is that each example from AudioSet can belong to multiple classes, while each example from the DCASE task only belongs to one class.

Traditionally, many methods for performing the task of audio event classification have been adapted from the speech recognition field, such as using mel-frequency cepstral coefficients (MFCCs) as features and Gaussian mixture models (GMMs) as classifiers [1]. Other generated features, which mainly use signal processing techniques, include the mel-frequency spectrum, stabilized auditory image (SAI) [53], and recurrence quantification analysis features (RQA) [58]. Certain studies have also incorporated many psychoacoustics features, such as loudness, brightness, pitch and timbre. GMMs, hidden Markov models (HMMs) [54], non-negative matrix

factorization (NMF) [37], and support vector machines (SVMs) were commonly used classifiers. However, recent research has shown that these traditional machine learning techniques, which have been employed in many fields, such as computer vision, speech recognition, and natural language processing, can be outperformed by deep learning. While a part of machine learning, deep learning is inspired by the communication patterns of our neural system and can perform all kinds of AI tasks using different neural network structures. In recent years, the DCASE challenges have seen the wide usage and effective performance of different neural networks.

Therefore, the aim of this thesis is to study deep learning methods in order to provide a general solution to the task of audio event classification. For the purpose of accomplishing that, this thesis will compare different neural network structures on both Google AudioSet and the dataset for DCASE 2018 Task 2. The results will be evaluated using the score of mean average precision (MAP). This metric is explained in Section 3.5. The scope of the thesis will be limited to the classification of the audio events. The thesis will not attempt to solve the audio event detection problem, where the onset and offset of the event are also needed. The proposed solution to the tasks mainly used convolutional neural networks (CNNs) for feature extraction from the log-mel spectrograms and multi-level attention models for recognition of the time series of features.

The remainder of this thesis is divided into five chapters. Chapter 2 presents the background of this task, including motivations, applications, challenges, reviews of previous work and development of deep learning. Chapter 3 explains in detail the research theory and methods, including the basics of machine learning and some important concepts of deep learning. Chapter 4 shows the task settings and dataset analysis for both Google AudioSet and DCASE 2018 Task 2. Chapter 5 defines the settings for the experiments, such as different model structures and hyper parameters. Chapter 6 provides and analyzes the results of the thesis. Chapter 7 concludes the thesis by discussing the performance of different deep learning models and making recommendations for further research.

## 2 Background

This section presents the background of the task of audio event classification. It firstly states the applications and motivations of this task for various fields, such as public surveillance, speech recognition, music information retrieval and entertainment industries. This section then demonstrates how methods for performing this task evolved along history and currently remaining challenges. In addition, the section introduces the development of deep learning. The development shows how deep learning gain its significant role after the birth of Rosenblatt's perceptron concept.

### 2.1 Applications and Motivations

Teaching machine to hear aims at building a 'smart environment' system that can converse with its occupants. This system is expected to conduct a variety of tasks, including keeping track of things, serving as a security, surveillance, and diagnostic system, and providing entertainment and communication services [40]. One of the most important tasks is audio event classification. This classification task can either be used directly or added into the pipeline of another hearing task. The applications and motivations of this task are stated here in this subsection.

Recognizing the source of general audio events is one of the most important goals of machine listening. For the purpose of public surveillance, indication of possible dangers is from the recognized abnormal audio events from our surrounding environments, such as gun-shot, scream, and glass crash. These cues of dangers could help reduce accident rates and provide faster responses. For example, if a machine hears a gunshot from a certain direction, then a security camera can quickly turn into that direction to capture important information. A robotic navigation system could also be built by using the recognized source information of sound events [7]. For example, the recognition results of an acoustic environment, such as indoor or outdoor, can help disabled people automatically adjust the control modes of their wheelchairs to adapt to different conditions of roads. In addition, recognizing sources of sound can benefit many other fields such as gathering ecological data [5] by recognizing species in the recordings of a environment, and generating sports highlights [71] in broadcast programs.

More accurate and robust speech recognition can be obtained using the classification results of audio event classification. A good audio event classifier can detect if an audio signal belongs to speech, music or general sounds. Therefore, it can provide tools for voice activity detection and segmentation. In addition, different speaker dependent models can be utilized for better recognition according to the further classification of speech types, such as male speech, female speech, child speech or cross-talk speech. Speech recognition accuracy can also be raised by recognizing the type of a noisy environment firstly and then using corresponding environmental noise adaptation [32]. The improvement is obvious, especially for noisy and highly reverberant environments.

In the field of music information retrieval(MIR), recognizing audio events can provide more detailed analysis of music content. The types of instruments, tempos,

and changing tones of music can more easily be recognized, since the recognition is more objective and based on analysis of shorter segments. On the contrary, the genres and emotions of music are highly subjective descriptions with long decision time-scale [6]. Therefore, recognition of these labels could be harder. However, this challenge could be overcome by using neural network classifiers with long time span. In addition, the amount of music content is increasing heavily these days, thus providing enough dataset for training more accurate neural network classifiers. Overall, a good music classifier can help archive music more efficiently and accurately. Music content providers can also utilize the classifier to provide their consumers useful applications, such as music recommendation and music generation.

Another important usage of audio event classification is multimedia content analysis. More accurate analysis can be obtained by the combination of classification results from audio signals and visual images. This analysis could also be used to generate audio descriptions of multimedia content. Traditionally, these multimedia content are annotated by humans with a limited number of labels, which is laborious, inefficient and less informative. Nowadays, since the amount of multimedia content is huge and still continually increasing, there exists need to better describe these content through machines for better accessing, archiving and hyper-linking. For example, if a sound designer is asked to produce a gun-shot in a cave, then he can search these keywords to find similar segments for reference or inspiration in a big corpus. Not only the people who work in Creative Industries will receive new methods from multimedia analysis. The analysis will also broaden the audiences groups. The visually impaired people can listen to the generated audio descriptions and people with hearing loss can read through the audio descriptions for better understanding the contents.

Recognizing audio events can bring more new possibilities for entertainment fields, such as gaming, virtual reality, and augmented reality. In shooting games, audio can give cues on the locations of footsteps and gunshots, thus players can react much faster and more accurately. A generative model could also be built along with an audio event classifier to generate certain types of sound. The generative model can help a production crew to generate more realistic sound textures, such as footstep and rain sounds, without repeating the same sound in sound effects library. The generated sound could also have different characteristics, such as distance, direction, and reverberation, thus providing users an immersive experience.

In conclusion, audio event classification is an important task in the field of machine listening. It is motivated by many different fields and can be utilized as a powerful tool in a wide variety of applications.

## 2.2 Previous Work

Audio event classification has a relatively shorter history than other sound-related fields, such as automated speech recognition (ASR) and MIR. One of the earliest work dates back to the 1990s, a system [70] evaluated on the 'Muscle Fish' database was created. The system represented a sound class by utilizing various of perceptual features, including loudness, brightness, pitch, and timbre. It uses Euclidean distance

as decision criterion to reorder sound based on computed likelihoods. The database contains 410 sounds in 16 classes varying from 1 to 15 seconds in length. Later, Liu *et al.* [39] created a feed-forward neural network using the similar perceptual features with the addition of sub-band energy ratios. Their work was evaluated on collected audio clips containing five classes: news, weather reports, advertisements, basketball and football games. The result showed the effectiveness of the discrimination capability of perceptual features.

MFCCs were firstly used in Foote's work [15] in 1997 with a quantization tree-template classifier for audio classification/retrieval. The output of this system is a list of audio files ranked by similarities to expected classes. Then this list was evaluated by the score of average precision (AP). AP can simply be interpreted as the ratio between the number of relevant documents and the number of top-ranked documents. Foote's results were also compared with the 'Muscle Fish' system [70] mentioned previously. The results showed that their schemes performed differently with respect to different types of classes. The 'Muscle Fish' system, which utilized many psychoacoustically-derived features, performed better in music-related classes which contains important timbre information. The tree-based classifier tended to classify sounds based on pitch other than timbre. Thus, the choice of classifiers might be application dependent. Gaussian mixture models (GMMs) then became main standard classifier for audio classification.

In 2003, Guo and Li [25] combined perceptual features from [70] and the cepstral coefficients which can capture the shape of a frequency spectrum [25]. They focused on support vector machines (SVMs) as classifiers for audio event classification. Their results showed that SVM-based method had lower error rates and higher retrieval accuracy than other previous methods for all testing-sets. Until now, SVM is still a popular approach for classifications.

Hidden Markov models(HMM) were also quite popular in this field. In 1998, Couvreur *et al.* [9] incorporated a HMM-based classifier for the environmental noise recognition of five classes. Their results showed that this approach outperformed the average-spectrum-based classifiers and worked better than human's recognition. In 2003, Nishiura *et al.* [51] also proposed a HMM-based environmental sound source identification system for robust speech recognition. Their system showed good accuracy in identifying environmental noise sounds, speech, and the mixture of them. Later in 2006, Ma *et al.* presents a HMM-based acoustic environment classifier that incorporates a adaptive learning mechanism and a hierarchical classification model [41]. The proposed classifier could provide good results on recognizing everyday environments.

Many previous works [8] [44] treated sound as monophonic with only one audio event appearing in the time segments. However, general audio events are much more complex and have polyphonic characteristics. Thus, recognition of these polyphonic audio needs special consideration. One possible solution is to apply source separation on the polyphonic audio and analyze the separated audio separately [52]. A research [19] treated a polyphonic audio as the linear-combination mixtures of many monophonic audio events. The weights for each class could be obtained by non-negative matrix factorization. However, sound source separation will continue

to be an unsolved problem [67].

Public challenges, such as DCASE, MIREX and TRECVID, are beneficial for benchmarking and evaluation of proposed solutions, providing further directions, and comparing different systems. DCASE 2016 had the task of audio event classification in task 4, but this task was only for domestic sounds. In this year, DCASE 2018 Task 2 extended the recognition task on more general sounds. DCASE also had similar challenges called acoustic scene classification, which could also provide useful materials for research in the task of audio event classification. In DCASE 2016, almost all submissions incorporated MFCCs as features because of their compact representation of spectrum. Other features included the CQT-based time-frequency representations [4] where they used the Constant Q-transform to generate time-frequency matrix for recordings, the RQA parameters that models the stationarity of MFCCs [58], the gradient histograms of spectrograms [57], and the linear regression coefficients of local features which contains trending information along time axis of audio signals [17]. All the previously mentioned features show the importance of modelling the time evolution of an audio signal. As for learning algorithms, almost all submission implemented discriminative learning algorithms, such as SVM, GMM, and HMM, except for one using an i-vector based system [12]. SVM seemed to be the optimal choice of classifiers by then, since almost all the submission that outperformed the baseline utilized SVM methods. The results of DCASE 2016 also revealed that traditional MFCC+GMM/HMM based pipeline had reached a ceiling [2], alternative paradigms were needed for stepping into a next level.

Because of the fast development of deep learning, the submissions of DCASE 2017 changed greatly. Almost all submissions used some forms of neural networks, such as DNN, CNN, and RNN. In addition, log-mel energies and different forms of spectrograms were used in almost all the submissions that outperformed baseline results. The work achieved the best performance [48] used convolution neural networks (CNNs) on log-mel energies for extracting local frequency information and used long short-memory units (LSTMs) for modeling the temporal evolution of an audio signal. They used SVM and DNN as classifiers, and the final results were the ensemble outputs from both classifiers. One highlight of this work is that they used Generative Adversarial Networks (GAN) [22] for generating additional training samples. The second best work [26] utilized different mel-spectrograms as features including left, right, middle and side channels for more spatial information. They also separated the sound into harmonic and percussive parts using the harmonic-percussive sound separation (HPSS) algorithms. Then, the mel-spectrograms of the two parts were also computed. Background subtraction using median filtering [47] was applied for all the extracted mel-spectrograms for better representation of an audio signal. After that, all the mel-spectrograms were fed into different CNNs and then the output probabilities of all CNNs were averaged for final decision. Overall, the submissions and results in DCASE 2017 showed that deep learning and neural networks had climbed to the dominating position in this field of study.

Recently, Google released a large weakly labelled dataset, named as AudioSet [18]. AudioSet is well designed for the task of audio event classification. This dataset consists of around 2 million 10-second videos from YouTube with labels from an

ontology of 527 classes. This ontology has an hierarchy tree structure and covers a wide range of everyday sounds. FreeSound [14] is a dataset following the AudioSet ontology. It aims at providing internet users a platform to enrich and label the mass amount of sounds. Therefore, this thesis uses datasets that follow the AudioSet ontology for experiments.

In conclusion, deep learning has gained its dominating position nowadays and gradually replaced many traditional modules in the task of audio event classification. Therefore, this thesis focused on studying deep learning methods for the task.

### 2.3 Challenges

Deep learning is the current state-of-the-art approach for the task of audio event classification. As seen in many deep learning tasks, the lack of enough clean data is always one major challenge. Performing this task using deep learning methods involves training a model using many soundtracks with correctly labelled tags. The machine will learn from the data and gradually gain the ability of classifying sounds. For the purpose of building a good classifier, the machine needs a large amount of data with certain quality standards. Google AudioSet offers a great ontology to follow, and Freesound provides a platform for internet users to contribute and label sounds with respective labels. However, the existing datasets are highly unbalanced, and some classes only have limited number of training samples. Collecting and labeling audio samples will still take a huge amount of time and effort. In addition, the soundtracks of online resources are gathered from different recording and coding situations. Therefore, they can have different channels, formats, and qualities. This brings about a problem on how to properly use the various types of audio.

Another challenge is dealing with labels having different levels of abstractions. The labels for audio event classification can have different levels of abstractions [38], including objective descriptions that are more specific and global comprehensive descriptions that tend to depict the overall characteristics of an audio signal. It normally takes shorter time to identify objective descriptions, such as dog, gunshot, and laughter. These descriptions tend to have repetitive patterns along the whole segments of audio. On the other hand, global comprehensive descriptions are usually more abstract and take longer time to identify. For example, recognizing the genre of a music piece needs longer analysis time and finding how the music components interleaved together. Even for humans, these descriptions could take layers of considerations to infer, and the results can differ from person to person, because some classes can be highly subjective. Thus, the level of abstractions must be taken into account when deciding classification methods.

AudioSet has given a complete and well defined ontology with an hierarchical tree structure, which gives a challenge when compared to a classification task with a flat ontology. A class with lower hierarchy belongs to its parent class. For example, the classes 'saxophone' and 'flute' both belong to the class 'woodwind instruments'. For an 'acoustics guitar' sound, the labels from higher hierarchy to lower hierarchy are listed as follows: 'music', 'musical instruments', 'plucked string instruments', 'guitar', and finally 'acoustic guitar'. This hierarchical classification problem is different from

flat classification in a way that some prior probability exists from parent nodes. Taking into this connectivity into consideration might improve the performance. Structured probabilistic models are one solution to build models with connectivity relation between nodes. Another problem in the hierarchical classification is deciding the stop levels of classification. Sometimes, a parent class can describe a event better than wrongly identified sibling class. For example, 'acoustics guitar' is better be classified as its parent class 'guitar' rather than sibling class 'electric guitar'.

It will always be a challenge for creating a more advanced algorithm, including engineering features and designing different neural network structures. Considering the spatial and harmonic information of audio is one direction to enrich features. For example, the work [26] utilized spectrograms from four different directions as spatial information, and it also used the spectrograms of both the harmonic parts and the percussive parts of an audio signal. Modeling the time evolution structures and dealing with variable length of audio files are also important when designing an algorithm. Truncating and zero padding is one solution for the variable length problem, but the choice of fixed length must be carefully chosen. For modeling the time evolution structure, RQA parameters [58], gradient histograms of spectrograms [57], and linear regression coefficients of the local features [17] have been used in previous work. The temporal information can also be modelled in recurrent neural networks and attention structures. More advanced algorithms include modelling the hierarchical connectivity between classes, such as utilizing structured probabilistic models.

Another challenge and future research direction is combining audio features with other multimodal information for multimedia analysis. Multimodal information includes visual images, locations, geometric shapes, temperatures, and humidity values. These multimodal information can be combined with audio information to help build a better machine listening system [2]. In video content analysis, audio features and visual features can support each other and be combined together for providing better audio descriptions. The produced description can be used for storytelling, archiving and hyper-linking.

## 2.4 Development of Deep Learning

As part of machine learning, deep learning has recently achieved great success in many fields, such as computer vision, speech recognition, and natural language processing. However, deep learning was not always that popular and even suffered several huge drawbacks along the history. This subsection will discuss the history and development of deep learning.

The birth of neural nets dates back to 1958, when psychologist Rosenblatt developed the concept of perceptron [59]. Rosenblatt's presented perceptron model follows the working principles of neurons in our brains, and it showed great resemblance to linear regression. The perceptron takes from several binary inputs, each of which is multiplied by its respective weight. Based on the sum of all these multiplications and a threshold value, the perceptron would output 1 if the sum is greater than the threshold and output 0 otherwise. Figure 1 shows this working process of a

perceptron. The blue cells represent the nearby neurons that connected the red one. The weights represent the synapse strength between neurons. The hard thresholding also follows the working principles of neurons, because neurons either fire information or not to nearby neurons. This perceptron concept was inspired from previous research by McCulloch and Walter Pitts [42]. Their model could also output 1 or 0, based on the weighted sum of the binary inputs. Even though the concept of this model is simple, it is still quite innovative at that time. This mechanism could model OR/AND/NOT functions, which were seen as a big problem in AI research fields. Inspired by a neuropsychology research Donald Hebb, Rosenblatt's work surpassed in a way that he managed to let perceptron units learn to adjust weights according to outputs. His idea is to increase the weights of a perceptron when output is smaller than the threshold value and decrease the weights of a perceptron when output is larger than the threshold. Later, Rosenblatt utilized his perceptron concept into a hardware setting that could perform classifications of simple geometric shapes.

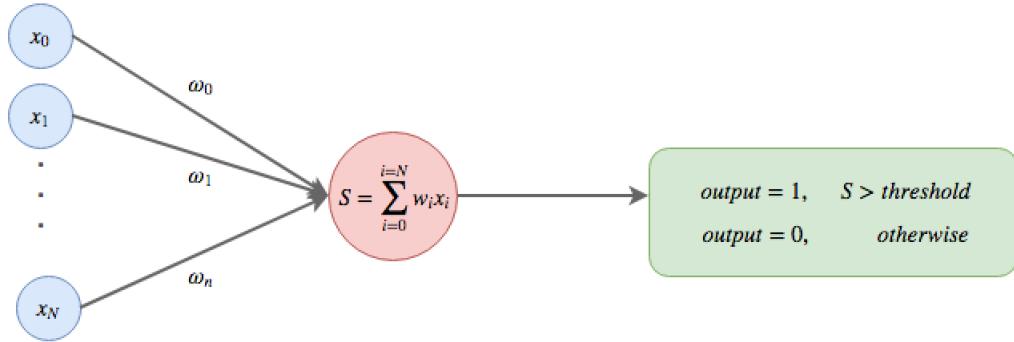


Figure 1: A diagram showing how the perceptron works.

The activation function in Rosenblatt's work is hard thresholding, which was later found not strictly necessary. A study found out that the learning mechanism of a perceptron can greatly improved by adjusting the weights of the perceptron using the derivatives of output errors. However, the concept of perceptron was not accepted well, and some researchers had published their skepticism in a seminal book [45] to argue the limitations of perceptrons. They concluded that the perceptron concept is a dead end for research due to the disability that perceptron could not model non-linear functions, such as XOR function (outputs true only when inputs differ). They did state the XOR function can be modeled in multilayer neural nets, but Rosenblatt's proposed model did not work because it could not adjust the weights before the final layer. This publication was thought as the opening of the first AI Winter, where there was no sufficient funding and publications. The two big winters approximately happened in 1974–1980 and 1987–1993.

The emergence of backpropagation put an end to this AI Winter. Backpropagation uses a differentiable activation function on the output of a perceptron. The activation function can connect layers in a way that make the error from the output layer backpropagate to previous hidden layers (layers between input and output layers), thus making neurons adjust their weights accordingly. Even backpropagation was already derived in the early 60s, its usability in neural nets was firstly analyzed

in the PhD thesis [69] of Werbos in 1974. Due to the effect of the AI Winter, training neural nets by backpropagating error only came to its popularity in 1986, when backpropagation is more precisely and clearly presented in the publication [60] of David Rumelhart, Geoffrey Hinton, and Ronald Williams. In the same year, these three authors published another work [60] stating the learning mechanism of multilayer neural nets in more details. This publication made the backpropagation concept widely known in the AI community.

