Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Sound Event Detection with Deep Neural Networks

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Seyedeh-Anahid Naghibzadeh-Jalali
Matrikelnummer E1229620

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.univ.Prof. Dr. Andreas Rauber
Mitwirkung: Dipl. Ing. Alexander Schindler

Wien, 28. Februar 2018

Seyedeh-Anahid
Naghibzadeh-Jalali

Andreas Rauber

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Sound Event Detection with Deep Neural Networks

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Informatik

by

Seyedeh-Anahid Naghibzadeh-Jalali
Registration Number E1229620

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.univ.Prof. Dr. Andreas Rauber
Assistance: Dipl. Ing. Alexander Schindler

Vienna, 28th February, 2018

Seyedeh-Anahid Naghibzadeh-Jalali

Andreas Rauber

Erklärung zur Verfassung der Arbeit

Seyedeh-Anahid Naghibzadeh-Jalali
Address

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 28. Februar 2018

Seyedeh-Anahid
Naghibzadeh-Jalali

Acknowledgements

At first, I would like to thank my thesis advisor, Prof. Andreas Rauber, who accepted my thesis proposal and made this work possible by offering his advice and supervision. Also my teacher, Rudolf Mayer, who helped me through with his constructive guidance. Furthermore, I would like to express my gratitude to Alexander Schindler for the considerable amount of time that he invested in answering my early questions, as well as for the interesting discussions and organizational support. Moreover, I would like to thank the Austrian Institute of Technology for their technical support and efforts which gave me a great platform to work and write my thesis. Finally, I would like to thank my family, who have always stood by me and dealt with my absence from many family occasions with a smile and never-ending support.

Kurzfassung

Acoustic Sound Event Detection (SED) wurde in den letzten Jahren intensiv erforscht und gilt als ein aufstrebendes Thema in der Computational Auditory Scene Analysis (CASA) Forschung. Das menschliche Gehirn hat die Fähigkeit, mehrere Töne gleichzeitig zu erkennen und das Hintergrundrauschen auszublenden, um sich auf ein selektives Ereignis zu konzentrieren. Dieses Phänomen ist auch als Cocktailparty-Effekt bekannt. Ziel von SED ist es, Systeme zu entwickeln, die es ermöglichen, Ereignisse in einer Geräuschkulisse zu erkennen. Daher werden diese so trainiert, dass sie Klangereignisse anhand der Audiosignale zuordnen. Ein Sound-Ereignis ist eine Bezeichnung, die von Menschen verwendet wird, um ein Ereignis in einer Audiosequenz zu beschreiben und zu identifizieren. Diese Arbeit konzentriert sich auf die beiden Teilaufgaben von DCASE 2017 Challenge, der seltenen Erkennung von Sound Events, die die Erkennung des Auftretens und Auslösens eines Sound Events in einem Audio und die Erkennung von Sound Event in Umgebungen mit mehreren Quellen darstellt. Die vorgeschlagene Methodik basiert auf Künstlichen Neuronalen Netzen (KNN), die eine robuste Leistung bei komplizierten Aufgaben wie Spracherkennung, Verarbeitung natürlicher Sprache und Bildklassifizierung gezeigt haben. Verschiedene Audioeingangsdarstellungen, wie z. B. konstante Q-Transformation, Mel Frequency Cepstral Coefficient (MFCC) und Mel Spectrogram, werden ebenfalls getestet, wobei sich das Mel-Spektrogramm als die bessere Darstellung unter den genannten erwies. Die in dieser Arbeit untersuchten ANN-Architekturen sind das Recurrent Neural Network (RNN) und seine Erweiterung, Long Short Term Memory (LSTM) und das Convolutional Neural Network (CNN). Die RNN-Architektur wurde wegen ihrer Fähigkeit gewählt, das zeitliche Verhalten ihrer Eingänge und ihrer CNN-Architektur zu erfassen, da sie die Funktionen auf hoher Ebene durch ihre Faltungsschichten erlernen kann. Um diese Architekturen zu trainieren, wurden verschiedene Hyperparameter getestet. Das Verhalten jedes Modells wird analysiert und ihre Ergebnisse verglichen. Um die Konvergenz und Generalisierbarkeit der Modelle zu verbessern, wurde eine Data Augmentation durchgeführt, und die Dropout-Technik wurde angewendet, um Over-Fitting zu vermeiden. Um die Leistung dieser Modelle zu bewerten, wurden zwei Datensätze aus der DCASE 2017 Challenge verwendet. Im Rare Sound Event Detection-Dataset werden die Audio Events synthetisch in die Audio-Aufzeichnungen eingefügt, bei denen jedes Ereignis nur einmal oder gar nicht auftrat. Der zweite Audio-Datensatz enthält Umweltaufnahmen mit unterschiedlichen Längen von 3 Minuten bis zu 5 Minuten, welche manuell annotiert wurden. Die experimentellen Ergebnisse dieser Arbeit zeigen die Robustheit von tiefen

neuronalen Netzwerken im Vergleich zum konventionellen Multilayer Perzeptron, das als Referenzsystem betrachtet wird. Unter Verwendung dieser Architekturen wurde eine Fehlerrate von 0,30 erreicht, die eine Verbesserung von 43.39% im Vergleich zu einer Fehlerrate von 0,53 aufweist, die durch das Referenzsystem der Rare Sound Event Detection Task erreicht wurde. Bei Real Life Sound Event Detection erreichte die beste Performance eine Fehlerrate von 0,77, die eine Verbesserung von 17,20% im Vergleich zu der Fehlerrate von 0,93 aufweist, die durch das Referenzsystem erreicht wurde.

Abstract

Acoustic Sound Event Detection (SED) has been extensively studies over the past years and is considered an emerging topic in Computational Auditory Scene Analysis (CASA) research. The human brain has the ability to recognize multiple sounds at once, reducing the background noise to focus on a selective event. This phenomenon is referred to the cocktail party effect. SED aims to implement systems, which enables them to detect any events occurring in the environmental sound in their surroundings. Therefore, those are trained in such a way that they classify sound events in the input audio signals. A Sound event is a label used by humans to describe and identify an event in an audio sequence. This thesis focuses on the two subtasks of DCASE 2017 Challenge, the rare sound event detection which is the identification of the onset and offset of a sound event in an audio, and the sound event detection in multi source environments. The proposed methodology used for this thesis is based on Artificial Neural Networks (ANNs) which have shown robust performance on complicated tasks such as Speech Recognition, Natural Language Processing and Image Classification. Different audio input representations such as Constant Q-transform, Mel Frequency Cepstral Coefficient (MFCC) and Mel Spectrogram are also tested from which Mel-Spectrogram proved to be the better representation among the ones mentioned. The ANN architectures studied in this work are the Recurrent Neural Network (RNN) and its extension, Long Short Term Memory (LSTM) and the Convolutional Neural Network (CNN). RNN architecture was chosen because of its ability to capture the temporal behavior of its inputs and CNN architecture because of its ability to learn the high level features through its convolutional layers. To train these architectures, different hyper parameters were tested. The behavior of each model is analyzed and their results compared. To improve convergence and generalizability of the models, data augmentation was performed and also, the dropout technique was applied to avoid over fitting. To evaluate the performance of these models, two datasets provided by the DCASE 2017 challenge were used. The Rare Sound Event Detection dataset added the audio events synthetically in the audio recordings where each event appeared only once or not at all. The second audio dataset has environmental recordings with different lengths from 3 minutes up to 5 minutes which were manually labeled by the providers. The experimental results of this thesis show the robustness of deep neural networks in comparison with the conventional Multilayer Perceptron, which is considered as the baseline system. Using these architectures, an error rate of 0.30 was achieved which has 43.39% improvement compared to 0.53 error rate achieved by the

baseline system on the Rare Sound Event Detection Task. On Real Life Sound Event Detection the best performance achieved 0.77 error rate which has 17.20% improvement in comparison with the 0.93 error rate achieved by the baseline system.

Contents

Kurzfassung **ix**

Abstract **xi**

Contents **xiii**

1 Introduction **1**
1.1 Problem Statement 2
1.2 Contribution of this thesis 5
1.3 Outline 7

2 State Of The Art **9**
2.1 Data Augmentation 10
2.2 Architecture 12

3 Theoretical Contribution **15**
3.1 Audio Features 15
3.2 Data Augmentation 24
3.3 Learning Process 28

4 Experimental Results **43**
4.1 Datasets 43
4.2 Framework 46
4.3 Hardware Specifications 47
4.4 Experiment Results 47

5 Conclusion and Future Work **91**
5.1 Conclusion 91
5.2 Future Work 92

List of Figures **93**

List of Tables **95**

Bibliography **97**

CHAPTER 1

Introduction

A sound event is a label used by humans to describe and identify an event in an audio sequence. These labels give people a clearance to understand and associate each event with their previously known events. Sound Event Detection (SED) also known as Acoustic Event Detection (AED) is an important task for computational auditory scene analysis (CASA)[MTX$^+$16]. The input of SED systems is a continuous signals. The system learns the appropriate features of the events, in order to recognize them within the given signals. Perhaps one of the questions which rises above at this point is the motivation behind such studies. Humans use their senses such as sight, touch, sound sense in order to understand the surroundings of them. To simulate such behavior, systems, applications and artificial agents require the same understanding. Therefore to answer the question, the need to detect sound events is to understand the situations, using acoustic signals.
SED could be employed in applications such as information retrieval [BPT$^+$09], military and automated surveillance applications [KŁC11] and also smart home systems [vHA09]. Adding listening ability to embedded systems allows them to be more aware of their environment [CNK09, CNKM06]. Automatically detecting the events of interest in industrial and surveillance systems as well as smart homes also attracts attention to applications which detect acoustic sounds and events [HMS05]. To detect a single event at the same time, approaches such as Mel Frequency Cepstral Coefficients (MFCC) and Hidden Markov Models (HMM) were used which are known as the conventional approaches [HMEV13]. However, in real life environments, it is more likely that more than a single event occur simultaneously in a signal. Therefore detecting these overlapping events is considered as a challenge and the mentioned methods may not be suitable. A solution to that was proposed in [CKBK16], using polyphonic SED system which aims to detect multiple sound events in the same time instance of the sound data. Using the development dataset from second task of DCASE 2017 challenge, the focus of the first phase of this work will be on using different approaches in order to detect each sound event more accurate and classify them into the following predefined target classes: Sound