Neural nets at that time were not only incorporated in supervised learning tasks, but also in unsupervised learning tasks. One example of unsupervised learning is learning the compression of original data and reconstruction from compressed data. Figure 2 shows the working process of an autoencoder. The output of the autoencoder is expected to have same number of neurons as the input, and the red layer contains the compressed information. The connection between the input and hidden layer can be seen as the encoding phase and the connection between hidden layer and the output can be seen as the decoding phase.

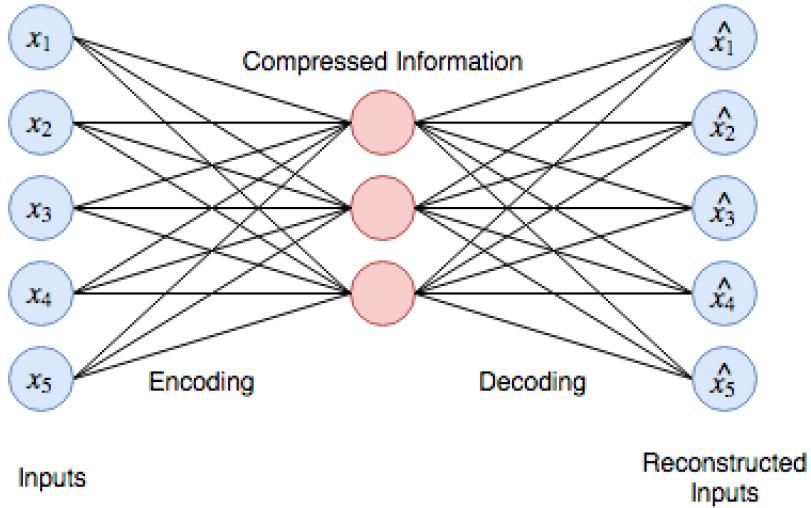


Figure 2: An autoencoder neural net.

In late 90s, Boltzman machines and deep belief networks were other two studies related to unsupervised learning. Boltzman machines are energy based networks that are similar to neural nets. However, the neurons in Boltzman machines connect to every other neurons. Therefore, Boltzman machines can form more complex structures than simple feed-forward structures, and they can also be seen as graphical models. Figure 3 illustrates an example of a structure of Boltzman machines. The structure has a energy function that defines the relations of these neurons. The unsupervised training of this structure aims at reducing the energy function. However, the training speed can be relatively slow. Luckily, a similar net called belief net was introduced by Neal in 1992 [50]. It could be seen as another version of Boltzman Machines with directed and feed-forward structures. These new structures can boost the training speed of these networks. Overall, all these mentioned structures for unsupervised learning aim at maximizing the probability of matching hidden and

visible units. In 1989, a real world application on handwritten digit recognition

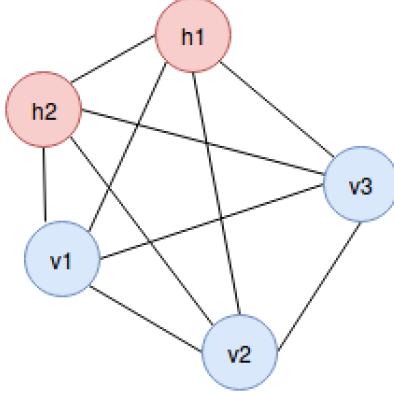


Figure 3: An example Boltzmann machine.  $h_1-h_2$  are hidden units, and  $v_1-v_3$  are visible units. It does not have the feedforward structure but form a complex graph.

was presented [36]. The dataset of this recognition task was collected by US Postal service. This application showed the great capability of backpropagation and revealed the curtain of modern deep learning. Instead of using each neuron for each pixel, a weight sharing technique was used to learn local features. This technique is widely used nowadays in current convolutional neural networks. Sub-sampling is another characteristic technique developed at that time. It is now called pooling. The pooling process aims at reducing dimensions of previous layer to aggregate multiple low-level features and reducing the number of training parameters.

Even though neural nets had shown their great usability in handwritten digit recognition, it was still quite challenging in another important task of recognizing human speech. Speech recognition is more complex than recognizing the visual images of handwritten digits. Because the input of speech recognition task is a stream of sequences which has complicated internal relations, and this recognition needs to memorize previous information. Audio event classification and other stream processing tasks also faced the same challenge in dealing with long sequences of data. In 1989, an approach called time-delay neural networks(TDNN) was brought up for phoneme recognition [68]. Its structure is similar to normal neural nets with the addition of multiple models for input with different delays. This structure is similar to moving a processing window through the speech input from its beginning to end. TDNN also shares the same weights sharing technique with CNN. Each neuron from TDNN reacts to a sub-region area from previous layers rather than all neuron units in previous layers. However, CNN have been widely used in image processing and computer vision. Because of the limitation of backpropagation, another AI Winter arrived in the middle 90s. The main challenges of backpropagation were vanishing and exploding gradients resulted from multiple multiplications after many feedforward layers. The backpropagated error can either became extremely large or small, thus failing the model to learn efficiently. In addition, applying backpropagation to learn long time information was thought difficult[29] [3]. Even though Long Short Term Memory (LSTM) was already presented later in 1997, it still made no difference

for the development of deep learning because of many other limitations, such as slow functioning computers, limited algorithms, and people's lack of interest. Then, support vector machines (SVMs) and random forests became more popular because of their nice mathematical theories and relatively good performance.

Research on neural nets did not stop entirely in this AI Winter. Some researchers came up with a new name 'deep learning' for neural nets as the new branding technique. Later a study in 2006 ushered the research on neural nets into a new era [27]. This study stated that some weights initialization techniques that have the possibility of avoiding the problems of vanishing and exploding gradients. The initializing weights were firstly defined by training the nets under an unsupervised manner. The pre-trained weights then became good starting point for the mode to learn. In this method, not only weights are initialized in a more optimal approach, high level representations of the data are also learned. This initialization approach solved the dilemma that manually designed features failed to work well. Furthermore, the speech recognition field has another breakthrough study [46], where deep belief neural nets broke a decade-old record on a dataset.

The increasing computation power of computers and the usage of GPU are also major improvements in the development of deep learning. More advanced computers and GPUs allow computers calculate the weights in a faster speed. Furthermore, using GPUs for parallel computing makes training with bigger datasets and deeper networks possible. A study [56] in 2010 stated that using GPUs is approximately 70 times faster than using CPUs. During this time period, another improvement is the increased understandings and modifications of activation functions. Rectified linear units were discussed and seen as the best activation function [31] [49] [20] because of its simplicity with the function of  $f(x) = \max(0, x)$ . This simplicity allows faster learning than sigmoid and tanh, although having hard non-linearity at 0. Rectified linear units were proved to be able to achieve same or better performance without unsupervised pre-training for large labelled datasets [20], which is a milestone in training deep neural networks. Dropout, as another key concept of deep learning, was invented by Geoffrey Hinton *et al.* in 2012 [28] to prevent the co-adaptation of neuron units, thus avoiding the overfitting problem. The authors also submitted an entry [35] to the ILSVRC-2012 computer vision challenge and ranked in first place with an error rate of 15.3%, which was far better than the second place 26.2%. They incorporated a CNN structure, ReLU units, GPU computation, and the dropout concept. After that, CNN and deep learning finally started to gain enough attention and become popular. This was thought the climax in the development of deep learning. Deep learning then has been studied in many other fields, and various well-designed structures and advanced algorithms have been proposed. LSTM also made a huge comeback from 2009 to 2014 in many fields, including natural language text compression, handwriting recognition [23], and speech recognition [24]. However, longer traversing along cells in LSTM might cause vanishing gradient problem, and the LSTM structure could take many resources which make the current need of real-time processing hard to achieve. In recent years, the advent of attention [72] mechanism in 2015 caused gradually increased replacement of RNNs and LSTMs because of its simplicity and need for low resources. These replacement with attention-based models were already

seen in many huge companies, such as Facebook [16] and Google [66].

In conclusion, deep learning has become one of the most useful and popular research topic with the help from many aspects, such as the big amount of nicely defined large datasets, occurrence of faster computers, well-designed deep learning structures for different applications, optimal initializing techniques, non-linear activation functions, and more advanced optimization and generalization techniques. Therefore, this thesis focuses on studying deep learning methods for the task of audio event classification.

### 3 Research Theory and Methods

This section focuses on explaining the research theory and methods for our research tasks. The basics of machine learning is firstly introduced, including the relationship between deep learning and artificial intelligence, definitions of several technical terms, and the common work-flow for an AI task. Regarding of deep learning, several important concepts are introduced, such as activation function, loss function, backpropagation, optimization, regularization, and structures of different models. In addition, this section discusses the procedure of computing log-mel spectrograms.

#### 3.1 Machine Learning Basics

Machine learning is an approach to build AI systems that can perform tasks for humans, such as understanding language, recognizing visual and sound objects, playing chess, and even driving the cars. It uses data as resources and let computers gradually improve the performance on certain tasks without being explicitly programmed [61]. Deep learning is an approach belongs to machine learning, other common approaches include Bayesian networks, support vector machines, decision tree learning, and hidden Markov models. Understanding the basic principles of machine learning would help understand deep learning, since deep learning has been strongly shaped and influenced by some traditional machine learning methods.

Tom M. Mitchell provided a definition of machine learning. It indicates that a computer program is expected to complete some tasks  $T$  and then learn from the experience  $E$  when performing the tasks. 'Learning' means the evaluation of the performance  $P$  would improves as experience  $E$  progress.

The tasks  $T$  include common AI tasks, such as classification, regression, machine translation, speech recognition, natural language processing, speech or image synthesis, and denoising. In this thesis, the task is audio event classification involving specifying the categories of an audio signal. One task in this thesis is to infer one label from 41 classes for an audio signal, while the other implementation is to infer one or multiple labels from 527 classes for an audio signal. The former task is called multi-class classification, where a machine needs to classify a sample into only one class from many options. The latter task is called multi-label task where a machine needs to infer one or more classes from multiple selections.

As for performance  $P$ , every task needs a quantitative measure for evaluating the learning algorithms. The evaluation allows machine find the best model with great generalization ability. For classification tasks, the models often use accuracy as performance. Accuracy indicates the proportion of correctly predicted samples. Error rate is another equivalent for accuracy. But for some cases, accuracy might not be informative at all. In cancer prediction, where only a few samples are positive, the model can achieve very high accuracy if it predicts all samples to be negative. Other measures, such as precision and recall, could be better choice in that case. For certain tasks, such as machine translation, the evaluation might be a little complicated. Since a single input sentence might have multiple suitable translations with different length, special evaluation scores need to be designed. Two commonly used metric

for evaluating machine translation are BLEU (bilingual evaluation understudy) and METEOR (Metric for Evaluation of Translation with Explicit Ordering). In conclusion, the evaluation score for performance  $P$  should be application dependent, and sometimes a high score does not always indicate the performance of a model is good.

Experience  $E$  simply means experiencing a given dataset during the learning phase. The dataset is a collection of examples, each of which contains one or multiple features. For instance, the collection of heights and weights for students in a primary school is a dataset. Each student represents a training example, which contains two features height and weight. The experience can be roughly divided into two main categories, supervised learning and unsupervised learning. For supervised learning, each example is associated with a label or a target result. In our primary school example, the label is the age of a student. So a machine will experience this labelled dataset and learn to infer the age of a primary school student given the height and weight. As for unsupervised learning, the machine is expected to learn some characteristics only from the dataset. An example demonstrates unsupervised learning would be clustering students into several groups, where the students have similar heights and weights.

Based on the explanations of  $T$ ,  $P$ ,  $E$ . The common work-flow of a machine learning task is demonstrated as follows. For example, the dataset is a collection of heights and weights from 2000 students in a primary school. The task  $T$  is to infer the age of a student from his height and weight. Since this is a supervised learning task, the ages of the students are also given. First of all, features need to be extracted from the dataset. In this case, the features can be represented as a matrix  $X \in \mathbb{R}^{2000 \times 2}$ , where 2000 represents the number of students and 2 represent the two features, height and weight, for each student.  $x_i$  is a 2-dimensional vector represents the height and weight of the  $i$ th student. The labels can be represented as a vector  $y \in \mathbb{R}^{2000 \times 1}$ , where  $y_i$  is the age of the  $i$ th student. The task then can be described as finding a model  $f$  which can map the height and weight of a student to his age:  $f(x_i) = y_i$ . In order to evaluate the performance of the model, some students need to be selected as testing data. The students should be chose randomly so that they can represent the general distribution of this dataset. The split ratio between training and testing data may vary with different cases, a common reasonable choice is 4:1. With that ratio, the model is trained with 1600 examples and evaluated using the rest 400 examples. The evaluation of performance  $P$  on the testing data can be accuracy in this case, which is the ratio between the correctly predicted number and the number of all students in the testing set. Then the training data along with the labels are fed into a machine learning model to learn. Designing a good model might be the most important part in the work-flow. A well-defined model will gradually reduce training error as the training proceeds. However, small training error does not always mean the model is good enough. A good model should have great generalization ability. Generalization means it should also have good performance on general data other than the training data. The testing error is a good indicator on how well the model generalizes. As the training starts, both the training and testing error would start to decrease. At a certain point, when the model starts to memorize the original

training data, the testing error would start to increase. Thus, stopping the training at the optimal point would give a relatively good model. If both the training error and the testing error are high, then the model is suffering from underfitting. If the training error is quite small and the testing error starts to increase, then the model is overfitted. These overfitting and underfitting problems can also happen with different model capacity or complexity.

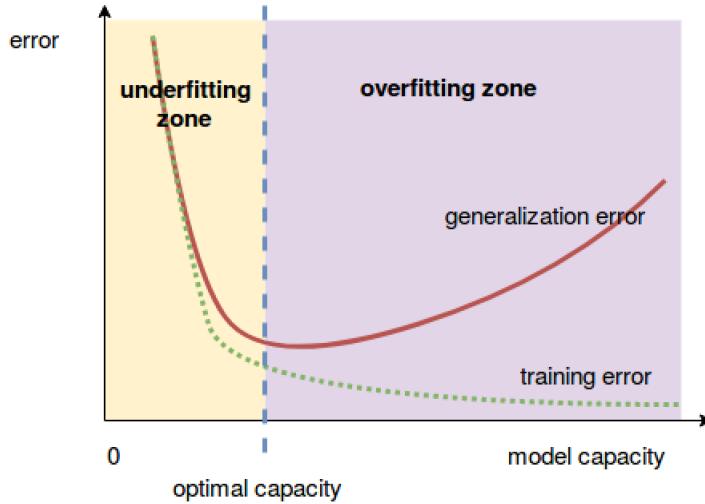


Figure 4: Typical relationship between capacity and error.

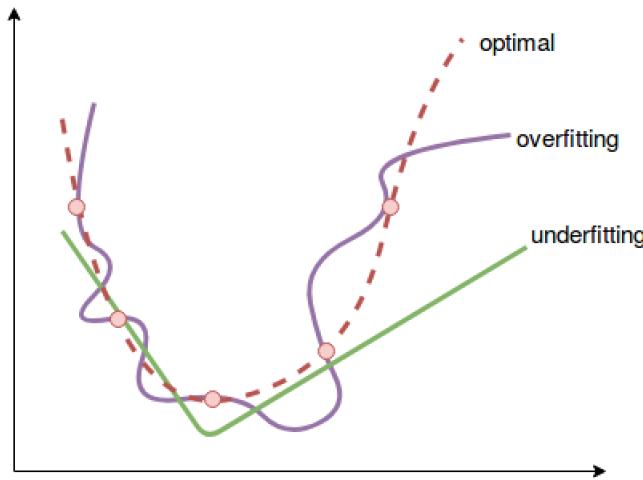


Figure 5: Comparison between optimal modeling, overfitting and underfitting. Red circled dots are the dataset that need to be modelled.

As can be seen from Figure 4 and 5, small capacity can lead to underfitting problem. because the model is too simple. On the other hand, if the model is too complex, the model then can only model the training data well but with a relatively high generalization error. Once we get a model with great generalization ability after carefully designing the model and fine-tuning parameters, the model then can be

put into use. In our example case, the fully-trained model can predict the age of a student based on his or her weight and height. This concludes a simple example of the machine learning work-flow.

## 3.2 The Theory of Deep Learning

With the introduction of some machine learning basics, this subsection will focus more on the theory of deep learning that is used in later experiments. The basic building blocks, feedforward neural networks, are described firstly. Then this subsection describes some important concepts, including loss function, backpropagation, mini-batch gradient decent, optimization, and regularization. Furthermore, this thesis explained some typical neural network structures, such as CNN, RNN, and multi-level attention models.

### 3.2.1 Feedforward Neural Network

Feedforward neural networks, or multilayer perceptrons (MLPs), are the quintessential models of deep learning [21]. As mentioned previously, the age-prediction task is trying to find a mapping  $f$  that defines the relationship between input  $x$  and output  $y$  the most:  $y = f(x)$ . A feedforward neural network then defines a mapping  $y = f(x; \theta)$  using parameters  $\theta$ . The model will gradually learn the value of  $\theta$  during the training process and finally provide an optimal approximation [21]. This network is the main building block of deep learning, and it appears in almost every neural network structures. Following the knowledge from neural science, the feedforward neural networks are composed of layers of neurons. The input flows through the layers in a feedforward manner. Each neuron, or perceptron, will learn its own contribution to the input while feeding the model with more training examples. Feedforward means the input only flows at one direction. Recurrent neural networks have feedback connections to the input, while convolutional neural networks are special kind of feedforward neural networks with weight sharing and pooling mechanism. Figure 6

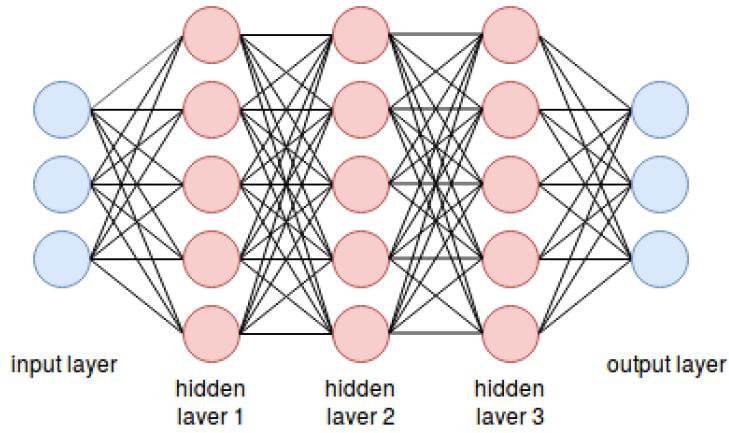


Figure 6: 3-layer feedforward neural networks

represents a 3-layer feedforward neural network. The blue cells of input layer accepts

input features, and the red cells of the middle layers would learn how units from previous layers contribute to their activation computation. Figure 7 explains in detail

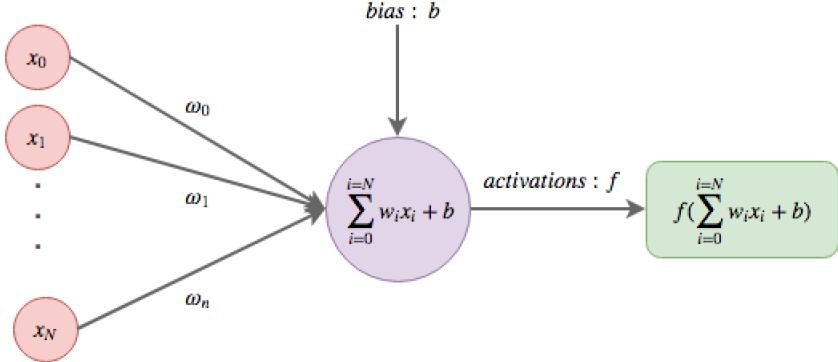


Figure 7: A graph showing how a neuron works.

the computation procedure of a neuron. Each neuron has the same number of weights as the number of neurons in previous layer. Each weight  $w_i$  controls the contribution of the  $i$ th neuron in previous layer. The sum of all weighted contributions with an additional bias  $b$  would complete the computation. This feedforward neural network is expected to learn something new in each layer. The results and error can be obtained from the output layer. In a classification task, each neuron in the output layer has a value representing the occurrence probability of its corresponding class.

### 3.2.2 Activation Function

Even though the birth of neural nets dates back to decades ago, they were thought useless for a long time. Because neural nets back then were unable to model non-linearity. Since almost all the real-life tasks are somewhat complex, the ability to model non-linearity is heavily needed. Differentiable non-linear activation functions are one solution for this problem. The differentiability allows neurons change their weights mathematically according to the output results. Sigmoid, tanh, ReLu, and softmax are the most common used differentiable activation functions that can model non-linearity. Activation functions are also decision functions with their values represent if a neuron is 'fired' or not. However, the sum of a neuron can range from minus infinity to plus infinity. Thus, compression is needed to avoid the overflow of neurons. Step function is ideal but not differentiable. Therefore, smoother versions are ideal options because their adaptation ability to gradient based optimization algorithms.

Figure 8 shows the plots of three common activation functions. Sigmoid activation follows the Equation 1, which compress a value to  $(0, 1)$ . Not only non-linearity is added, the activation value is also a good indicator representing the activated percentage of a neuron. For example, the output of 0.7 means a neuron is 70% activated. Equation 2 shows the equation of tanh activation. It has similar shape as sigmoid, but it maps the value to  $(-1, 1)$  and has zero-centered characteristic. ReLU is the default recommendation for modern neural networks [21]. Even though it is not

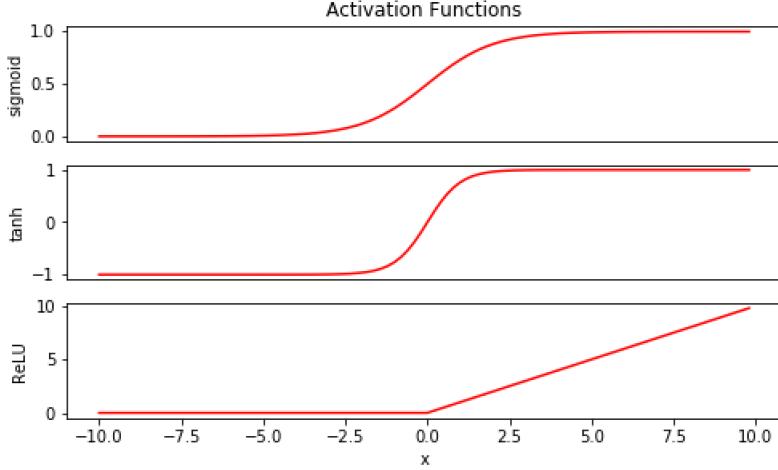


Figure 8: Plots for sigmoid, tanh and ReLU.

differentiable at 0, it still has been proven empirically useful. The equation 3 shows the simplicity of ReLU, thus causing less computation load. In addition, gradient vanishing problem could be overcame using ReLU because the gradient would always be 1. As for the sigmoid and tanh, they both suffer the gradient vanishing problem, thus stopping the further learning of a model.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2)$$

$$ReLU(x) = \max(0, x) \quad (3)$$

The classification tasks of this thesis will use both sigmoid and softmax activation functions. Sigmoid is used in the multi-label classification task, where each neuron in the output layer can have a probability ranging from 0 to 1. As for the softmax, it would normalize all the scores in the output layer to make sum of all probabilities exactly equal to 1. Softmax is used in the multi-class classification task that only has one correct label for each training example. The normalizing process can be seen in Equation 4 and Figure 9.

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad (4)$$

### 3.2.3 Loss Functions and Backpropagation

After building brain-like neural network structures and the activation mechanism, next step is defining how a model will learn from this settings. For a supervised learning task, its loss function represents error between predicted and correct results. Each time when a model is fed into some training examples, the model is expected to modifying its weights and biases accordingly to more closely fit the training data.

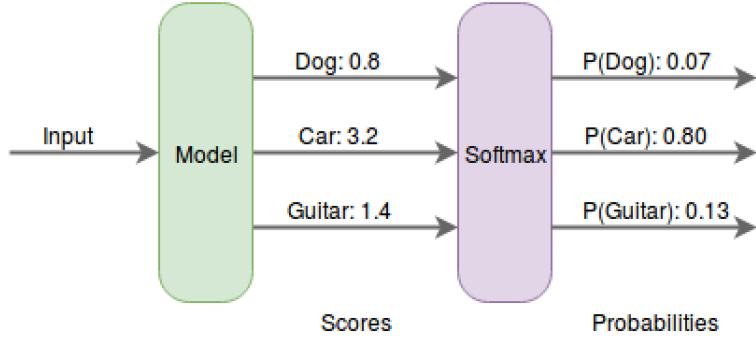


Figure 9: Explanation on softmax activation. After a softmax layer, the input probabilities are normalized so that their sum equals to 1.

Thus, the learning of a model refers to gradually making small changes when fed into training examples continuously. This learning process requires the loss function to be a smooth function so that gradient based algorithms can be used. The mean squared error is a common error function for regression tasks. It is defined in Equation 5.

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N (f(x_i; \theta) - y_i)^2 \quad (5)$$

The  $\theta$  represents the weights and biases of a model.  $f(x_i; \theta)$  refers to predicted results and  $y_i$  represents the correct results. The goal is minimizing the loss function  $J(\theta)$ . As shown in Figure 10, the gradient value refers to the derivative of that point. If

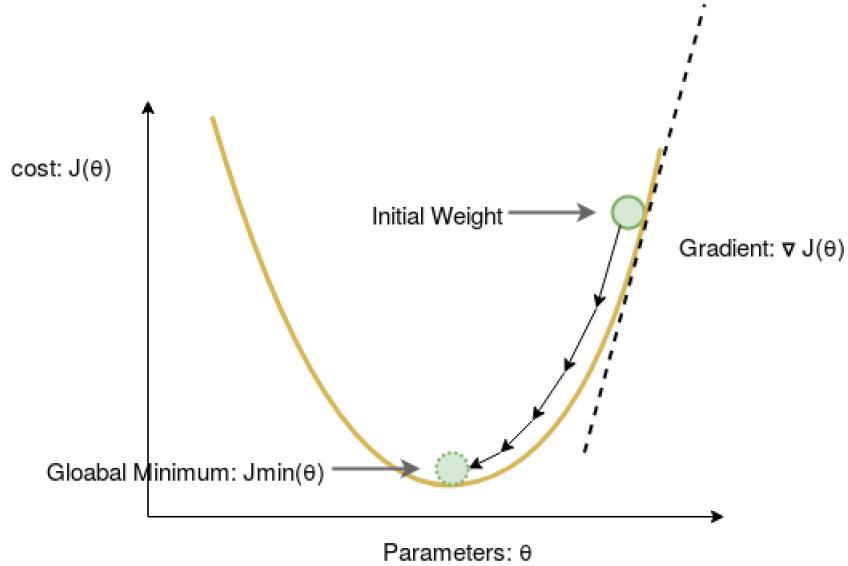


Figure 10: Gradient decent visualization.

the weights are adjusted according to the direction of this gradient, the loss function would be gradually minimized and reach the global minimum.

$$\theta = \theta - \alpha \frac{\partial}{\partial \theta} J(\theta) \quad (6)$$

Equation 6 shows the iterative updating process of these weights.  $\alpha$  is called the learning rate, which represents the moving speed of error function flowing to its minimum.

According to the chain rule, error can propagate back to all the weights of previous layers. For the example shown in Figure 11, Equation 7 shows the calculation of  $\frac{\partial z}{\partial w}$

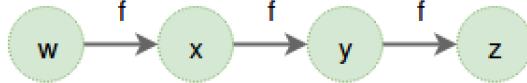


Figure 11: A graph with chain connections:  $x = f(w)$ ,  $y = f(x)$ ,  $z = f(y)$ .

to update  $w$  using the chain rule. This back-propagation algorithm make all weights and biases be able to learn when error flows back to the input. This thesis uses binary crossentropy as the loss function for the multi-label classification task and categorical crossentropy as the loss function for the multi-class classification task.

$$\begin{aligned}
 \frac{\partial z}{\partial w} &= \frac{\partial z}{\partial x} \frac{\partial x}{\partial w} \\
 &= \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \frac{\partial x}{\partial w} \\
 &= f'(y) f'(x) f'(w) \\
 &= f'(f(f(w))) f'(f(w)) f'(w)
 \end{aligned} \tag{7}$$

### 3.2.4 Regularization

As mentioned in Section 3.1, a model can suffer from overfitting when the model starts to memorize the limited training data. Memorizing training data make a model failed to learn new characteristics. A good model should also have a small testing error. Regularization refers to the strategies that are explicitly designed for reducing testing error, possibly at the expense of increased training error [21]. This subsection describes some common regularization techniques, including early stopping, data augmentation,  $L^2$  and  $L^1$  regularization, and dropout.

Since overfitting means a model performs bad on unseen data, we can separate a part of training data as validation set. The validation set is not used in training process but used for evaluating the generalization ability of the model. As the training proceeds, validation error would decrease first and then increases when the model starts to over-learn and failed to generalize on unseen data.

Early stopping means stopping training at the turning point when validation error starts to increase, thus making a model have optimal generalization ability. Figure 12 illustrates the concept of early stopping. However, these two curves only represents the overall tendency and error can alternate quickly in actual training process. Because of the fast alternation, the first turning point when validation error starts to increase could be a wrong step to stop, a few more steps need to be

<sup>0</sup><https://www.slideshare.net/xavigiro/methodology-dlai-d612-2017-upc-deep-learning-for-artificial-intelligence>

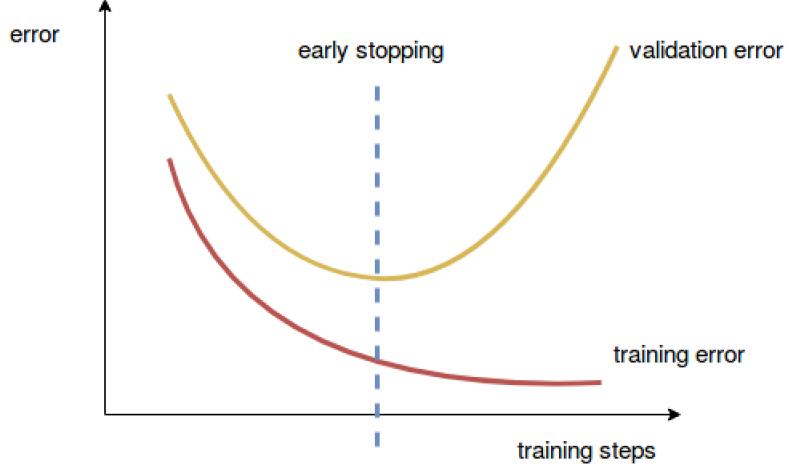


Figure 12: Ideal training and validation error curve. The vertical line represents the early stopping point.

monitored. If the validation error does not decrease after these steps, then training can be early stopped. The number of the steps is called patience.

Data augmentation is the simplest regularization method to avoid overfitting by adding more training examples. However, it is too laborious and expensive when collecting and labeling large amount of new training examples. Other alternatives for augmentation are needed. Augmentation for images include shifting, re-scaling, and shearing. In audio domain, soundtracks can be augmented by pitch-shifting and time stretching for a classification task. Building a generative model is a more advanced augmentation technique. Generative models are mostly trained in an unsupervised manner. They aim at generating new data that is similar to original data but not exactly the same. One generative model for generating image is created by Radford *et al.* in 2015 [55]. This model draws 100 random numbers from a uniform distribution and then learns to generate colorful images from that. The 100-dimensional input and the 3 channels  $64 \times 64$  image can be seen at the leftmost and rightmost parts respectively in Figure 13. Thus, this model can learn the generating mechanism of images instead of memorizing them. Generative Adversarial Network(GAN) is another approach for generating images by utilizing a discriminator network after a generator network [22]. This would result in a two-player game, where the generator will generate fake images to fool the discriminator and the discriminator will also learn to detect fake images at each round. Ideally, the generator will generate indistinguishable images from the real ones as the battle continues. GAN has also been used in audio domain to synthesize intelligible words and other sound events, including bird vocalization, drums, and piano [10]. In addition to GAN, WaveNet is another neural network for generating raw audios. WaveNet has the state-of-the-art performance on text-to-speech and can generate highly realistic musical fragments [65].

$L^2$  and  $L^1$  are the most common regularization techniques focusing on penalizing weights by adding a special regularization term  $\Omega(\theta)$  into the original lost function

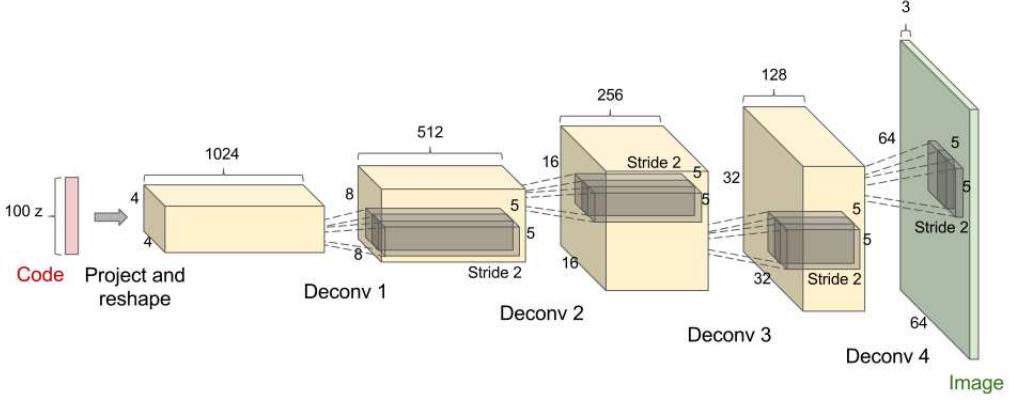


Figure 13: DCGAN network structures [55]. In this model, fully connected layers and deconvolution layers are used. The deconvolution layer is just the reverse of a convolution layer which will be discussed in later section.

$J(\theta)$ . The modified loss function  $\tilde{J}(\theta)$  then becomes:

$$\tilde{J}(\theta) = J(\theta) + \alpha\Omega(\theta) \quad (8)$$

The  $\alpha \in [0, \infty]$  is the hyperparameter that controls the contribution of the penalty.  $\alpha = 0$  means no regularization, and larger  $\alpha$  means more regularization [21]. For  $L^2$  regularization, the regularization term is defined as:  $\Omega(\theta) = \frac{1}{2} \|\theta\|_2^2$ . Equation 9 and 10 show how new weights are updated. Parameter  $\epsilon$  is the learning rate, and the weight decay term has modified the learning rule to multiplicatively shrink weight vectors by a constant factor  $\epsilon\alpha$  at each step [21]. Thus,  $L^2$  regularization is also called weight decay.

$$\begin{aligned} \nabla \tilde{J}(w) &= \nabla(J(w) + \frac{\alpha}{2} w^T w) \\ &= \nabla J(w) + \alpha w \end{aligned} \quad (9)$$

$$\begin{aligned} w &\leftarrow w - \epsilon \nabla \tilde{J}(w) \\ w &\leftarrow w - \epsilon \nabla J(w) - \epsilon \alpha w \end{aligned} \quad (10)$$

For  $L^1$  regularization, the regularization penalty term is defined as:  $\Omega(\theta) = \|\theta\|_1 = \sum_i |\theta_i|$ . From Equation 11 and 12, it has been shown that  $L^1$  regularization reduces constant value from the weight. Some weights can reduce to 0 with  $L^1$  regularization, thus having in a sparse model. Since  $L^2$  regularization reduces weights in a constant ratio manner, the weights will progressively move to zero with decreasing steps.

$$\begin{aligned} \nabla \tilde{J}(w) &= \nabla(J(w) + \alpha \|w\|_1) \\ &= \nabla J(w) + \alpha \text{sign}(w) \end{aligned} \quad (11)$$

$$\begin{aligned} w &\leftarrow w - \epsilon \nabla \tilde{J}(w) \\ w &\leftarrow w - \epsilon \nabla J(w) - \epsilon \alpha \text{sign}(w) \end{aligned} \quad (12)$$

Overall,  $L^1$  regularization is less sensitive to outliers dataset than  $L^2$  due to its sparsity. And  $L^2$  is more computationally efficient and stable, since it has a closed form symmetric solution while  $L^1$  is non-differentiable at the edges. Figure 3.2.4 shows the edges of  $L^1$  and the smoothness of  $L^2$  regularization in 3-dimensional plots. In conclusion,  $L^1$  is a better choice when needing a compressed model or selecting features, and  $L^2$  regularization is preferred otherwise.

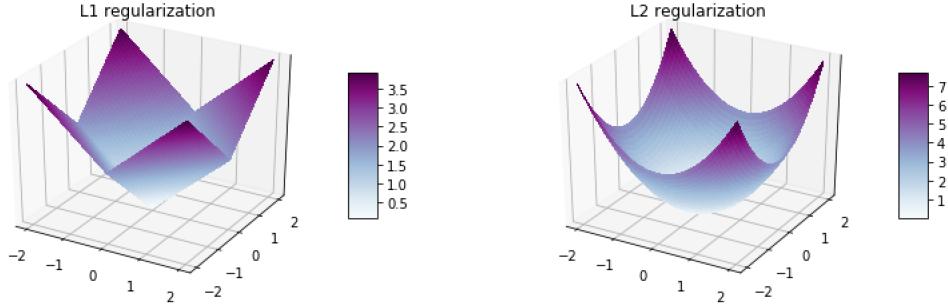


Figure 14: Comparison between  $L^1$  and  $L^2$  regularization.

Dropout was firstly introduced in 2014 by Srivastava *et al.* [62] and has become one of the most popular regularization techniques. The main idea is randomly deactivating some neurons in a layer with all their connections during training process. Dropout ratio, a number between 0 and 1, is used to represent the percentage of deactivated units. This process can reduce complexity of neural networks and avoid co-adapting between neurons. The deactivating process can be simply done effectively by setting the value of the non-activated neurons to zero. However, since dropout can reduce model capacity, the original model should have a relatively high complexity. Figure 15 shows the usage of dropout. Only the 2nd and 3rd neurons are activated in the left graph, and only the 1st and 3rd neurons are activated in the right graph. In every iteration of training, dropout neurons are selected randomly to form a sub-network. All those sub-networks of the whole model are then combined together for evaluation. Thus, dropout can also be seen as an approximated bagging method by averaging over multiple stochastic decisions [21], which is effective to avoid overfitting.

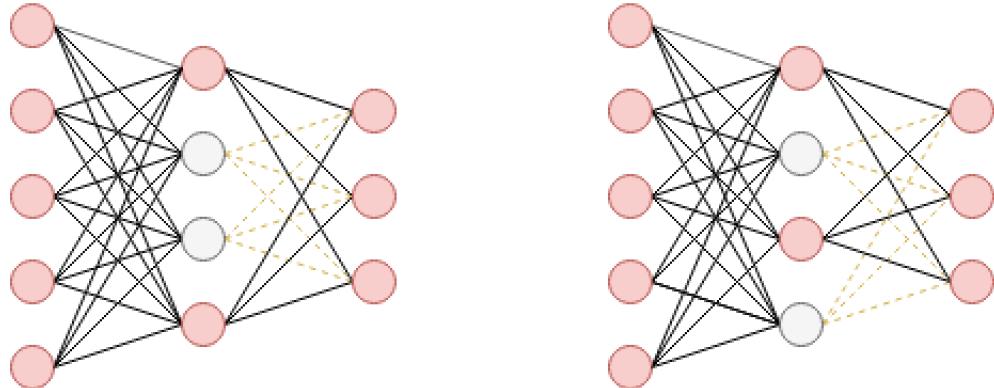


Figure 15: Examples of dropout configurations.

### 3.2.5 Optimization

The optimization stated here focuses on finding weights and biases that can minimize the loss function the most. The challenges during optimization process are firstly introduced, such as the saddle points and local minima. Several gradient decent variants are also explained and compared, including batch gradient decent, stochastic gradient decent, and mini-batch gradient decent. In addition, this part discusses some gradient descent optimization and learning rate adaptation techniques.