of a crying baby, Gunshot, Breaking glasses. For phase two, the development dataset of the third task is used which brings the focus of this thesis into multilabel classification and event detection with overlapping sound events. The predefined classes in this phase are: Brakes squeaking, Car, Children, Large vehicle, People speaking, People walking. Considering deep learning methods have shown good performance in many different applications such as image and speech recognition [MHV16b, HWT+16] , the method used for this project in order to achieve the goal is chosen to be an architecture (or a hybrid system) of deep learning methods. Another issue to mention here is the data. Classifiers tend to perform better when the data is large enough to train. A common strategy adopted to enlarge the amount of the training data, increase the robustness of the model and avoiding the problem of over fitting, is data augmentation. Therefore this strategy will also be used in order to enlarge the training data and improve the performance of the models. In this chapter, first the problem is formally defined. Next the contribution of this project is briefly introduced, followed by the chapter on state of art and the analysis of the existing approaches. Afterwards, the methods used in this work to achieve more accurate detection is described in more depth. Eventually, in order to achieve a final conclusion, the obtained results is presented and discussed.

1.1 Problem Statement

Sound event detection task tries to address the problem of detecting sound events within overlapping signals. These events are referred to the segments of audio which human listeners constantly label and distinguish in an acoustic environment [APP+16]. The automatic SED task is to recognize the sound event(s) in a continuous audio signal. This task is one of the emerging topics of CASA research (Computational Audio Signal Analysis) where the researchers try to replicate the phenomenon effect of human brain, the cocktail party effect [PT17], which is the ability of listening to multiple sound and reducing the background noise. Applications of this task are used in militarily [HMEV13], smart homes [HMS05], embedded systems with listening capability to make them aware of their surrounding environment [CNKM06].

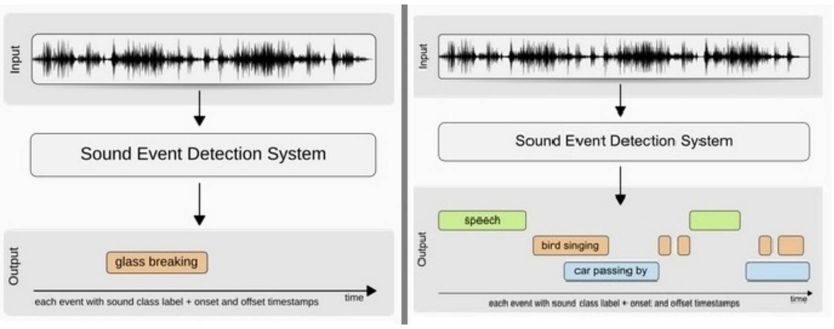

Input
Sound Event Detection System
Output
glass breaking
each event with sound class label + onset and offset timestamps
time
Input
Sound Event Detection System
Output
speech
bird singing
car passing by
each event with sound class label + onset and offset timestamps
time

Figure 1.1: An overview of a sound event detection[1]

In figure 1.1, an overview of the sound event detection is depicted. Given an input signal, SED system will output the label of the event occurring in the given input as well as the onset and offset of the event's time frames (time steps). This problem can be categorized into two subcategories; first is to detect only one specific event (a rare event) within the signal which is illustrated on the left side of the figure. This category addresses the binary detection issue which was covered by the second task of DCASE 2017. The other subcategory is to detect multiple events which can happen at the same time. The multiple overlapping sound event detection addresses the multi-label classification problem and is shown on the right side of the figure. SED systems try to locate the starting and label the sound event classes present in a polyphonic audio signal. [PHH^{+}17] formally defines the SED problem as follows; by extracting frame-level sound features for each time frame t in the audio time series, a feature vector $xt \in R^F$ is then obtained. F is the number of features per frame. Afterwards, the probabilities $p(y_t(k)|x_t, \theta)$ for classification phase needs to be estimated. $k = 1, .., K$ denoted the events in the frame t and θ refers to the set of the model parameters. Finally, using a specified threshold, these event probabilities are binarized, namely, if the label k occurs in the frame t, the output $y_t(k)$ will be set to 1 and otherwise 0. By breaking down SED into smaller steps, the factors which are required to be considered for more accurate detection.

Scientific Question 1.1: SED Problem Statement

INSTANCE:
Given two different sets of recorded audio as input and the ground truth for one of the sets, together with a sound event detection system.

PROBLEM:
How to determine the starting point of the rare event occurring in each input stream and classify it into the correct predefined class with granularity of 500 ms.

Scientific question 1.1 describes this project's focus. However, as mentioned before, SED task needs to be broken down to number of subtasks.

Scientific Question 1.2: Input Representation selection

INSTANCE:
Given a set of recorded audio as input and an approach for input representation such as mel spectrograms, mfcc, ...

PROBLEM:
Which representation is most effective to use as the input of deep learning model in order to detect the audio event.

---

[1] http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/

The first subtask addresses the issue of input representation. The inputs of the deep learning classifiers are feature vectors and the classifiers will be trained over these selected feature. In order to have a robust detection, these selected features are considered as very important factors. Conventional SED and Acoustic Speech Recognition (ASR) used features such as Mel-Frequency-Cepstral-Coefficients (MFCC), Frequency Filtering (FF), Linear Prediction Cepstral (LPC) and Hidden Markov Models (HMMs) which were mostly for monophonic systems to detect a single event at a time. However, in real life, the occurrences of multiple events simultaneously is more likely to happen. In the work of [PHV16] the proposed polyphonic SED system, MFCC features were selected and HMM was the chosen classifier. A polyphonic SED system has the focus on detecting concurrent occurrences of events. In other works, Generalized Hough Transform (GHT) was selected in order to detect overlapping acoustic events [DTC13] and Nonnegative Matrix Factorization based (NFM-based) approach was applied for source separation and detecting the events occurring in each audio stream [HMVG13]. Modeling overlapping sound events with the traditional DNN also showed a good performance in the work of [CHHV15]. For this work, features such as MFCC, Short Time Fourier Transformation (STFT) and Constant Q Transform (CQT) will be implemented and compared. The purpose of this subtask is to observe and analyze the differences between each feature in accuracy of the detection task in order to detect overlapping/non-overlapping events.

Scientific Question 1.3: Generalizability of the Model

INSTANCE:
Given a sets of recorded audio as input and a deep learning architecture for event detection.

PROBLEM:
Can the generalizability of model be improved by the solutions such as data augmentation and dropout?

Question 1.3 formally describes the problem of generalizing the model. Model generalization refers to the high classification performance of the model on any type of data. In many applications, deep learning classifiers have shown good performance. For that, some factors such as data augmentation and dropout are pointed at. Having large training data allows these classifiers to learn from the variations of samples under the same labels in the dataset. For this matter, the training data from each class should be sufficient enough to cover its labeled class variations, otherwise using poor data will lead to a weak generalization ability of classifiers. In order to overcome such issue data augmentation methods have been introduced. There exist numerous approaches to augment data. Applying transformations and adding noise to existing data is counted as the simplest approach of data augmentation.Also employing dimensional reduction and imputation methods in order to increase the number of instances in sparse areas of the dataset are counted as another ways of augmenting data. As more advanced approaches simulating the data based on evolutionary and dynamic systems can be mentioned. To mention some: approaches such as Vocal Tract Length Perturbation (VTLP) and Equalized Mixture Data Augmentation (EMDA) which have shown improvements for the sound event detection task was implemented and tested in [CGK15]. This project will implement the applied data augmentation methods using the librosa library. More detailed of the used approaches and their performances are explained in theoretical contribution part of this thesis. The last subtask of this thesis, mentioned in question 4, is the implementation of different deep learning approaches such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNN), CNN-RNN, the hybrid approach Bidirectional Long-Short-Term-Memory (BLSTM) and Hidden Markov Models (BLSTM-HMM). The behavior of each model is analyzed and compared with other implemented approaches and a conclusion will be drawn.

Scientific Question 1.4: Classifier Selection

INSTANCE:
Given a feature vector from the recorded audio dataset and a deep neural network model.

PROBLEM:
Does this model have a better performance than the baseline system? How does RNN model and its extensions perform compare to Multilayer perceptron?

In this project, we tried to analyze different proposed approaches and argue which of the architecture, using which features lead to more accurate detection of the sound events. Reference to the questions 2,3 and 5, the main interest of these work, given the DCASE2017 rare sound event dataset, is to utilizing different classification methods and compare them in order to detect the best matching deep learning architecture for the discussed questions.

1.2 Contribution of this thesis

The key contribution of this thesis is to apply deep neural network on the sound event detection task and use solutions such as data augmentation, dropout technique in order to generalize the deep model as well as trying to explain the behavior and performance of the model buy changing its number of inputs, extracted features and parameters. A numbers of factors (i.e. input representation, model parameters, ..) need to be considered for the task of sound event detection and classification. As mentioned earlier, we consider the task in different steps which each can be seen as a sub-task to be further analyzed. Here, the first problem to address is how to augment the available data in order to create more instances for each class, for the purpose of model training and improvement in detection and classification process. What manipulation on the instances is mostly needed and whether the chosen data augmentation approaches or even the combination of them will improve the learning rate.

Moreover, the representation of each audio segments should be strong enough to capture the different energy levels. Which input representation captures the features and characteristics of the sound events and is most effective for the defined task. Also reducing the background noise which enhances the event's features, leading to more accurate classification. The importance of this work is to comprehend the environment and events arising within surrounding areas using the extracted information from the received signals. multiple architectures and the training process with manipulating their parameters in order to analyze the behavior of each model on the task is another step of this procedure. Following sections show the operations which are performed in each of these steps.

Training Instances

As mentioned earlier, the robustness of deep models highly depends on the number of input data. Therefore, one of the contributions of this thesis is to apply data augmentation techniques such as pitch shifting and time stretching on audio signals to create more training samples for the model and study the model's performance on these techniques. For this purpose, two different audio scenes provided by Tampere university for TUT Acoustic scene 2017 challenge are used. Two provided datasets are: Rare-sound event detection dataset and real life sound event detection dataset.

The rare-sound event detection dataset comprises of isolated sound events for each target class. The background sound served in each audio is the everyday acoustic scene recordings. To evaluate the model's performance, an evaluation dataset is also provided which comprises of 250 recordings for each classes.

The real life sound event detection dataset is only the street recordings, where the events are sound of cars and people. Unlike the rare-sound event detection dataset, this dataset is not synthesized and is an actual recording of the street sound. Therefore, the quality of the recordings for this dataset is much lower than the first dataset.