The contour of a 1-dimensional loss function is already shown in figure 10. However, the weight could have tens of thousands dimensions, thus causing an extremely complicated loss function. One challenge is that the local minima also has a gradient at zero as the global minima. If weights are initialized at a point close to local minimum, the loss of a model can easily converge into this minima point, thus arriving at the wrong destination. Saddle point has a local minima in one direction and a local maxima in the other direction. This structure could make gradients oscillate along the local-minima direction. Even though gradients are fortunate enough to land in the point of global minima, it might also oscillates in the valley because of a large learning rate. Thus, the goal of optimization is to quickly reach the global minima without being trapped in local minima and saddle points.

Applying gradient decent after different number of training examples results in 3 gradient decent variants, which are batch, stochastic, and mini-batch gradient decent. Batch gradient decent only updates a model using all training examples. This variant has fewer updates, thus it is more computational efficient but might require big memory for large dataset. Since it is based on the whole training dataset, the error gradient would flow in a more stable manner. This stability might also make gradient trapped in local minima or saddle point. On the contrary, stochastic gradient descent updates a model for one random training example. This frequent updates behaviour makes it possible for online monitoring, even though it is more computationally heavy. Updating a model only on one example also means adding random noise on gradient, thus making the error jump around and avoiding local minima. Mini-batch gradient descent is a combination between the previous two. It randomly select certain number of training examples for training and then apply gradient descent. The selected number is called batch size. If the batch size equals to 1, then it will become stochastic gradient descent. If the batch size equals to the size of the training data, then it will be the same as batch gradient descent. Thus, the ability of adjusting batch size provide more freedom for users. Smaller size would resemble stochastic gradient descent which has faster converging time and noise. Larger size would resemble batch gradient descent which has more accurate estimate but the risk of being trapped at local minima or saddle points. Figure 16 shows the validation loss with different number of batch size. It indicates that smaller batch size can converge faster but can be more noisy. Overall, mini-batch gradient descent is the most recommended option due to its variability. The common choice should be numbers at the power of 2 to fits the memory requirements of the GPU or CPU.

For the optimization of gradient decent algorithms, two main strategies are incorporating the momentum and adjusting learning rates adaptively. The momentum

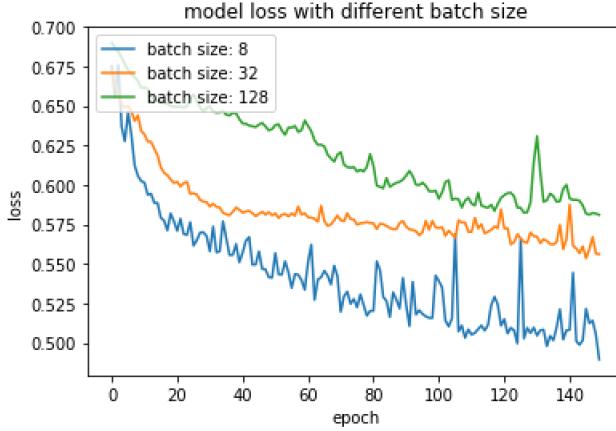


Figure 16: Comparison of the loss updates between different batch size.

concept originates from a physical analogy. The negative gradient is similar to a force dragging a particle through the parameter space [21]. The particle is initialized with velocity 0 at certain point in the space. Then the velocity would increase because of that force. Taking velocity into account means previous velocity has an impact on current location because of momentum. Equation 13 shows the process of previous gradients making contributions with a ratio of  $\alpha \in [0,1)$  to current gradient computation.

$$\begin{aligned} v &\leftarrow \alpha v - \epsilon \nabla_{\theta} J(\theta) \\ \theta &\leftarrow \theta + v \end{aligned} \tag{13}$$

As can be seen in Figure 17, this momentum concept can accelerate the learning progress. The green path with momentum converges faster, and the red dot lines represent the momentum at those points. Nesterov Momentum is a slightly modified

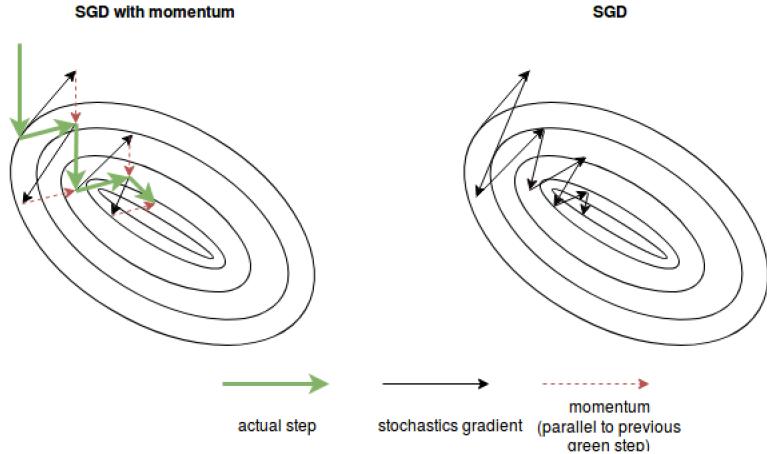


Figure 17: The green paths depicts how the loss converges with the momentum, while the black path represent the noisy stochastic approach.

version of original momentum. The difference can be seen in Figure 18. Compared

from Equation 13 and Equation 14, it can be seen that Nesterov momentum update is based on "lookahead" gradient.

$$\begin{aligned} v &\leftarrow \alpha v - \epsilon \nabla_{\theta} J(\theta + \alpha v) \\ \theta &\leftarrow \theta + v \end{aligned} \quad (14)$$

As for algorithms with adaptive learning rates, AdaGrad, RMSprop and Adam are the most common three. AdaGrad adapts the learning rates for all model parameters with a scaling factor that inversely proportional to the square root of the sum of all historical squared values of the gradient [11]. Since the sum is always increasing, then the learning rate will always be decreasing. Thus, larger partial derivative will decrease the learning rate more. However since this accumulation starts from the beginning, this might reduce the learning rate excessively. Overall, AdaGrad performs well for models with relatively simple loss structures. RMSprop [64] modifies the accumulation of AdaGrad into an exponentially weighted moving average. Therefore, the gradient from far past will not be considered. However, an additional hyperparameter is needed for controlling the window length for averaging. Adam is derived from the name "adaptive moments" and was proposed by Kingma and Ba in 2014 [33]. Being similar to RMSprop, Adam also stores the running average of past squared gradients. The difference between Adam and RMSprop is that it uses a smooth version of the gradient and has bias correction mechanism. Adam also has some extension versions such as AdaMax [33] and Nadam. Overall, RMSprop works slightly better than AdaGrad because its ability to solve the problem of radically decreasing learning rates. Adam adds momentum and bias-correction to RMSprop, thus making it the most preferred algorithm in practical implementation.

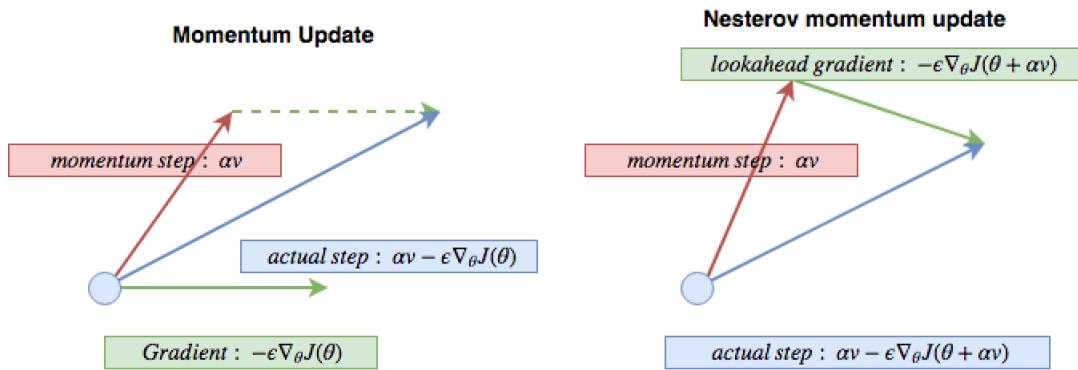


Figure 18: Comparison between two momentum update algorithms.

### 3.2.6 Batch Normalization

Batch normalization is introduced by Ioffe and Szegedy in 2015 [30]. This study shows that batch normalization improves the top result of ImageNet(2014) with a reasonable amount and even exceeded human raters. Currently, batch normalization is used in many neural networks.

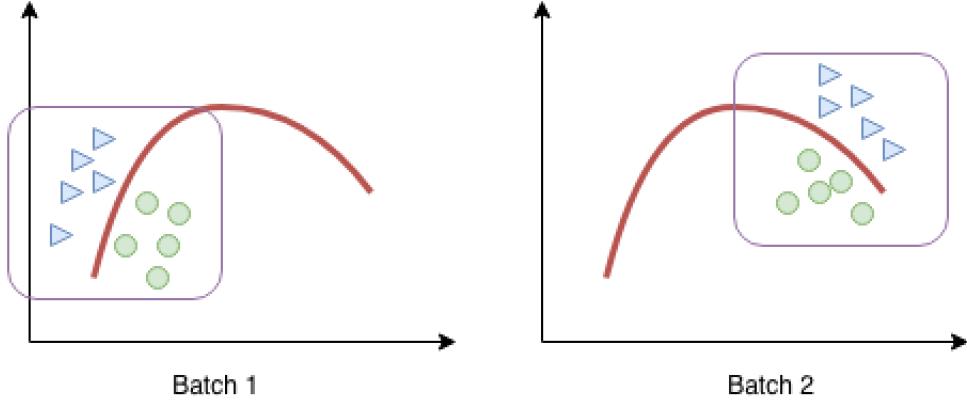


Figure 19: An illustration showing how a model learns different parts of the decision boundary when distribution in each batch differs.

Batch normalization is inspired by the covariate shift problem. An illustration explaining this problem can be seen in Figure 19. When a model is trained only on the left batch, it will only learn certain parts of its decision boundary. When it comes to the other batch, it will learn different parts of the decision boundary. This problem not only exists in the input layer but also in the hidden layers. Solving this requires lower learning rates and careful parameter initialization, thus making training extremely hard [30]. Figure 20 shows the implementation process of batch normalization. The parameters  $\gamma$  and  $\beta$  are two extra parameters defining the mean

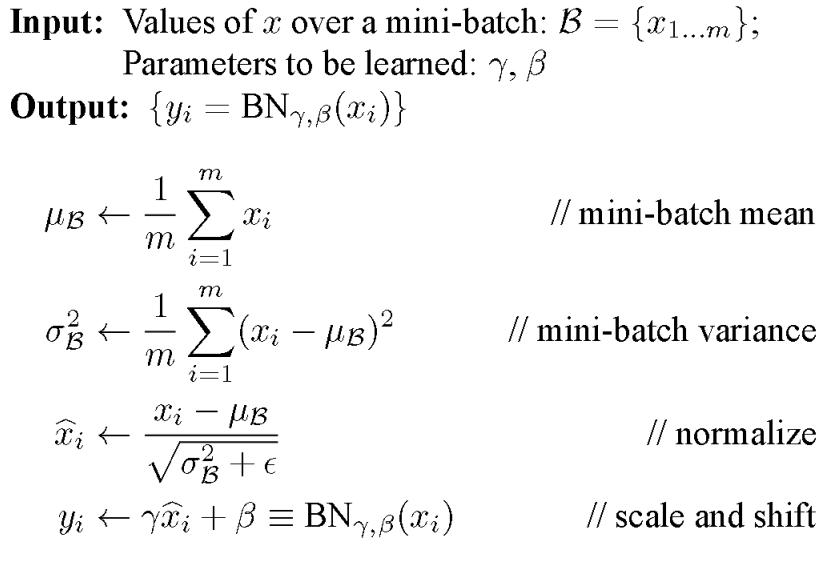


Figure 20: Batch normalization transform with input  $x$  [30].

and variance of a batch. The two parameters can also be learned by a neural network.

As can be seen in Figure 8, the gradient will be 0 if the weights are too large or small. This is highly possible after the multiplications of several layers. The model

will learn nothing with gradient equaling to 0. Using batch normalization before activation functions can avoid the problem of gradient saturation and vanishing. Batch normalization also makes hidden layers independent of each other. Therefore, smaller error in one layer will not be amplified and propagated to other layers, thus making the model more stable and have larger learning rate. Although batch normalization can also add some noise into the layers, this could be useful somehow due to the regularization effect of the noise.

### 3.2.7 Convolutional Neural Networks

Convolutional neural networks (CNNs) generally consists of two main building blocks, convolution layers and pooling layers. The input of a CNN is normally an image. For the classification tasks of this thesis, the spectrograms of audio can also be seen as the input for CNN. The input is then organized as a 3-dimensional box with height, width and depth, where the depth represents the number of channels. For instance, an RGB image has three channels with each channel having a two-dimensional digits array representing the values of respective color.

The most characteristic feature of a CNN is convolution operations, where only part of the neurons in previous layer contributes to the activation of each neuron in the next layer. This is performed by a filter or kernel. The input of a CNN is a image, and a filter normally has a 2-dimensional size that is smaller than the size of the image. Then the filter would move around the whole image using matrix multiplications to get the values for features maps. One obvious advantage of this operation is large reduction of weights. If the image has a size of  $2000 \times 2000$  pixels with depth 3, a neuron in a fully connected network needs  $2000 \times 2000 \times 3 = 12000000$  weights. But for a convolution operation, the parameters of the filter are shared through the whole image. Therefore, a filter with size of  $5 \times 5$  only needs  $5 \times 5 \times 3 = 75$  weights. Occasionally, zero-padding can be used to add zeros around the borders of the input, thus controlling the spatial size of the output. Stride in a CNN represents the number of moved pixels for filters after each computation. If the stride is equal to 1, the filter would jump only 1 pixel each time, and the output size would remain the same if the zero-padding is also utilized. When the stride is larger than 1, the output size would become smaller. Figure 21 shows a example with stride of 2.

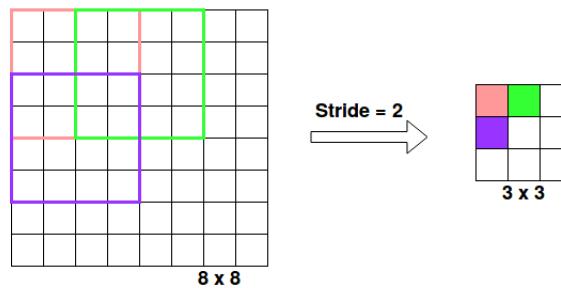


Figure 21: The  $8 \times 8$  layer becomes  $3 \times 3$  after the stride 2 convolution. The filter size is  $4 \times 4$  and the color boxes from the left plot correspond to the respective color boxes at the right.

Pooling layer is commonly used between two convolution layers in a CNN. For each channel of the 3-dimensional box input, pooling operation reduces the dimension at certain scale. Therefore, the number of weights and computation is also heavily reduced. Filters of a square size of  $2 \times 2$  along with a stride size of 2 is normally used for downsampling. For each pooling operation, the maximum or average value is taken over the 4 neurons. After a pooling process with a filter size of  $2 \times 2$  and a stride size of 2, the height and width of the input would become half of their original values. The depth will remain the same after the process. Figure 22 gives an example illustrating how a  $8 \times 8$  layer is halved after a pooling operation.

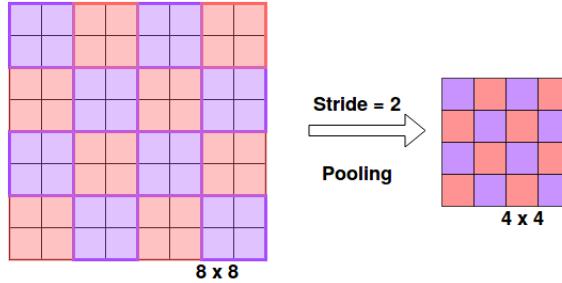


Figure 22: The  $8 \times 8$  layer becomes  $4 \times 4$  after the stride 2 pooling operation with filter size of  $2 \times 2$ .

Convolution layers need to be converted into fully connected layer at certain point, since many tasks only need a 1-dimensional vector as vector. This can be simply done by resizing. Then the converted layer can become input for final classification layer or part of other neural network structures. Figure 23 is a fully structured CNN for classification of 10 classes. The 3-channel image with a size of  $8 \times 8$  is concatenated by a convolution layer with 16 filters, pooling layer, 64-filter convolution layer, and another pooling layer. Then the 3-dimensional box is converted into a fully connected layer. After another fully connected layer with its dimension of 500, the final classification layer can output probabilities for the 10 classes.

### 3.2.8 Recurrent Neural Networks and LSTM

Unlike regular fully connected neural networks, recurrent neural networks (RNNs) have a time-step notion that can deal with time series input. RNN incorporates a loop structure which allows information pass from one time-step to the next one. Figure 24 shows an unfolded loop structure, where  $x_t$  represents an input vector at time step  $t$ ,  $A$  is some operation of the input and  $h_t$  is a hidden state vector at time step  $t$ . Therefore, time series input can be processed, and one or multiple output values can be obtained at any time-step. These advantages of this structure are beneficial for applications that deal with sequences, such as speech recognition and machine translation. Equation 15 represents the vanilla RNN.

$$h_t = \sigma(W^H h_{t-1} + W^X x_t) \quad (15)$$

It can be seen from the equation, the computation of the hidden state of current

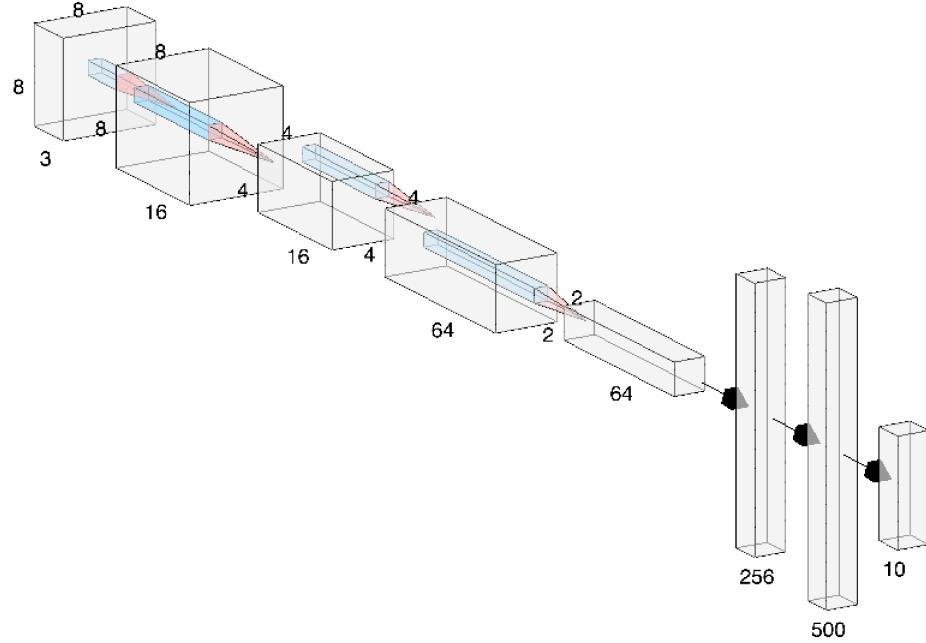


Figure 23: A typical example of CNN with  $8 \times 8$  pixels and 3 channels input and output probabilities for 10 classes.

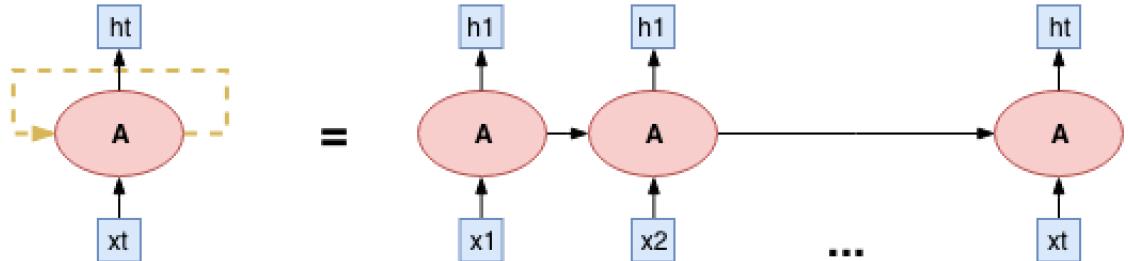


Figure 24: An unfolded presentation of the loop structure of RNN.

time-step relates the values from both the previous hidden state  $h_{t-1}$  and the current input vector  $x_t$ .  $W^H$  controls the importance of previous information, while  $W^X$  controls the importance of current information. This mechanism is similar to the human's perception of sequences utilizing both previous memory and the newest information. However, it is still difficult to solve long time temporal dependencies problems due to the problem of vanishing gradient.