Features

As mentioned before, the challenging part of the task is to determine the onset of each event in the given audio inputs; for that, conventional approaches such as HMM and MFCC need to be modified to accurately detect the sound events. Also other features such as CQT and Mel-band energy are applied and analyzed. These features help the classifiers to deal with the variation of instances and try to improve the accuracy of detection.We thus provide, as a result of this thesis, insights into these input representation techniques and evaluation of the effectiveness of Mel-band energies and CQTs in comparison to conventional MFCCs.

Classifiers

Many approaches exist to evaluate the outcome of different classifiers applied to the SED task. The chosen machine learning approach for this work is different deep learning architectures. Through SED challenge, when facing different level of complexity such as noise or polyphony, these methods will be studied and compared. Previous works showed the high performance of architectures such as Recurrent Neural Networks [PHV16] and Convolutional Neural Network (CNN) [CKBK16] separately utilized for acoustic scene classification and audio tagging. RNN has feedback structure and in comparison to feed-forward layered neural network, earlier time information can be propagated forward to the current time which in case of event detection is very important (the beginning and ending of an event will be detected). CNN is a given raw input signal which is divided into segments called frames, and outputs a score in each predefined class. This architecture is the combination of several filter stages and a max-pooling layer. BLSTM-HMM hybrid system for polyphonic sound detection is another approach which had outperformed the conventional RNN [HWT$^+$16]. Other contribution of this thesis is to investigate these different options and analyze the behavior of each approach on the same dataset and to be able to explain and justify the model's prediction.

Classifier Parameters

Finding the appropriate set of parameters in order to achieve more accurate result from the employed method is another important aspect of the detection and classification task. These parameters have the capability of noticeably varying the behavior of the mentioned algorithms. Therefore, parameters for each algorithm will be altered and the results of corresponding cases will be shown in the future chapters.

1.3 Outline

In the following chapter, we discuss the state of art for the related issues and different applied approaches in those works are analyzed. In chapter 3, we describe the theoretical contribution of this thesis as well as explaining the pre-processing steps, implemented algorithms, employed dataset and the detailed plan of the experimental section of this work. Chapter 4 comprises of the details for practical part of the thesis, training process of classifiers and the experimental results are explained. Eventually in chapter 5, we provide a summary of the conclusions and mention the possible future plan for this work.

CHAPTER 2

State Of The Art

In recent years, Sound Event Detection and Classification (SED/C) have been active fields of research and there have been number of studies over these issues [TMZ$^+$06]. As mentioned in chapter 1, event detection is more complicated than classification of acoustic scenes. This statement can be explained by the fundamental differences of the two following folds. It is apparent that the detection task requires the discrimination of event categories as well as the target event categories from extremely rich background audio. [PKK$^+$17] stated the second fold as having access to the global context of the events (namely, the full event which is fed into the classifier) in a classification task while, in the detection task, we do not know in advance boundaries of the events and usually need to rely on unreliable local audio features (namely, the segment size which is used for the analysis window and do not contain the global context of the event) for inference. Two alternative approaches for the sound event detection have been studied by Tampere University of Technology [MHV16b] where we need to find the most prominent event at each time instance, which will output a monophonic event sequence and was named monophonic sound event detection. The second study is to find a predefined number of overlapping events where produces a polyphonic event sequence as an output, and therefore is called polyphonic sound event detection. The figure 2.1 illustrates an example of the difference between outputs of these two approaches. As shown in the figure, monophonic sound event detection refers to sound events which appears as a sequence with no overlapping signals. Unlike monophonic SED, in polyphonic SED, the event signals overlap which is a realistic example of real life situation.

In the recent research work (from 2016 to 2017) there have been published papers, applying deep learning approaches on sound event detection. In the chart 2.2, number of published works on sound event detection task for DCASE challenge using deep learning approaches within these two years illustrates the large number of dedicated works on convolutional, recurrent and dens layers. The robust performance of CNNs and RNNs on SED task lead to further investigation on hybrid architectures such as

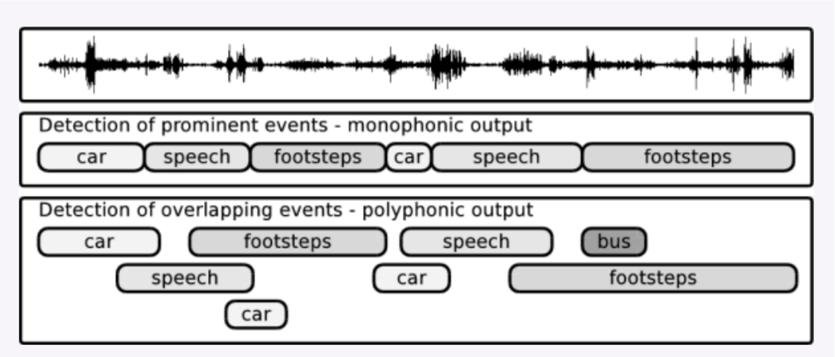

Detection of prominent events - monophonic output
car
speech
footsteps
car
speech
footsteps
Detection of overlapping events - polyphonic output
car
footsteps
speech
bus
speech
car
footsteps
car

Figure 2.1: An example of the differentiation between the outputs of monophonic SED and polyphonic SED [MHV16b].

CRNN, CNN-LSTM and other extentions of RNN model in the year of 2017. The chart analysis was done by categorizing the papers based on the publication year and the applied methods. The color blue represents the year 2016 and color orange represents the year 2017. In the following of this chapter some of these studies will be pointed and their approaches will be analyzed for both monophonic and polyphonic SED tasks in terms of the two most faced issues with this study, data and model selection.