Long short-term memory network (LSTM) is a special type of RNN that can learn long time dependencies in a higher complex level computation. The computation  $A$  in Figure 24 is divided into four parts as shown in Figure 25. With the addition of cell states  $C_t$ , the process of LSTM can be roughly explained as follows. The forget gate  $f_t$  outputs  $[0, 1]$  to represent the amount of retained information:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (16)$$

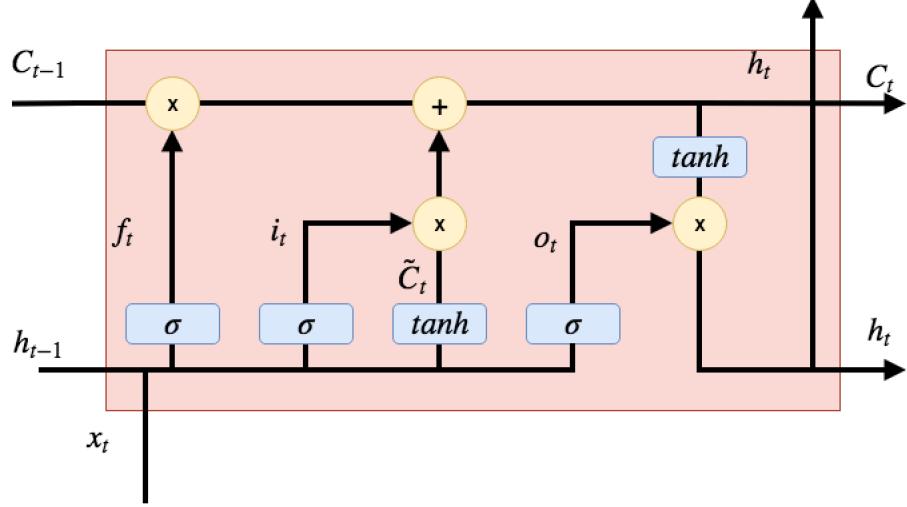


Figure 25: A LSTM structure.

Then input gate  $i_t$  outputs  $[0, 1]$  to represent the amount of stored information to a cell:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (17)$$

The output gate  $o_t$  outputs  $[0, 1]$  to represent the amount of output information from the current cell state:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (18)$$

A new cell state  $\tilde{C}_t$  that needs further modification is created through  $\tanh$  activation:

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \quad (19)$$

Modifications for the cell state consists of using the forget gate on previous cell and using the input gate on non-modified current cell:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (20)$$

The new cell state after a  $\tanh$  activation is then multiplied by output gate  $o_t$  to obtain the update of a hidden cell.

$$h_t = o_t * \tanh(C_t) \quad (21)$$

Because of these controlling gates, LSTM is able to learn long time dependencies in a sequence.

### 3.2.9 Multi-level Attention Model

Attention model is another strategy for dealing with time series input by assigning different weights for the instances of a time series. Although having simpler structure than RNN, attention model has been shown to perform better an audio event classification task [34].

For the tasks of audio event classification, dynamic changes are important characteristics to model. Although spectrum structures are quite useful for the recognition of melodic sound and instruments, sometimes dynamic changes might also be needed in classes with high abstract level descriptions. Some high abstract level descriptions include music genres, such as 'hip hop music', 'rhythm and blues', and 'rock music'. Recognizing them is harder than instruments, because this recognition involves the time evolution of musical components. For recognizing those less musical classes such as 'gunshot' and 'knock', the temporal envelopes are more helpful.

An example of attention model can be seen in Figure 26, where  $B_n$  represents a time series with fixed length  $T$ , and  $x_1 \sim x_T$  are the  $T$  instances of this series.  $f_k$

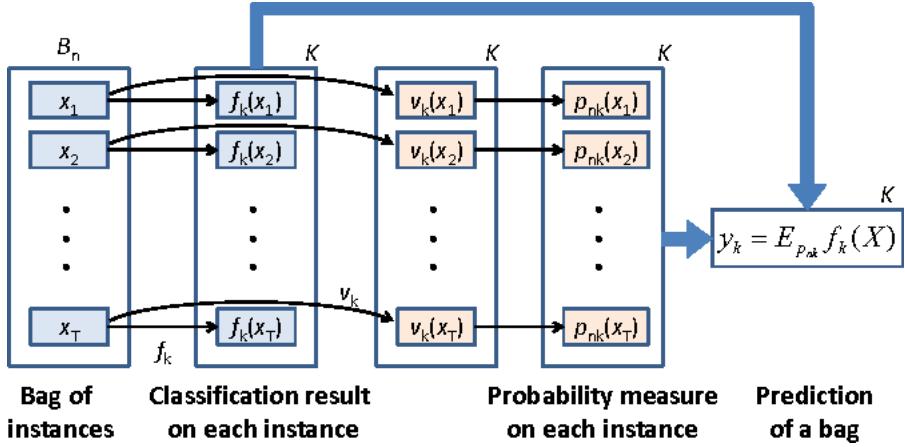


Figure 26: An attention model example [34].

represents the classification process and  $f_k(x_n)$  is the classification results for instance  $x_n$ . For audio input,  $x_n$  could be a image of spectrogram, and  $f_k$  could therefore be some types of CNNs applying on the image. Attention models attempt to model the weights or importance of each instance in a time series, thus making more relevant instances have higher contributions for classification. The weights  $v_k(x_1) \sim v_k(x_T)$  are firstly learned for each instance. However, the sum of the weights is not equal to 1. Then a softmax layer can be used to normalize the weights to  $p_{nk}(x_1) \sim p_{nk}(x_T)$  so that  $\sum_i p_{nk}(x_i) = 1$ . Finally, as shown in Equation 22, the final prediction probability can be obtained for a class by multiplying the classification results  $f_k(x_1) \sim f_k(x_T)$  with respective normalized weights and summing them together.

$$\begin{aligned} y_k &= E_{p_{nk}} f_k(X) \\ &= \sum_i f_k(x_i) p_{nk}(x_i) \end{aligned} \quad (22)$$

Since 2-dimensional audio spectrograms can also be treated as image data, audio event classification therefore can take similar CNN models as image classification. However, the classes for audio are highly diverse, and they favor different levels of abstractions and time spans. Recognizing a piano sound only takes a short segment of spectrograms, but higher level analysis might be needed for global and comprehensive classes, such as acoustic scenes and music genres. Combining features extracted

from different levels in a CNN has been proven useful in computer vision tasks [43]. Different levels of convolution layer could learn different characteristics of a image, such as edges and curves. A CNN-based architecture [38] has also shown great performance for music tagging by aggregating multi-level and multi-scale features. This thesis failed to incorporated multi-scale features, because the features extractor we used only accepts fixed length audio.

Inspired by the studies [34] [73] on Google AudioSet, this thesis decides to choose a similar multi-level attention structure for audio. Figure 27 shows the structure of a 6-instance multi-level attention model. Other models with different number of instances and levels have similar structures. The audio is split into a time series

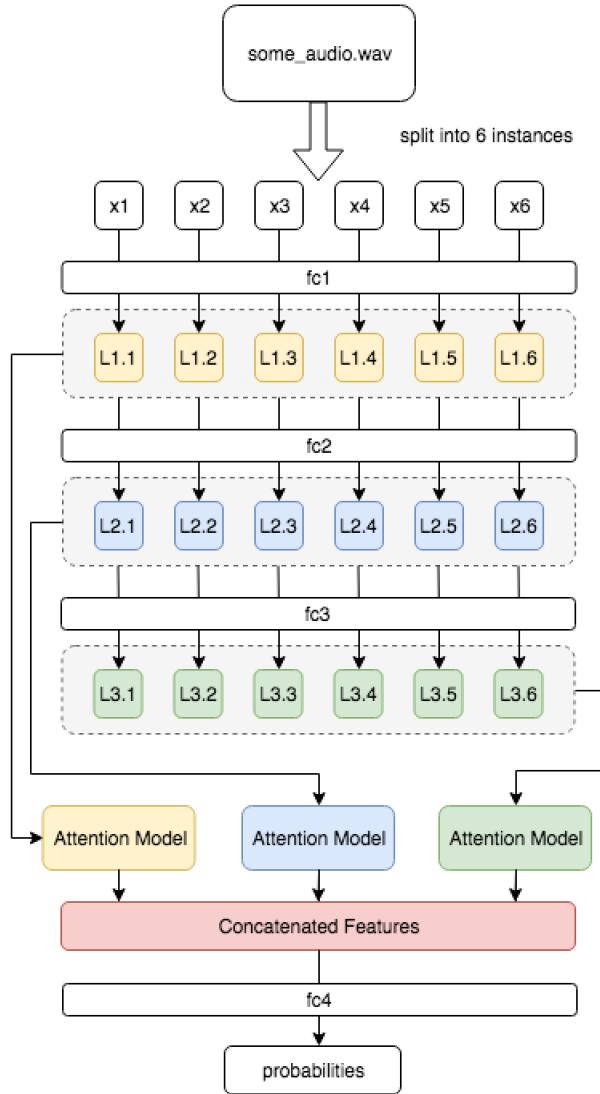


Figure 27: An 6-instance multi-level attention model example.

with 6 instances  $x_1 \sim x_6$ . Three fully connected layers are added for representing different levels. The yellow, blue, and green boxes represent the 1st, 2nd, and 3rd level features respectively. Then the time series from the gray boxes are fed into

their own attention models. Finally, the classification results from these attention models are aggregated together for the final layer of classification.

### 3.3 Log-Mel Spectrogram

Audio input features are less obvious than images, and lossless raw audio waveform contains a long stream of digits. Historically, hand-crafted features are specifically designed and computed for relevant applications. Methods for building the features are mainly based on different forms of spectral shapes involving fast Fourier transform on raw audio waveform. Recently, utilizing raw audio waveform as direct input for neural networks becomes possible, and it can achieve relevant good results. However, utilizing raw audio waveform is still not fully developed yet, many applications still use the traditional features of spectral shapes.

MFCCs, which contain lossy spectral shapes for audio segments, have shown their effectiveness for phoneme recognition. However, MFCCs are compact and compressed features that are quite limited for general audio event classification. Therefore, the various types of audio events need a more detailed spectral representation.

Spectrogram is a better alternative which has a series of spectrum representing the time evolution of frequency components. The log-mel spectrogram resembles a lot to the working mechanism of our cochleas. Cochleas are hearing organs of human used to perceive sound. Spectrum contains an array of values which representing the strength of corresponding frequencies, while cochleas also perceive sound according to different frequencies. Sampling rates could affect the effective frequency range of spectrum. According to Nyquist theorem, loss does not exist for frequency components whose frequencies are smaller than half of the sampling rate. The frequency range of humans' hearing is approximately between 20 Hz to 20,000 Hz, which provide good reference when choosing sample rates. However, the perception of frequencies in cochleas is not constant nor linearly proportional with frequency. Stevens *et al.* proposed the perceptual mel scale of pitches based on the experimental and averaged results of a set of people subjects [63]. This scale provides a more suitable model for the sound perception of human. Therefore, common spectrograms need to be converted into mel spectrograms by a mapping shown in Equation 23 [63].

$$m = 1127 \ln\left(1 + \frac{f}{700}\right) \quad (23)$$

$f$  refers to actual frequency, and  $m$  is the mel frequency. Since human perceive the loudness of sound logarithmically, a log operation is performed for the values of mel spectrograms. The converted spectrograms are called log-mel spectrograms and are used for the tasks of this thesis.

The computation of the log-mel spectrogram is described as follows. The raw waveform for an audio signal is firstly split into overlapped frames. The frame length is normally 15-25ms where audio can be perceived as approximately stationary. The hop-size, normally around 10ms, represents the time-shift between frames. This shift can reduce information loss caused by the split. Besides, a windowing function is generally applied for each frame. The windowing functions normally are smooth

and have zeros at both ends, which can avoid spectral leakage. The leakage is mainly caused by the usage of rectangle windows in frequency domain. Discrete Fourier transform is then applied to transfer the signal of time domain into frames of spectrograms. With transformation from  $Hz$  to  $mel$  and the log operation, the final frames of log-mel spectrogram are obtained. Figure 28 shows a normal log-frequency spectrogram with 1025 frequency bands and a log-mel spectrogram with 128 mel bands.

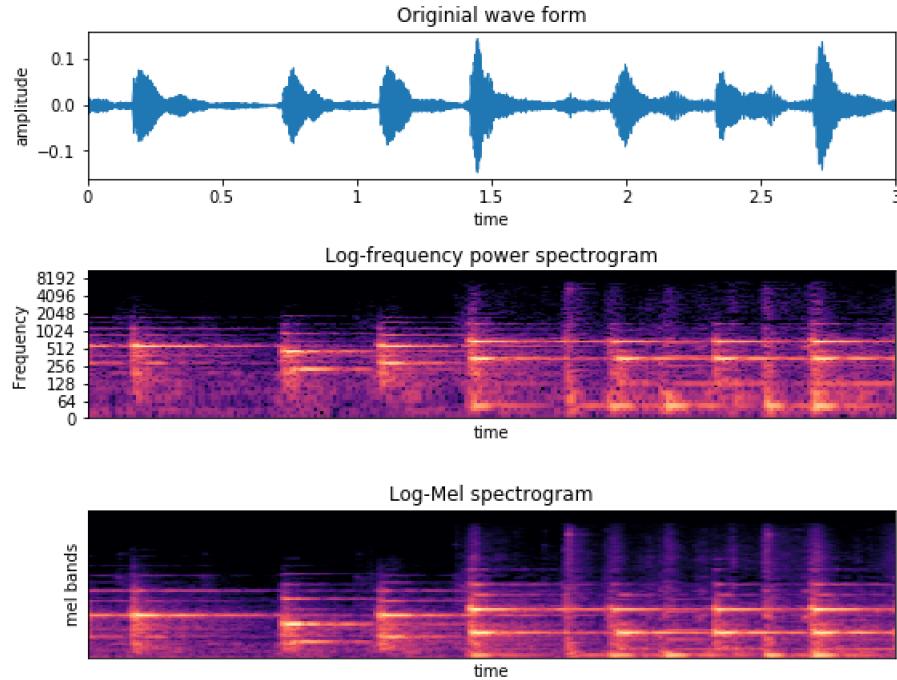


Figure 28: Plots from top to bottom are original waveform, a log spectrogram with 1025 frequency bands, and a log-mel spectrogram with 128 bands.

## 4 Research Tasks

This section describes the two research tasks of the thesis: Google AudioSet and DCASE 2018 Task2. It introduces the settings for both tasks and analyzes both datasets in detail. Furthermore, the evaluation metric for these tasks is also explained here.

### 4.1 Google AudioSet

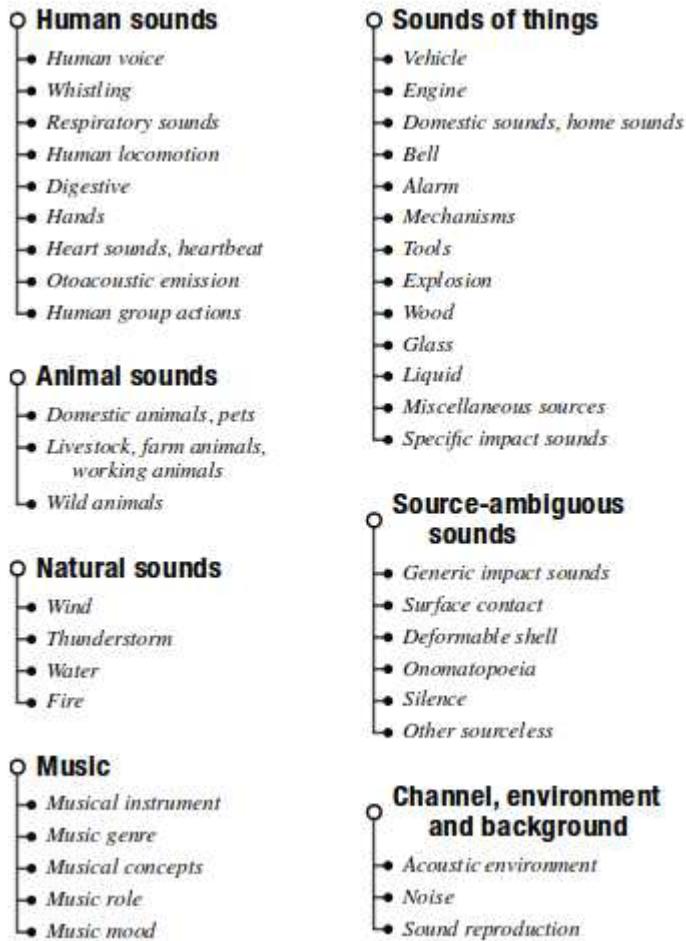


Figure 29: Top 2 layers of Google AudioSet ontology [18]

Google has created a dataset called AudioSet along with a carefully structured hierarchical ontology of 632 classes. Figure 29 shows the top two layers of this ontology. This dataset can be used for performing a multi-label classification task involving the assignment of one or more labels from the given classes. The creation of AudioSet involves human annotators probing the presence of audio classes from the given 10-second segments of YouTube video [18]. All the audio tracks in the dataset are chosen as 10-second long. The given information includes the video ID from YouTube, start

Table 1: VGGish model structure. The kernel size for CNN is (3, 3) and the kernel size for pooling is (2, 2). The activation function is 'relu' for all the layers.

layers	filters@size	layers	filters@size
<b>1. input</b>	1@96×64	<b>9. conv</b>	512@12×8
<b>2. conv</b>	64@96×64	<b>10. conv</b>	512@12×8
<b>3. pooling</b>	64@48×32	<b>11. pooling</b>	512@6×4
<b>4. conv</b>	128@48×32	<b>12. flatten</b>	12288
<b>5. pooling</b>	128@24×16	<b>13. fc</b>	4096
<b>6. conv</b>	256@24×16	<b>14. fc</b>	4096
<b>7. conv</b>	256@24×16	<b>15. fc</b>	128
<b>8. pooling</b>	256@12×8	<b>16. ...</b>	...

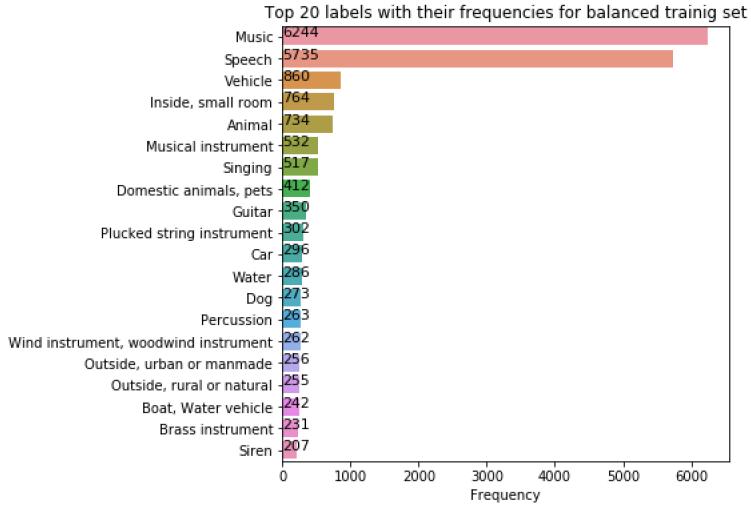


Figure 30: Distribution of the samples of the balanced training set.

& end time indexes, and labels. Instead of releasing original soundtracks, Google provides a compact version with each 10-second segment represented as  $10 \times 128$  features. 10 means the 10-second audio is split into 10 successively non-overlapping 960 ms frames, and each frame is fed into a pre-trained CNN model to generate the 128-dimensional features. The model is a VGG-inspired acoustic model trained on a preliminary version of YouTube-8M. The structure can be seen in Table 1. The generated  $96 \times 64$  log-mel spectrogram for 960 ms audio is fed into this CNN-like model. The bottle neck 128-dimensional features were PCA-ed and quantized to be compatible with the audio features provided with YouTube-8M. The detail of the model can be seen in the GitHub repository<sup>1</sup>.

The dataset has three parts: balanced training part, unbalanced training part, and evaluation part. Only 527 classes from the ontology are used. The balanced training set has 22,160 training samples lasting around 61 hours, where each category has at least 59 training samples. Evaluation dataset has 20,371 samples corresponding

<sup>1</sup><https://github.com/tensorflow/models/tree/master/research/audioset>

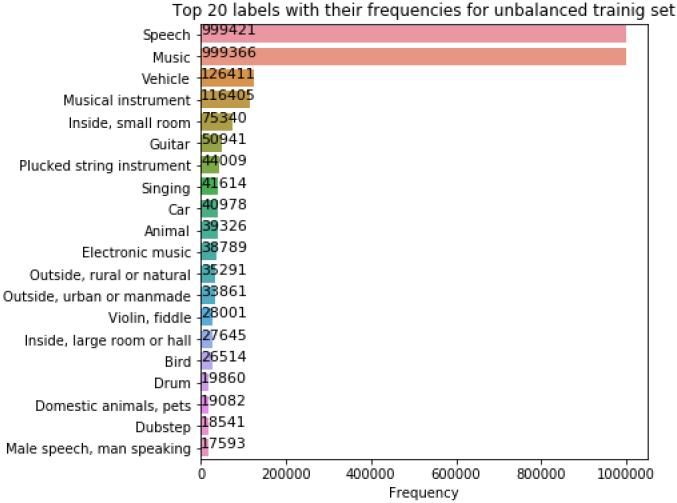


Figure 31: Distribution of the samples of the unbalanced training set.

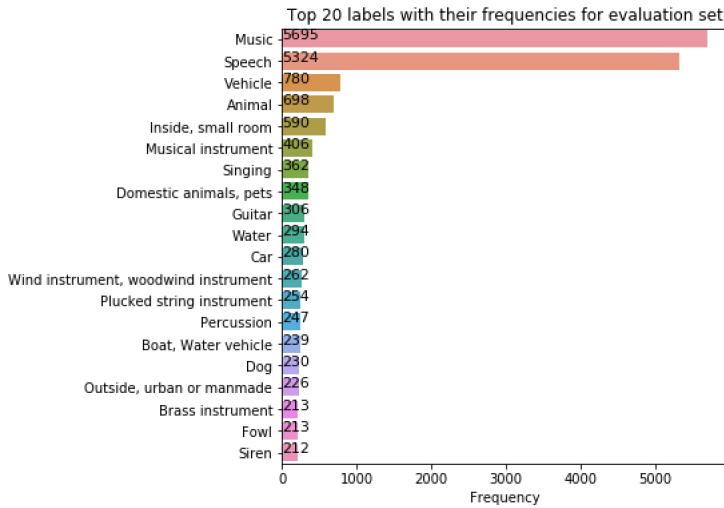


Figure 32: Distribution of the samples of the evaluation set.

to around 56 hours, where each category also has at least 59 different samples. The unbalanced training data contains the rest 2,041,789 samples corresponding to 5,671 hours.

Figure 30, 31, and 32 respectively illustrate the distributions of most frequent 20 classes for balanced training, unbalanced training, and evaluation set. Since it is a multi-label classification task, each sample could have one or more classes meaning overlapping audio events could happen at the same time. The labels of an sound segment can come from different branches of the ontology, such as 'male speech', 'dog', and 'music'. However, sometimes the labels have a hierarchical structure, such as 'music', 'musical instrument', 'plucked string instrument', and 'guitar'. In the hierarchical labels, previous class is the mother of the next class. In addition, some labels only have child node classes without having all their ancestor classes. The

connectivity between classes need to be considered when building a more advanced model.

## 4.2 DCASE 2018 Task2

DCASE 2018 task2, 'General-purpose audio tagging of Freesound content with AudioSet labels', is a multi-class classification task. The goal of this task is to assign one tag for each audio clip from 41 classes. These classes are originated from the AudioSet ontology, including 'flute', 'saxophone' and 'cough'.

The given data of this task consist of compressed PCM 16 bit, 44.1 kHz, mono audio files, which are collected from Freesound dataset [14]. The data is split into training and testing parts. The training set has 9473 samples and has an unbalanced distribution of classes. The minimum and maximum number of samples for each class are 94 and 300. Figure 33 shows the distribution of the 41 classes, where around 20 classes have 300 training samples. Unlike the AudioSet, which has fixed

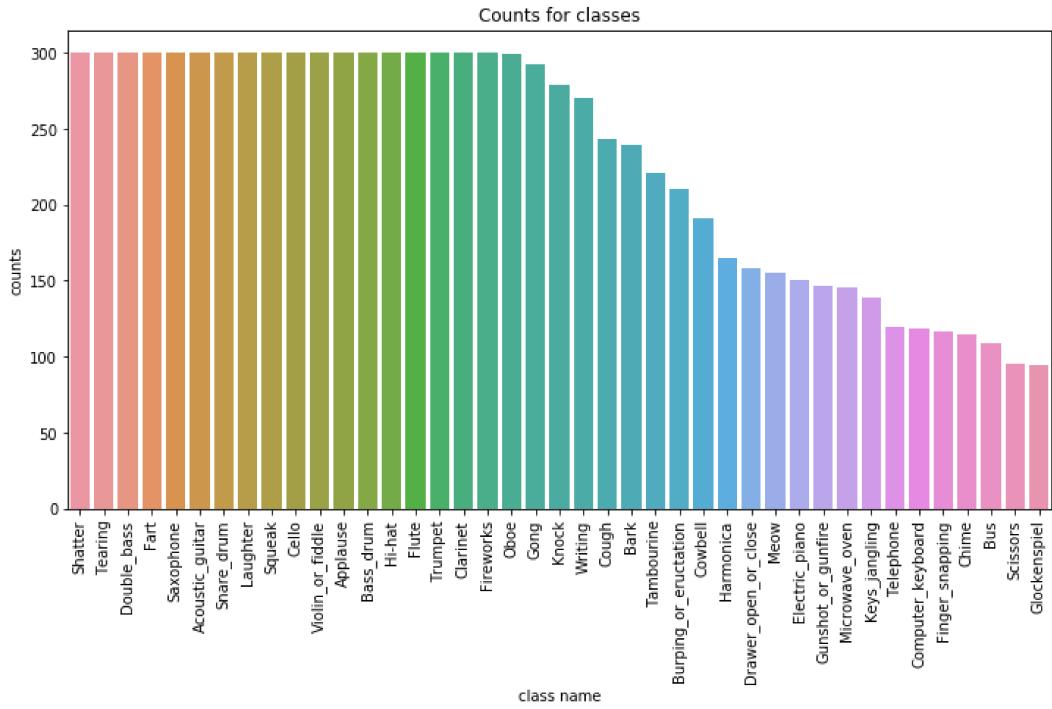


Figure 33: Distribution of the samples along the 41 classes.

10-second length audios, the length of the audios for this challenge varies from 300ms to 30s. The audio length distribution for these 41 classes can be seen in Figure 34. In addition, 3710 among the 9473 samples are also labelled as manually-verified. These verified samples have labels that are surely present and predominant. Users can opt to use the rest unverified samples for developing their systems [13]. The testing dataset contains 1600 samples which share similar distribution as the training set. It also has 7800 padding sounds that are not used for evaluation. This task is also

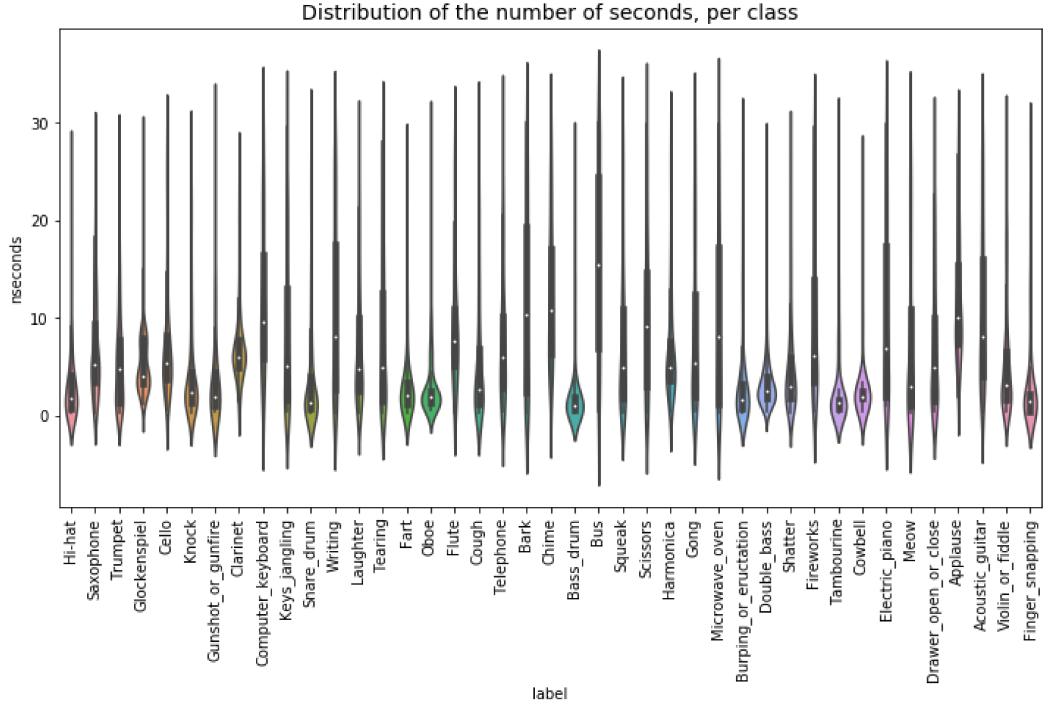


Figure 34: Distribution of the number of seconds for each class.

hosted on Kaggle, a platform for machine learning competitions<sup>2</sup>. For evaluation of the online submissions, 19% of the testing samples is used for evaluation in public leaderboard, while the rest 81% is used for evaluation in private leaderboard when the competition ends.

### 4.3 Evaluation Methods

#### 4.3.1 Mean Average Precision

Average precision(AP) is a commonly seen evaluation metric in information retrieval. The averaged single-value version over all the queries is called mean average precision(MAP), and it is very popular among many classification challenges, including Google AudioSet and DCASE 2018 Task 2.

Suppose a series of classification results for the audio event class 'guitar' is given. The precision is calculated at every correctly classified audio, and AP is the average of all the calculated value. For example, if the returned results is 1, 0, 1, 1, 0, 1, where 1 represents the sound is correctly classified, while 0 is not. Table 2 shows the precision values of four correctly classified points. The AP is then calculated as the average of these four values:  $(1 + \frac{2}{3} + \frac{3}{4} + \frac{4}{6})/4 = 0.77$ . Thus an AP of 0.5 would have results like 0, 1, 0, 1, 0, 1, ... where every second classification is correct while an AP of 0.3 would have results 0, 0, 1, 0, 0, 1, 0, 0, 1 where every third classification is correct. Therefore, AP over 0.5 means only one correctly classified samples for every

<sup>2</sup><https://www.kaggle.com/c/freesound-audio-tagging>

Table 2: Average Precision for the returned classification results.

	1	2	3	4	5	6
Classification Results	1	0	1	1	0	1
Precision	1	/	$\frac{2}{3}$	$\frac{3}{4}$	/	$\frac{4}{6}$

two samples. Equation 24 shows the calculation of MAP by averaging AP scores over all classes.

$$MAP = \frac{1}{N} \sum_i^N AP_i \quad (24)$$

For the DCASE task, the evaluation uses Mean Average Precision @ 3 (MAP@3)<sup>3</sup>. The detailed implementation and explanation can be seen in the footnote link. Simply speaking, up to three predictions can be given even though only one correct label exists. The order of predictions matters in this setting. If the correct label is predicted at the 1st position, a system can get a score of 1. Placing in a 2nd position means a score of 1/2, and 3rd position represents a score of 1/3. If no correct answer is given in the three predictions, the score would be zero. The average of scores for all testing samples is defined as the final evaluation score.

### 4.3.2 Area Under Curve

Table 3: Explanation on TP, FP, TN, and FN.

audio event		
	present	absent
positive	True Positive(TP)	False Positive(FP)
negative	False Negative(FN)	True Negative(TN)

Area under curve(AUC) is the area under the receiver operating characteristic (ROC) curve, where true positive rate (TPR) is plotted against false positive rate (FPR) for different cut-off criterion. Table 3 explains true positive and false positive. Positive or negative means if a sample belongs to a class, while true or false means if a sample is correctly classified or not. Therefore, true positive means the prediction of an audio sample is true and the corresponding audio event does present in the sample. False positive means the prediction is false but the expected audio event does not present. The following equations explain the calculations of TPR and FPR. Based on different criterion value, many TPR and FPR pairs can be obtained. Plotting the pairs into a graph would give the ROC curve shown in Figure 35. Since a good classification system should have a high TPR and low FPR, the system with perfect

---

<sup>3</sup>[https://github.com/benhamner/Metrics/blob/master/Python/ml\\_metrics/average\\_precision.py](https://github.com/benhamner/Metrics/blob/master/Python/ml_metrics/average_precision.py)

discrimination should have a ROC curve that passes through the upper left corner.

$$\begin{aligned} TPR &= \frac{TP}{TP + FN} \\ FPR &= \frac{FP}{FP + TN} \end{aligned} \quad (25)$$

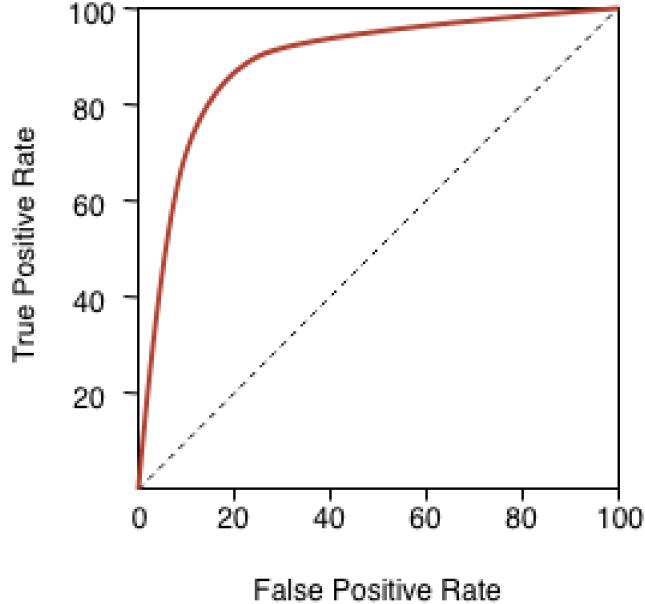


Figure 35: An example ROC curve

AUC then simply means the ratio between the area under the ROC curve and the whole square area of the plot. Therefore, bigger AUC indicates a better system.

Another metric related to AUC is d-prime. Equation 26 shows the calculation process of d-prime, where  $Z$  represents the inverse of the cumulative distribution function of Gaussian distribution. Simply speaking, d-prime is a special mapping from AUC and has no maximum, while AUC ranges from 0 to 1. Both AUC and d-prime are used for evaluation of Google AudioSet.

$$\text{d-prime} = \sqrt{2}Z(AUC) \quad (26)$$

## 5 Experiments

This section states both experiments for Google AudioSet and DCASE 2018 Task 2. Methods used for AudioSet are explained in detail, including the preparation of the input features, mini-batch balancing techniques, and different model structures. For DCASE Task 2, an additional fine-tuning process is introduced before generating input features for neural network models. Pitch-shifting, truncating and repeating, and mixup are used for augmentation and generalization. In addition, this thesis also trains multi-level attention models with different number of segments for the DCASE task. Both experiments aimed at finding difference among multi-level attention models with different combination of levels and segments.

### 5.1 Google AudioSet

#### 5.1.1 VGGish Generated Features

Firstly, the forms of training data need to be specified for the input of a neural network. As mentioned in previous section, each training example from AudioSet contains 10 instances, each of which has 128 compact features representing the corresponding 960ms audio segments. Thus, the input size is  $(N, 10, 128)$  for AudioSet with  $N$  representing the number of training data and 10 representing the 10-second segments.

To get features with dimension  $10 \times 128$ , the 10-second audio is firstly split into 10 non-overlapping 960ms frames. For each 960ms frame, the  $96 \times 64$  log-mel spectrograms are computed following the hyper parameters shown in Table 4. Since

Table 4: Parameters for getting the log-mel spectrogram

Parameters	
Re-sampling Rate	16kHz
STFT Window Length	25ms
STFT Window Type	Hann
STFT Hop Size	10ms
Number of Mel Bins	64
Min Frequency for Mel	125Hz
Max Frequency for Mel	7500Hz

these audio files are gathered through many YouTube videos from all over the world; they might have different types of recording qualities, including sampling rates, number of channels, bit-depth. Therefore, a standard, which is neither too high nor low, is needed to convert those files into one format. For example, a higher sampling rate will not help if the original sampling rate is already too low. The audio files are firstly averaged into one channel and then re-sampled into 16kHz. According to Nyquist theorem, the frequencies below 8kHz are perfectly retained if the sampling rate is 16kHz. Frequencies below 8kHz represents a reasonable range for identifying common audio events. Increasing sampling rate will give more information of audio contents and possibly increase the performance of an audio event classifier.

Then log-mel spectrograms are computed following the parameters in Table 4 and procedures in Section 3.3. Since the number of mel bins is 64 and the hop size is 10ms, the log-mel spectrogram for 960ms audio would have dimension of  $96 \times 64$ . Finally, the computed spectrograms are fed into the pre-trained VGGish model for the generation of 128 features. The structures and explanation of VGGish can be seen in Table 1. These features are further processed using principal component analysis (PCA) and 8-bit quantization.

### 5.1.2 Mini-batch Balancing

The training data contains a balanced dataset and an unbalanced dataset. The balanced dataset has at least 59 examples for each class, many classes have more examples due to the label co-occurrence. The classes with higher hierarchies normally have more examples. For example, the class 'guitar' and 'flute' both belong to the class 'musical instrument'. The balanced dataset has only 22,160 training examples, while the unbalanced dataset has 2,041,789. The distribution of these classes are highly unbalanced, which is shown in Figure 30 and 31. This could be a problem during the training phase, because a model tends to focus more on classes with many training examples and stops learning from classes with fewer examples.

To fully use the data and avoid the problem of data imbalance, this thesis uses a technique called mini-batch balancing. Mini-batch balancing means assigning equal number of training examples for each class in a training batch. After the combination of the balanced and unbalanced training set, this thesis set a certain batch size. The parameters of a model are updated after each batch. For every batch, equal number of examples from each class are selected for training. Since the batch size is not always the integer times of the number of the classes, extra examples have to be left out into the next batch. If the batch size is 500, and one example from each of the 527 classes is taken, 27 extra examples are needed to be fed into the next batch. But for the next batch, only  $500 - 27 = 473$  examples are needed. Thus, the training will still overall be balanced.

### 5.1.3 Different Model Structures

Inspired by studies [34] and [73], this thesis decides to compare the performance among deep neural nets (DNN), long short-term memory networks (LSTM), attention models, and multi-level attention models. All the activation functions are 'relu' except that the final classification layer use 'sigmoid' activation with AudioSet being a multi-label classification task.