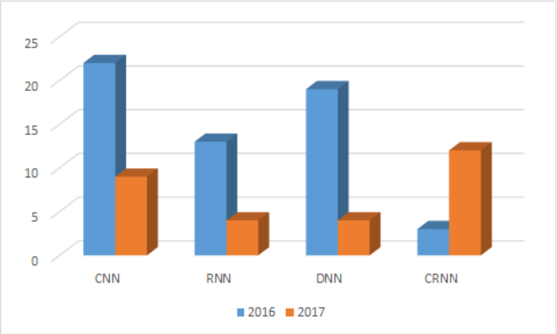


Figure 2.2: The number of published work using deep learning architectures on sound event detection task

2.1 Data Augmentation

In order to achieve good classification/detection results, we need to train the model on numbers of samples. The larger the size of the training sample, the more reliable the performance of the model. However, collecting the training samples is an expensive process. Therefore, there have been proposed approaches which create artificial training samples by manipulating the already existing data. These methods normally create new data by changing the amplitude and/or length of the audio or adding noise to the signals.

Utilizing these artificial signals, improves the complexity of the classifier which makes the machine more general to classify variation of same label audio signals. A method was proposed by [CGK15] which is called Vocal Tract Length Perturbation (VTLP) with the purpose of adding noise to each recorded stream and create more artificial samples. This method have shown very good results for Acoustic Speech Recognition (ASR) tasks. The attempt of this method is to change the length of vocal tracts while extracting the descriptors. Another approach mentioned in [TGPVG16] called Equalized Mixture Data Augmentation (EMDA) which randomly mixes two different audio signals from the same class and further, by boosting a particular frequency band, perturb the sound by moderately modifying frequency characteristics of each source sound. The using a deep convolutional network in [CGK15] achieved 80.6% accuracy with the data augmentation and 76.1% without the data augmentation. The figure below, depicts the effect of using the mentioned methods for data augmentation and improvement on performance of their model.

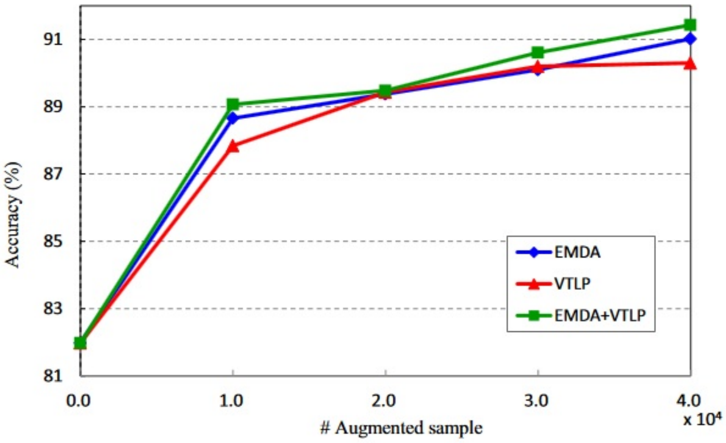


Figure 2.3: Effects of EMDA and VTLP (data augmentation) methods with enlarging number of augmented data [CGK15]

Other popular data augmentation approaches are pitch shifting, time stretching and blocks mixing which is used in the works of [PHV16, LD17a, SB17, SG15]. Blocks mixing method generates new recordings with either equal or higher polyphony which is by combining different blocks of the signals within the same context. In a frequency domain, this can be directly achieved by using the mix-max principle (overlapping two blocks of the log mel spectrogram at the time, overall the whole spectrograms). Pitch shifting is done by scaling linear-frequency spectrogram excerpts vertically which raises or lowers the pitch while time stretching is the horizontal scaling of the linear-frequency spectrogram and is the process of slightly slowing down or speeding up the recording. In recent related works on audio information retrieval, pitch shifting and time stretching have been very popular and have also shown very good results [SG15]. A library for music and

audio analysis called librosa[1] is providing these two functions which is used for the data augmentation purposes in this work. The functionality and effects of these two functions will be explained in detail later on in chapter 3, where the theoretical contributions of this thesis are presented.
The procedure of augmenting the data is placed at the beginning of the SED system, where the actual data is read and is ready to be pre-processed. Figure 2.4 illustrate the flowchart of a SED systems using data augmentation to enlarge the dataset.

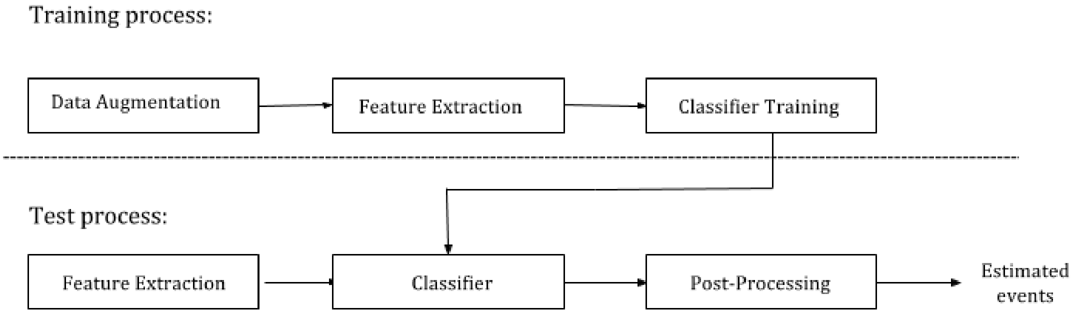


Figure 2.4: Flowchart of the sound event detection system

2.2 Architecture

The focus of the state of the arts in sound event detection systems points to the extreme usage of deep learning architectures [CKBK16, AV17, CKBK16, TGPVG16, PHH$^+$17, PHH$^+$17]. The work of Adavanne, Parascandolo, Pertilae, Heittola and Virtanen in [AV17] is motivated by performance of RNN-LSTM over DNN. For the automatic sound event detection task, they have proposed the use of harmonic and spatial features combining with long short term memory (LSTM) Recurrent Neural Network (RNN) using multi-channel audio. The performance of their approach was with 0.91 for error rate and 35.4 for F-measure slightly better than the Gaussian Mixture Model (GMM) classifier with 0.91 for error rate and 23.7 for f-measure. To face the challenge of multiple event detection, there have been introduced methods such as multiple path Viterbi Decoding or multi-labeled DNN explained in [DHV13] and [CHHV15]. The detection method proposed in [GPMK16] uses concatenated NMF (sparse-CNMF): a given dictionary of events and determines the spectral patches for each class, over time. Based on [GAFC$^+$16], as the addition of two signals in temporal domain does not necessarily result in the addition of their power spectra, a priori decomposition of sound spectra in several components is therefore problematic. Their proposed solution was to use MFCC to directly coding the spectrum of recorded signal. Via splitting the signals into frames before the processing and

---

[1] https://librosa.github.io/librosa/

calculating the first derivatives of MFCC, the temporal dimension of the issue in detecting the event was acknowledged. The proposed system used a non-parametric classifier and have shown that the performance is directly depends on two factors: a proper selection of the sound recordings in order to train the machine and key spectral information to detect the sound event had seemed to be concentrating on the range lower than 8000 Hz.
For the task of rare (monophonic) sound event detection, the hybrid system of convolutional-recurrent neural networks was extensively used. The robustness of this system was proved in [LPLH17] by achieving 0.1307 for the error rate and 93.1% for f-measure and placing them as the winner of the DCASE2017 Rare Sound Event Detection competition. In [PKBGM17], the CNN architecture was used and outperformed the Multi Layer perceptron (MLP) baseline system by achieving 0.2773 error rate and 85.3% for f-measure. Other architectures used for RSED task were the ensemble learning, with MLP, CNN and recurrent Neural networks (RNN)- Long Short term memory (LSTM) applied by [GSR$^+$17] which achieved 0.4267 and 78.6% for error rate and f-measure values, respectively.
In the more complicated task of polyphonic sound event detection (PSED), [AV17] used a deep model which outperformed the baseline system with the error rate value as 0.7914 and 41.7 % f-measure by applying the CRNN architecture on the DCASE 2017 SED task and achieved the first place in this competition. The RNN architecture used by [LD17b] on the same dataset resulted error rate value as 0.8306 and 39.2 % f-measure. A summarization of the state of the art mentioned in this chapter on both polyphonic and monophonic SED task is presented in the table 2.1.

These works motivated this thesis to investigate the behavior of deep models on the SED task using the same datasets provided by the DCASE 2017 for comparison and evaluation purposes. The architecture studied for this project are RNNs and its popular extension, LSTMs and the hybrid system CRNN which is explained and the effect of each hyper parameter in these architectures on the results is analyzed and explained in the following chapter.

Source	Dataset	Featur-Extraction	Method	Err	F-score
[LPLH17]	RSED DCASE2017	log-mel energies	CRNN	0.1307	93.1%
[CV17]	RSED DCASE2017	log-mel energies	CRNN	0.1733	91.0%
[KLB17]	RSED DCASE2017	log-mel energies	CNN	0.3173	82.0%
[RD17]	RSED DCASE2017	log-mel energies	Ensemble	0.4267	78.6%
[WL17]	RSED DCASE2017	log-mel energies	DNN	0.4320	73.4%
[AV17]	SED DCASE2017	log-mel energies	CRNN	0.7914	41.7%
[JLHL17]	SED DCASE2017	log-mel energies	CNN	0.8080	40.8%
[LD17b]	SED DCASE2017	MFCC	RNN	0.8251	39.6%
[Zho17]	SED DCASE2017	log-mel energies	LSTM	0.8526	39.3%
[LL17]	SED DCASE2017	MFCC	Bi-LSTM	0.9523	41.0%
[WL17]	SED DCASE2017	MFCC	RNN	0.9749	40.8%
[HWT+16]	SED DCASE2016 (synthetic)	log-mel energies	Bi-LSTM	0.4958	76.0%
[HWT+16]	SED DCASE2016 (synthetic)	log-mel energies	Bi-LSTM-PP^2	0.4082	78.1%
[CKBK16]	SED DCASE2016 (synthetic)	log-mel energies	DNN	0.3660	78.7%
[KSWP16b]	SED DCASE2016 (synthetic)	log-mel energies	DNN	3.5464	12.6%
[VW16]	SED DCASE2016 (synthetic)	CQT	RNN	0.8979	52.8 %
[APP+16]	SED DCASE2016 (real life audio)	log-mel energies	RNN	0.8051	47.8%
[VW16]	SED DCASE2016 (real life audio)	log-mel energies	RNN	0.9124	41.9%
[KSWP16a]	SED DCASE2016 (real life audio)	MFCC	DNN	0.9557	36.3%
[GMS16a]	SED DCASE2016 (real life audio)	log-mel energies	CNN	0.9799	41.1%

Table 2.1: A tabular summarization of the similar works on sound event detection.