The detailed model settings can be seen from Table 5, 6, and 7 for DNN, LSTM and multi-level attention models respectively. All of them used batch normalization, dropout rate of 0.4 and Adam optimizers. For the DNN models, a 1-dimensional global average pooling layer is also added to get the average results for these 10 instances. For the LSTMs, they use the same number of hidden units as DNN and feed the last output from LSTM layer to the final classification layer. As for the multi-level attention models, they basically have the same structure as DNN with the additional attention mechanism added after each levels. In Table 7, (h1, h2,

Table 5: Parameters for DNN model  
DNN Parameters

Number of Hidden Layers	3
Number of Hidden Units	[600, 600, 600]
Batch Size	500
Optimizer	Adam( $\beta_1=0.9$ , $\beta_2=0.999$ )
Learning Rate	0.001
Dropout Rate	0.4
Batch Normalization	Yes

Table 6: Parameters for LSTM model  
LSTM Parameters

Number of LSTM Layers	1
Number of Hidden Units	600
Return Sequences	False
Batch Size	500
Optimizer	Adam( $\beta_1=0.9$ , $\beta_2=0.999$ )
Learning Rate	0.001
Dropout Rate	0.4
Batch Normalization	Yes

h3) means the first, second and third hidden layer. One or many output from these hidden layers can be selected as input for the attention layers. For the example shown in Figure 27, levels=[h1, h3] means only concatenating the output from the yellow and green attention layers for the input of final classification layer. In total, these experiments have 7 different settings for 3-level attention models: [h1], [h2], [h3], [h1, h2], [h1, h3], [h2, h3], and [h1, h2, h3].

During the training process, binary crossentropy is used as the loss function. Since the combination of balanced and unbalanced training set is too larger. Thus, MAP, AUC, and d-prime scores for the evaluation set are computed for every 1000 iterations. The model will stop training when MAP scores stop increasing for continuous 10 times.

## 5.2 DCASE 2018 Task2

### 5.2.1 Preprocessing

The dataset for DCASE 2018 Task2 uses actual audio files instead of the compact 128-dimensional features of AudioSet. This is a more general setting for the audio event classification task, and the usability of the VGGish feature-extractor from AudioSet can also be tested in this challenge. The files are provided as PCM 16 bits, 44.1 kHz, mono format. However, this format is the result after conversion. The original recording qualities can be quite different, since they are collected from users all over the world. For the preprocessing, the silence parts at the beginning and the

**Table 7: Parameters for multi-level attention model**

Multi-level Attention model Parameters	
Number of Hidden Layers	3
Number of Hidden Units	[600, 600, 600]
Levels	1 or many from [h1, h2, h3]
Batch Size	500
Optimizer	Adam( $\beta_1=0.9$ , $\beta_2=0.999$ )
Learning Rate	0.001
Dropout Rate	0.4
Batch Normalization	Yes

end are trimmed because of their irrelevance for classification. Librosa<sup>4</sup> toolbox is used for the trimming. The silence parts in the middle remained untouched, because they might contain important dynamic information for the classification. The volume of these audio files can be quite different, thus their amplitudes are normalized to [-1, 1] using the same librosa toolbox. Finally, the trimmed and normalized audio files are split into non-overlapping 960ms frames for the generation of log-mel spectrograms with dimension  $96 \times 64$ . These spectrograms are used for fine-tuning the VGGish model.

### 5.2.2 Fine-tuning the VGGish Model

The original VGGish model is trained for Google AudioSet with 527 classes following the AudioSet ontology. Compared to AudioSet, DCASE 2018 Task 2 is a multi-class classification task with only 41 classes originated from AudioSet. This task only needs one correct label for each training example. VGGish model can be fine-tuned for this task. After the layer that output 128-dimensional features for given  $96 \times 64$  log-mel spectrograms, one hidden layer with 100 units is added. Furthermore, the final classification layer has 41 units with 'softmax' activations. The training data for the fine-tuning is log-mel spectrograms with dimension  $96 \times 64$  for each 960ms segment rather than the whole audio clips. Therefore, audio files are needed to split into non-overlapping 960ms segments, and then the segmented clips need be assigned with the same label that belong to the original audio clip.

Section 4.2 has described the dataset used for this task in detail. For the fine-tuning, the 9473 audio files need to be split into a training part and a validation part. The validation loss is used as stopping criterion to avoid overfitting. At the beginning, the first 8000 audio files are selected as training data, and the rest 1473 are selected as validation data. However, the last 1473 audio files were not learned by the model. To fully use the data, the thesis uses another training and validation split with the last 8000 audio files as training data and the first 1473 as validation data.

Mini-batch balancing, which assigns equal number of training examples for each class in a batch, is also used during the fine-tuning process. The batch size is set as

---

<sup>4</sup><https://github.com/librosa/librosa>

Table 8: Different fine-tuning strategies.

	Training/Validation Split	
	[1-8000]/[8001-9473]	[1474-9473]/[1-1473]
with Mini-batch Balancing	Balanced First	Balanced Last
without Mini-batch Balancing	Unbalanced First	Unbalanced Last

41. Therefore, each batch contains one training example from every class. However, the balancing is not perfect, because the audio files might contain different number of 960ms segments. Since the average length of audio files is 6.7 seconds, around 280 log-mel spectrograms will be existing in each mini-batch. Unbalanced models are also fine-tuned by randomly choose 41 audio files in each batch for comparison. Table 8 has shown all 4 fine-tuned models.

### 5.2.3 Data Augmentation

The dataset is unbalanced, and only around half of these classes have 300 soundtracks as training data. Figure 33 and 34 have shown the detailed distributions. The imbalance problem can make a model emphasize on classes with more training examples and neglect to learn from classes with limited number of training examples. Therefore, data augmentation is needed for dealing with this problem. This thesis uses several augmentation techniques, such as pitch-shifting, repeating, and truncating of audio clips.

Pitch-shifting chooses an integer number between -12 and 12 for each audio file, and the corresponding number of semitones are then shifted to create pitch-shifted audios. This is achieved repeatedly until having 600 training examples for each class. Since around half of these classes are melodic instruments or have stable spectrum structures. This maximum 12-semitone shifting does not affect perception heavily. For the noise-like classes, such as 'shatter' and 'fireworks', relatively good perception can still be obtained.

Some models need fixed length audio input, such as multi-level attention models. However, provided audio files have variable length. Thus, taking only 6-second audio out of a 30-second audio would be a total waste. For the purpose of fully utilizing the training data, a 30-second audio is better to be split into 5 6-second audio segments. For audio files that are shorter than the specified length, fixed length can be achieved by repeating and concatenating them together.

### 5.2.4 Mixup

Mixup are both generalization and augmentation techniques. It creates new training examples by using the convex combinations from pairs of examples and their corresponding labels. This mixup helps neural networks to favor the simple linear combination between the training examples. This process can improve the generalization ability, reduce memorization of corrupted labels, increase the robustness of adversarial examples [74]. The training data has another label representing if an audio signal is manually verified. Thus, this mixup method can help reduce the

potential effect of learning wrong information from unverified audio signals. Besides, this will also make a model more sensitive to distinguish between classes. Equation 27 explains the working principle of mixup.

$$\begin{aligned}\tilde{x} &= \lambda x_i + (1 - \lambda)x_j \\ \tilde{y} &= \lambda y_i + (1 - \lambda)y_j \\ \lambda &\sim \text{Beta}(\alpha, \alpha) \\ \alpha &\in (0, \infty)\end{aligned}\tag{27}$$

The  $(x_i, y_i)$  and  $(x_j, y_j)$  are two training pairs with  $y_i$  and  $y_j$  representing the labels. Parameter  $\alpha$  controls Beta distribution and  $\lambda$  is drawn from the Beta distribution and lies in the region of  $[0, 1]$ . The experiments in study [74] shows that increasing the  $\alpha$  would increase the training error on real data and minimize the generalization gap, thus meaning large  $\alpha$  might result in underfitting. Even though no clear understanding has been shown behind the theory, they found that  $\alpha \in [0.1, 0.4]$  would lead to improved performance. In our experiments, we used mixup to generate the same amount of training data based on four different  $\alpha$ : 0.1, 0.2, 0.4, and 1.0. Pitch-shifting, repeating, and truncating are firstly used to create a balanced dataset. Then, mixup is used to create the same amount of training examples as the balanced dataset. Finally, all these training examples are fed into neural networks for training.

### 5.2.5 Different Model Structures

For testing the performance of VGGish models with the 4 different fine-tuning techniques, these features are tested on a same 1-segment multi-level attention model. The structure follows the Figure 27 with the combination of all three levels for 1-segment input.

Table 9: Different model structures.

Model Structure	Input Size	Note
1-Segment Multi-level Attention	(N, 1, 128)	
LSTM	(N, var, 128)	
6-Segment Multi-level Attention with Pitch-shifting	(N, 6, 128)	
6-Segment Multi-level Attention with Pitch-shifting and Mixup	(N, 6, 128)	$\alpha$ values: 0.1, 0.2, 0.4, 1.0
L-Segment Multi-level Attention with Pitch-shifting and Mixup ( $\alpha = 0.2$ )	(N, L, 128)	L values: 1, 2, 4, 6, 8, 10

Since this thesis uses fine-tuned VGGish model for generating the 128-dimensional features, the input size should be  $(N, var, 128)$  where  $N$  represents the number of audio files and  $var$  represents the variable number of segments for an audio signal.

The original length of the files are retained for LSTMs. For other model structures that need fixed length input, truncating and repeating method are used to make audio files have the same length. So the input size becomes  $(N, L, 128)$ , where  $L$  is the fixed number of segments. Table 9 shows different model structures used in this thesis. The settings of hyper parameters are the same as those showed in Section 5.1.3, except that the numbers of segments for the models are different. Besides, this experiment also tests the effect of pitch-shifting and different  $\alpha$  values for mixup on a 6-segment multi-level attention models.

Unlike Google AudioSet, this challenge does not have a released evaluation set, when the experiments are performed. Thus, for each model structure, 5-fold cross validation training is used for fully using the training data. Then, geometric means of the outputs from all 5 models are taken to get the final classification results.

## 6 Results

This section gives and discusses the implementation results for Google AudioSet and DCASE 2018 Task 2. MAP, AUC, and d-prime scores are used for evaluation of the AudioSet models. MAP@3 is used for the DCASE task. These evaluation metrics are explained in Section 4.3. For the results with little difference, additional paired t-test is applied. Detailed explanations of paired t-test could be found in the footnote <sup>5</sup>. The  $p$ -values are calculated through the paired t-test between two models, values that are smaller than the significance level means the difference is significantly enough. In addition, this section also shows class-wise performance between models with the combination of different levels and number of segments.

### 6.1 Google AudioSet

Table 10 shows the evaluation results for different model structures in Section 5.1.3. DNN-1 is the model trained with only the balanced dataset. DNN-2 has the same structure as DNN-1, but it was trained with the combination of the balanced and unbalanced dataset. The results indicates that utilizing the unbalanced dataset gives clear improvement. The models level-1, level-2, and level-3 are the single-level attention models. The difference between DNN-2 and level-3 is that level-3 has an additional attention layer to model the temporal information, while DNN-2 simply uses the average results for the 10 segments as final result. LSTM performs a little bit better than DNN, and attention models has clear superiority over others.

Table 10: Evaluation on different model structures.

Model	MAP	AUC	d-prime
DNN-1	0.253	0.934	2.133
DNN-2	0.294	0.960	2.482
LSTM	0.307	0.953	2.367
level-1	0.334	0.957	2.422
level-2	<b>0.346</b>	0.961	2.497
level-3	<b>0.347</b>	0.963	2.520
level-12	0.338	0.957	2.423
level-13	0.335	0.959	2.457
level-23	<b>0.345</b>	0.962	2.506
level-123	0.340	0.958	2.439

Since the difference between models with different levels are quite small, this thesis performed significance tests between those models. The average precision (AP) for each class are compared between corresponding two models using the paired t-test. The  $p$ -values between all the combination of the models are calculated based on their statistics. Table 11 shows the results of paired t-tests. Simply speaking, a  $p$ -value represents the conditional probability  $P(\text{model2}|\text{model1})$ . The significance

---

<sup>5</sup><https://www.statisticssolutions.com/manova-analysis-paired-sample-t-test/>

Table 11:  $p$ -values for the significance paired t-test among the models.

	level-1	level-2	level-3	level-12	level-13	level-23	level-123
level-1		0	0	3.9e-4	<b>0.065</b>	3.3e-15	2.2e-7
level-2	0		<b>0.21</b>	4.8e-10	1.1e-16	<b>0.10</b>	6.6e-7
level-3	0	<b>0.21</b>		1.2e-11	1.0e-15	0.018	1.1e-7
level-12	3.9e-4	4.8e-10	1.2e-11		0.019	4.0e-8	0.041
level-13	<b>0.065</b>	1.1e-16	1.0e-15			4.0e-13	7.0e-5
level-23	3.3e-15	<b>0.10</b>	0.018	4.0e-8	4.0e-13		1.1e-4
level-123	2.2e-7	6.6e-7	1.1e-7	0.041	7.0e-5	1.1e-4	

AP scores for models with different levels for given classes.

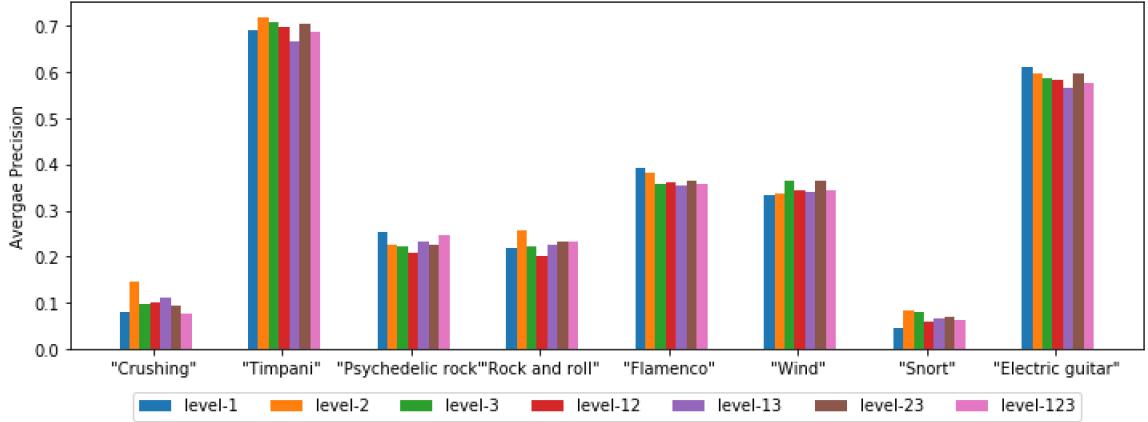


Figure 36: Comparison of AP scores between the models for randomly selected 8 classes

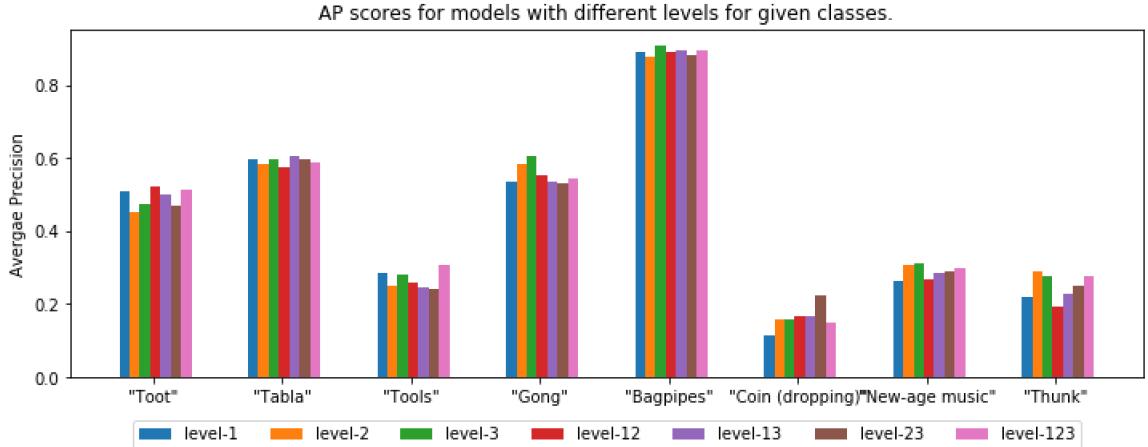


Figure 37: Comparison of AP scores between the models for another randomly selected 8 classes

level is set as 0.05, and any  $p$ -value smaller than 0.05 indicates strong confidence that the two models have different results. The  $p$ -value that is larger than the significance

value, which is colored in gray in the Table 11, means there is not much difference between these two models. For example,  $P(\text{level-2}|\text{level-3}) = 0.21$  means these two models do not have strongly different performance. From the gray cells shown in Table 11, it can be seen that level-2, level-3, and level-23 have relatively similar performance. Level-1 and level-13 also have similar performance.

Randomly 16 classes are selected to see the class-wise performance of these models, which can be seen in Figure 36 and 37. Even some models have better scores, different models might favor different classes. For example, level-1 model has the best result in the classes 'Psychedelic rock', 'Flamenco', and 'Electric guitar', while the classes 'Crushing', 'Timpani', 'Rock and roll' and 'Thunk' favor the level-2 models.

## 6.2 DCASE 2018 TASK2

Firstly, the performance for 4 fine-tuned models with different fine-tuning techniques are tested on 1-segment multi-level models. All three levels are utilized for the final predictions. Since this model is a 1-segment model, no attention mechanism is needed. The results can be seen in Table 12. The original VGGish model for the 527 classes of Google AudioSet has a MAP@3 of 0.606, which is lower than the baseline of 0.704. All the fine-tuned models have outperformed the baseline. 'Balanced Ensemble' means taking the geometric means of probabilities for all the classes from both 'Balanced First Part' and 'Balanced Last Part'. Similarly, 'Unbalanced Ensemble' means taking the geometric means of probabilities for all the classes from both 'Unbalanced First Part' and 'Unbalanced Last Part'. No significant difference has been shown between models with mini-batch balancing and those without. However, taking the geometric mean of all 4 results will give the best score of 0.903. Thus, this strategy is used for following implemented models.

Table 12: The evaluation results for 1-Segment Multi-level Attention model on different fine-tuned VGGish models.

Feature extractor	MAP@3	Feature extractor	MAP@3
Baseline	0.704	Unbalanced First Part	0.858
Original VGGish	0.606	Unbalanced Last Part	0.872
Balanced First Part	0.870	Unbalanced Ensemble	0.892
Balanced Last Part	0.864	All Ensemble	<b>0.903</b>
Balanced Ensemble	0.891		

Table 13 shows the results between different model structures and different random mix factors  $\alpha$ . The details of the models structures and hyper parameters can be seen in Section 5.2.5. Model A has the same setting as 'All Ensemble' from Table 12. Model B with LSTM units modelling the temporal information improves the results from 0.903 to 0.914. The 6-segment multi-level attention models are better than LSTM with our settings. The pitch-shifting for generating a more balanced data set also improves the result a little bit. From the comparison between model E, F, G, and H, it can be seen that mixup with the  $\alpha = 0.2$  has the best performance of 0.936. Overall, the temporal information plays an important role in audio event classification

Table 13: Evaluation results on different model structures and hyper-parameters.

Index	Model Structure	MAP@3
A	1-Segment Multi-level	0.903
B	LSTM	0.914
C	6-Segment Multi-level Attention	0.925
D	C + Pitch shifting Augmentation	0.930
E	D + Mixup ( $\alpha = 0.1$ )	0.930
F	D + Mixup ( $\alpha = 0.2$ )	<b>0.936</b>
G	D + Mixup ( $\alpha = 0.4$ )	0.931
H	D + Mixup ( $\alpha = 1.0$ )	0.930

and attention model has better ability to model the temporal information than LSTM in our settings. Then model F is used as a reference to see the influence from different levels and different number of segments.

Table 14: MAP@3 and  $p$ -values for the significance paired t-test among models of different levels

MAP@3	0.930	0.929	0.934	0.935	0.929	0.928	<b>0.936</b>
	level-1	level-2	level-3	level-12	level-13	level-23	level-123
level-1		0.360	0.108	0.165	0.120	0.323	<b>0.050</b>
level-2	0.360		0.204	0.286	0.238	0.258	<b>0.048</b>
level-3	0.108	0.204		0.400	0.428	0.066	0.169
level-12	0.165	0.286	0.400		0.475	0.098	0.159
level-13	0.120	0.238	0.428	0.475		<b>0.019</b>	0.101
level-23	0.323	0.258	0.066	0.098	<b>0.019</b>		<b>0.012</b>
level-123	<b>0.050</b>	<b>0.048</b>	0.169	0.159	0.101	<b>0.012</b>	

Table 14 shows the MAP@3 scores for models with different levels and the  $p$ -values of corresponding paired t-tests, and cells representing significant difference are marked in gray. The significant level is 0.05. In this task, combining all three levels would give the highest MAP@3 of 0.936. However, the difference is not clear between level-3, level-12, level-13 and level-123.

Table 15 shows the comparison for models with different number of segments. 6-segment model has the best MAP@3 of 0.936 and is significantly different from 1-segment, 8-segment, and 10-segment models. It can also be seen that 4-segment, 6-segment, all-ensemble have relatively similar performance.

As for the class-wise performance, Figure 38 shows the AP@3 of each class for the 6-segment level-123 model. Around half of these classes achieved AP@3 above 0.90, but the class 'Squeak' only has a score around 0.4. Table 16 shows the misclassified classes. It can be seen that many wrongly classified classes sound quite similar to the original classes. For examples, the woodwinds instruments 'Flute', 'Saxophone', and 'Clarinet' are similar classes and the string instruments 'Cello', 'Violin or fiddle', and 'Double bass' quite similar.

Table 15: MAP@3 and  $p$ -values for the significance paired t-test among models of different number of segments

MAP@3	0.917	0.919	0.932	<b>0.936</b>	0.930	0.932	0.932
	1	2	4	6	8	10	all
1		0.083	<b>0.046</b>	<b>0.037</b>	0.208	0.144	<b>0.008</b>
2	0.083		0.114	0.121	0.495	0.476	<b>0.036</b>
4	<b>0.046</b>	0.114		0.480	0.061	0.087	0.485
6	<b>0.037</b>	0.121	0.480		<b>0.044</b>	<b>0.043</b>	0.494
8	0.208	0.495	0.061	<b>0.044</b>		0.466	0.087
10	0.144	0.476	0.087	<b>0.043</b>	0.466		<b>0.025</b>
all	<b>0.008</b>	<b>0.036</b>	0.485	0.494	0.087	<b>0.025</b>	

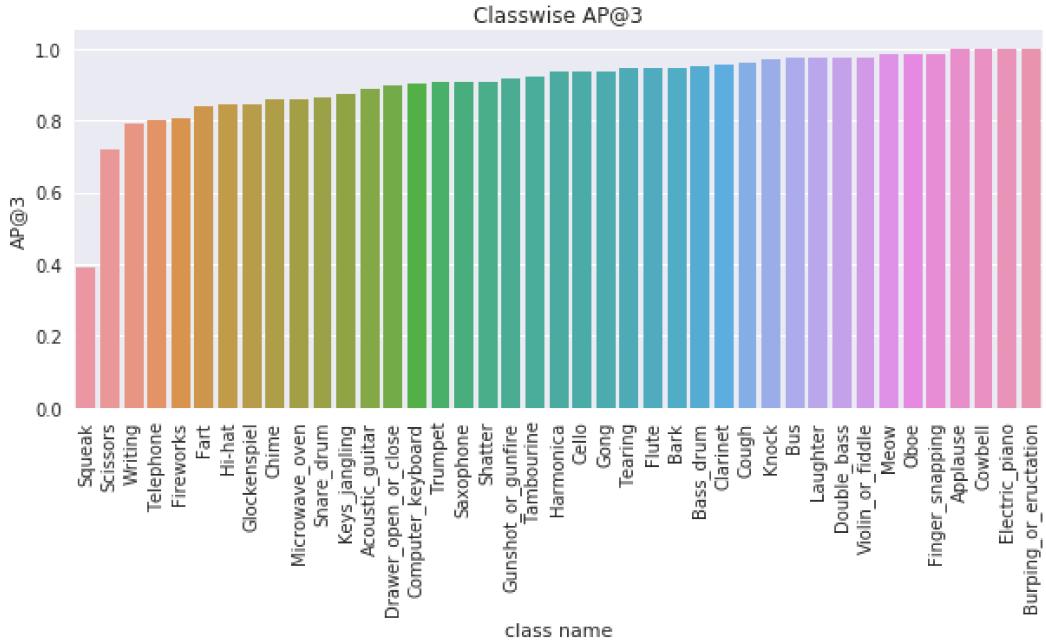


Figure 38: Distribution of the samples along the 41 classes.

This thesis also chooses the top 4 classes and the bottom 4 classes for class-wise comparison for models with different levels or number of segments. The comparison for models with different levels can be seen in Figure 39. The top 4 classes all have the same AP@3 of 1.0, and the models performs differently for the lowest classes. Figure 40 shows comparison for models with different number of segments. The results also show that different classes might favor different length of segments, thus also suggesting that incorporating the multi-scale features for future experiments.

Table 16: This table represents wrongly classified classes with more than 5% probability.

Original Class	AP@3	wrongly classified as(more than 5%)
Squeak	0.391	Oboe, Fart
Scissors	0.720	Tearing, Finger snapping
Writing	0.793	Fireworks, Tearing
Fireworks	0.807	Gunshot or gunfire, Tearing
Hi-Hat	0.842	Tambourine, Snare drum
Glockenspiel	0.845	Chime
Chime	0.856	Glockenspiel
Microwave oven	0.856	Drawer open or close
Snare drum	0.863	Hi-hat, Gong
Keys jangling	0.875	Shatter
Drawer open or close	0.896	Knock
Trumpet	0.905	Harmonica
Saxophone	0.906	Clarinet
Shatter	0.908	Tambourine, Keys jangling
Tambourine	0.921	Hi-hat
Harmonica	0.934	Clarinet
Cello	0.935	Violin or fiddle, Double bass
Flute	0.945	Clarinet
Cough	0.961	Burping or eructation

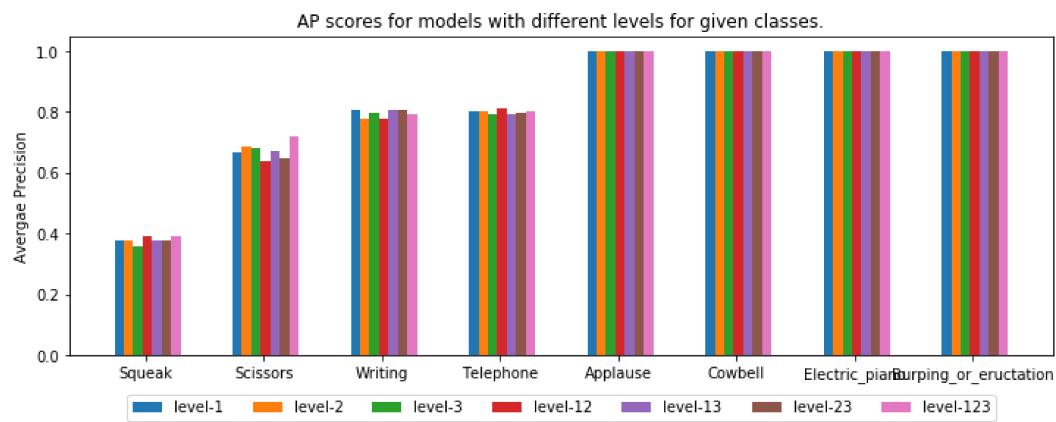


Figure 39: Comparison of AP@3 between models with different levels for 8 classes.

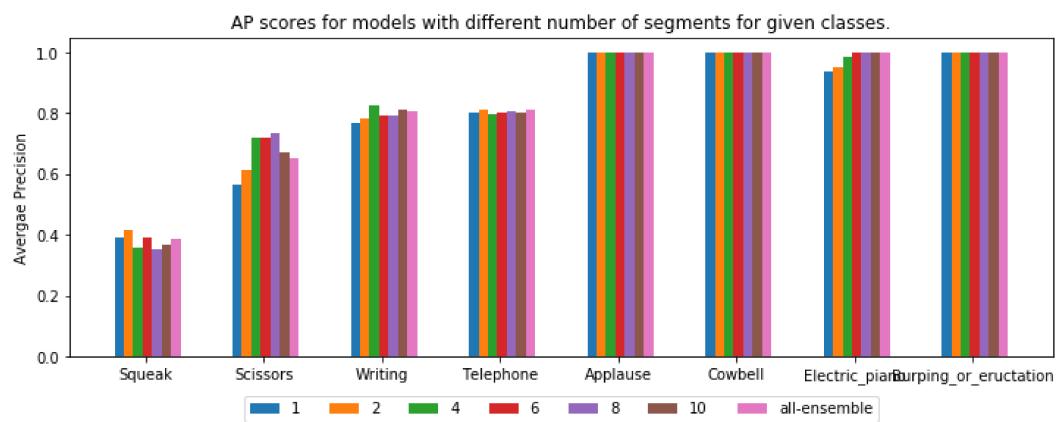


Figure 40: Comparison of AP@3 between models with different number of segments for 8 classes.

## 7 Conclusion

The goal of this thesis was to investigate the potential of deep learning methods to provide a general solution to the task of audio event classification. This thesis discussed various aspects of audio event classification, including applications, motivations, challenges, and development of deep learning. In addition, the research material and methods were also addressed in detail. Different neural network structures, including DNN, LSTM, and multi-level attention models, were designed for both Google AudioSet and the dataset for DCASE 2018 Task 2. The results of AudioSet exceeded the Google baseline, and the system submitted for the DCASE task was ranked among the top 8% in terms of results on the public leaderboard. AudioSet provided a deep CNN-like model for generating 128-dimensional semantically meaningful features from audio log-mel spectrograms with a frame length of 960ms. This process compressed the audio information into a more compact version, which made training the neural network much easier. These compact features achieved effective performance with various neural network structures in the study’s experiments, which demonstrated the effectiveness of CNN-like models in modeling the frequency characteristics of audio signals. The thesis chose to fine-tune the CNN-like model for the DCASE task because its effectiveness in generating compact 128-dimensional features had already been demonstrated. The DCASE task does not provide the 128-dimensional features directly but offers original soundtracks instead. Thus, the thesis firstly preprocessed the dataset by removing the silent parts, repeating and splitting the audio into a fixed length, pitch shifting augmentation, and by utilizing a mixup technique. The processed soundtracks then were fed into the fine-tuned model for generating a series of 128-dimensional features, which then became the input for different neural network structures.

The results showed that a neural network with multi-level attention structures performs the best with our settings for both tasks, indicating that attention structures are useful in modeling the dynamic information of sound. Even though multi-level attention models with a certain combination of levels and segment numbers showed better results than other configurations, classwise performance can differ when the configuration changes. The results for the DCASE task also indicated that fine-tuning the AudioSet features works well for other audio event classification tasks. However, the CNN-like model used for generating the 128-dimensional features was only for audio with a frame length of 960ms. This 960ms scale is quite limited for some types of sound. In addition, this thesis failed to find an optimal ensemble technique for combining the strength of different models. Therefore, further studies could focus on training their own CNN models for feature generation with scales other than 960ms, combining multi-scale and multi-level features for more detailed representation of the audio, and finding better ensemble techniques for utilizing strengths from different models. Besides, dealing with audio signals with noisy labels should also be considered when utilizing a dataset with various types of recordings. Possible approaches include pseudo-labelling noisy audio signals with a model trained with clean data and applying weighed loss so that the noisy signals contribute less to loss calculation. Furthermore, future studies should also concentrate on fusion with

other multimodal features, such as combining with ecological data for analyzing the amount of species in certain environments, building more robust public surveillance systems by providing audio events recognition results for security cameras, and helping analyze multimedia content along with images and speech recognition results.

## References

- [1] Hagai Aronowitz. Segmental modeling for audio segmentation. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–393, Honolulu, HI, USA, April 2007.
- [2] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D Plumbley. Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32(3):16–34, May 2015.
- [3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.
- [4] Victor Bisot, Romain Serizel, Slim Essid, and Gael Richard. Supervised nonnegative matrix factorization for acoustic scene classification. In *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, Budapest, Hungary, September 2016.
- [5] Forrest Briggs, Raviv Raich, and Xiaoli Z Fern. Audio classification of bird species: a statistical manifold approach. In *2009 Ninth IEEE International Conference on Data Mining(ICDM)*, pages 51–60, Miami, Florida, December 2009.
- [6] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark B. Sandler. A tutorial on deep learning for music information retrieval. *CoRR*, abs/1709.04396, 2017.
- [7] S. Chu, M. Mataric, C. Kuo, and S. Narayanan. Where am I? Scene recognition for mobile robots using audio features. In *2006 IEEE International Conference on Multimedia and Expo(ICME)*, pages 885–888, Toronto, ON, Canada, July 2006.
- [8] Courtenay V Cotton and Daniel PW Ellis. Spectral vs. spectro-temporal features for acoustic event detection. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 69–72, New Paltz, NY, USA, October 2011.
- [9] Christophe Couvreur, Vincent Fontaine, Paul Gaunard, and Corine Ginette Mubikangiey. Automatic classification of environmental noise events by hidden Markov models. *Applied Acoustics*, 54(3):187–206, 1998.
- [10] Chris Donahue, Julian McAuley, and Miller Puckette. Synthesizing audio with generative adversarial networks. *CoRR*, abs/1802.04208, 2018.
- [11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

- [12] Benjamin Elizalde, Howard Lei, Gerald Friedland, and Nils Peters. An i-vector based approach for audio scene detection. *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [13] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel PW Ellis, Xavier Favory, Jordi Pons, and Xavier Serra. General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline. *arXiv preprint arXiv:1807.09902*, 2018.
- [14] Eduardo Fonseca, Jordi Pons Puig, Xavier Favory, Frederic Font Corbera, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. Freesound datasets: a platform for the creation of open audio datasets. In *Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. Proceedings of the 18th ISMIR Conference*, pages 486–93, Suzhou, China, October 2017.
- [15] Jonathan T Foote. Content-based retrieval of music and audio. In *Multimedia Storage and Archiving Systems II*, volume 3229, pages 138–148. International Society for Optics and Photonics, 1997.
- [16] Jonas Gehring and Michael Auli. A novel approach to neural machine translation. <https://code.fb.com/ml-applications/a-novel-approach-to-neural-machine-translation/>, 2017.
- [17] Jürgen T Geiger, Bjoern Schuller, and Gerhard Rigoll. Recognising acoustic scenes with large-scale audio feature extraction and SVM. *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [18] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780, New Orleans, LA, USA, March 2017.
- [19] Jort F Gemmeke, Lode Vuggen, Peter Karsmakers, Bart Vanrumste, et al. An exemplar-based NMF approach to audio event detection. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, New Paltz, NY, USA, October 2013.
- [20] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.
- [21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

- [23] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868, May 2009.
- [24] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, Vancouver, BC, Canada, May 2013.
- [25] Guodong Guo and Stan Z Li. Content-based audio classification and retrieval by support vector machines. *IEEE Transactions on Neural Networks*, 14(1):209–215, 2003.
- [26] Yoonchang Han, Jeongsoo Park, and Kyogu Lee. Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification. Munich, Germany, November 2017.
- [27] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.
- [28] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.
- [29] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.
- [30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.
- [31] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153, Kyoto, Japan, October 2009. IEEE.
- [32] Suyoun Kim, Bhiksha Raj, and Ian Lane. Environmental noise embeddings for robust speech recognition. *CoRR*, abs/1601.02553, 2016.
- [33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [34] Qiuqiang Kong, Yong Xu, Wenwu Wang, and Mark D Plumley. Audio set classification with attention model: A probabilistic perspective. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, April 2017.

- [35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [36] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.
- [37] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.
- [38] Jongpil Lee and Juhan Nam. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE Signal Processing Letters*, 24(8):1208–1212, 2017.
- [39] Zhu Liu, Jincheng Huang, Yao Wang, and Tsuhan Chen. Audio feature extraction and analysis for scene classification. In *Proceedings of First Signal Processing Society Workshop on Multimedia Signal Processing*, pages 343–348, Princeton, NJ, USA, June 1997. IEEE.
- [40] Richard F Lyon. Machine Hearing: An Emerging Field [Exploratory DSP]. *IEEE Signal Processing Magazine*, 27(5):131–139, 2010.
- [41] Ling Ma, Ben Milner, and Dan Smith. Acoustic environment classification. *ACM Trans. Speech Language Processing*, 3(2):1–22, July 2006.
- [42] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.
- [43] Xianglai Meng, Biao Leng, and Guanglu Song. A multi-level weighted representation for person re-identification. In *Artificial Neural Networks and Machine Learning – ICANN 2017*, pages 80–88, Cham, 2017.
- [44] Annamaria Mesaros, Toni Heittola, Antti Eronen, and Tuomas Virtanen. Acoustic event detection in real life recordings. In *2010 18th European Signal Processing Conference*, pages 1267–1271, Aalborg, Denmark, April 2010.
- [45] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.
- [46] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep belief networks for phone recognition. In *Nips workshop on deep learning for speech recognition and related applications*, volume 1, page 39, Vancouver, Canada, 2009.
- [47] Alvin W. Moore and James W. Jorgenson. Median filtering for removal of low-frequency background drift. *Analytical Chemistry*, 65(2):188–191, 1993. PMID: 8430895.

- [48] Seongkyu Mun, Sangwook Park, David K Han, and Hanseok Ko. Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane. Technical report, Munich, Germany, November 2017.
- [49] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, Haifa, Israel, 2010.
- [50] Radford M Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113, 1992.
- [51] Takanobu Nishiura, Satoshi Nakamura, Kazuhiro Miki, and Kiyohiro Shikano. Environmental sound source identification based on hidden markov model for robust speech recognition. In *8th European Conference on Speech Communication and Technology*, Geneva, Switzerland, September 2003.
- [52] Hiroshi G Okuno, Tetsuya Ogata, and Kazunori Komatani. Computational auditory scene analysis and its application to robot audition: Five years experience. In *2008 Hands-Free Speech Communication and Microphone Arrays*, pages 124–127, Trento, Italy, May 2008.
- [53] Roy D Patterson, Mike H Allerhand, and Christian Giguere. Time-domain modeling of peripheral auditory processing: A modular architecture and a software platform. *The Journal of the Acoustical Society of America*, 98(4):1890–1894, 1995.
- [54] Ya-Ti Peng, Ching-Yung Lin, Ming-Ting Sun, and Kun-Cheng Tsai. Healthcare audio event classification using hidden Markov models and hierarchical hidden Markov models. In *2009 IEEE International Conference on Multimedia and Expo*, pages 1218–1221, New York, NY, USA, July 2009.
- [55] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.
- [56] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880, Montreal, QC, Canada, June 2009.
- [57] Alain Rakotomamonjy and Gilles Gasso. Histogram of gradients of time-frequency representations for audio scene classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):142–153, 2015.
- [58] Gerard Roma, Waldo Nogueira, Perfecto Herrera, and Roc de Boronat. Recurrence quantification analysis features for auditory scene classification. *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2, 2013.

- [59] F. Rosenblatt. *The Perceptron, a Perceiving and Recognizing Automaton Project Para.* Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957.
- [60] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.
- [61] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.
- [62] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [63] Stanley Smith Stevens, John Volkman, and Edwin B Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.
- [64] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.
- [65] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop*, page 125, Sunnyvale, CA, USA, 2016.
- [66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
- [67] Emmanuel Vincent, Shoko Araki, Fabian Theis, Guido Nolte, Pau Bofill, Hiroshi Sawada, Alexey Ozerov, Vikrham Gowreesunker, Dominik Lutter, and Ngoc QK Duong. The signal separation evaluation campaign (2007–2010): Achievements and remaining challenges. *Signal Processing*, 92(8):1928–1936, 2012.
- [68] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. In *Readings in Speech Recognition*, pages 393–404. Elsevier, 1990.
- [69] PJ Werbos. New tools for prediction and analysis in the behavioral science. *Ph. D. Dissertation, Harvard University*, 1974.
- [70] Erling Wold, Thom Blum, Douglas Keislar, and James Wheaten. Content-based classification, search, and retrieval of audio. *IEEE MultiMedia*, 3(3):27–36, 1996.

- [71] Zixiang Xiong, Regunathan Radhakrishnan, and Ajay Divakaran. Generation of sports highlights using motion activity in combination with a common audio feature extraction framework. In *Proceedings 2003 International Conference on Image Processing*, pages 5–8, Barcelona, Spain, September 2003.
- [72] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, Lille, France, July 2015.
- [73] Changsong Yu, Karim Said Barsim, Qiuqiang Kong, and Bin Yang. Multi-level attention model for weakly supervised audio classification. *arXiv preprint arXiv:1803.02353*, March.
- [74] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.