CHAPTER 3

Theoretical Contribution

In this chapter, the techniques for representing the input signals are presented. Afterwards, the data augmentation approaches used for this thesis are explained. Then, the layout of experimental part is presented which comprises of the preprocessing phase on the given datasets, as well as an overview of the selected deep learning architectures.

3.1 Audio Features

In sound detection task, audio features play a very important role. First a sound and its features must be known and for that, a brief explanation of it and its features is provided in this section. A sound is a vibration which propagates only through a transmission medium such as water, air or solid materials such as wood or iron. This means a sound cannot propagate through vacuum. There are numbers of properties which are used to define a wave. Such properties are:

- The wavelength, which is the horizontal distance between two points (either peaks or troughs) on a waveform.
- The amplitude which is the height of the wave.
- Frequency of a wave, known as the cycles that pass a set point in a second and is measured in Hertz.

However, the representation of the sound must give the information needed for the task. The most useful representation of the sound which is used is the spectrogram. Spectrograms are the time ordered series of frequency compositions which reveals information about the frequency content without sacrificing the time information [dOVG$^+$15].
In figure 3.1, on the left, the wave data from a mixture sound of a baby cry within an environmental noise is presented. Notice that the event can be detected, but there is no

information on how it sounds. The red lines was added later on to show the beginning and the ending of event. However on the right side of the figure 3.1, each individual sound and also how it sounds without using the spectral information over the time axis, can be seen. In comparison with other sounds such as a gunshot or a glass breaking, figures 3.2 and 3.3 illustrate the waveform and the spectrogram of a glass break and waveform and the spectrogram of a gunshot, respectively. The background noise used for the baby cry mixture is residential area, for glass break is sound of water waves and for the gunshot is the street which sound of running engine is part of it. Approaches which are used in this work to extract sound features and produce such spectrograms are Short Time Fourier Transform, Mel Frequency Cepstral Coefficient, Mel band energies, and Constant-Q Transform.

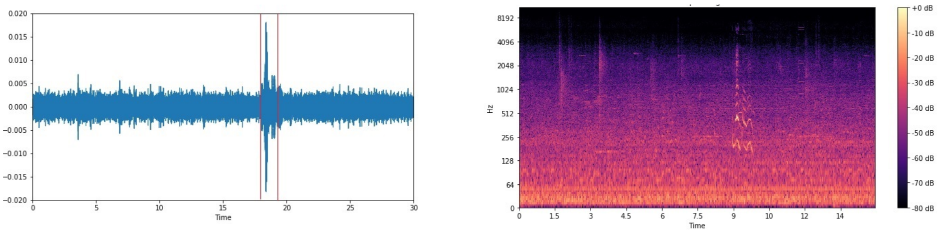


Figure 3.1: Sub-figure(a) illustrates the waveform and sub figure (b) is the calculated spectrogram of waveform, representing a baby cry

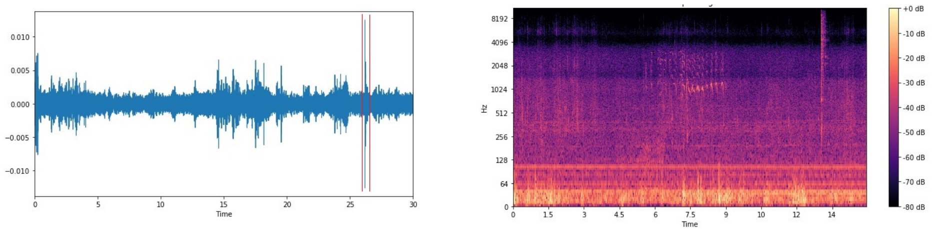


Figure 3.2: Sub-figure(a) illustrates the waveform and subfigure (b) illustrates the spectrogram of the waveform, representing a glass break

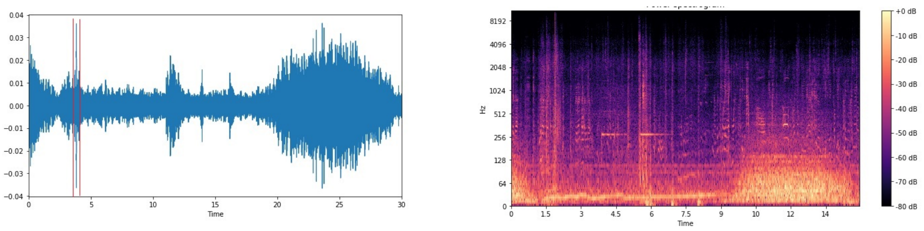


Figure 3.3: Sub-figure(a) illustrates the waveform and subfigure (b) illustrates the spectrogram of the waveform, representing a gunshot

3.1.1 Short Time Fourier Transform

Short Time Fourier Transform also known as Short Term Fourier Transform (STFT) is a powerful tool for audio signal analysis which defines a useful class of time frequency distribution [All77]. The procedure here is to divide a longer time signal into smaller segments with equal length segments and afterwards, computing the Fourier Transform on each short segment, separately.
Taking this procedure step by step, first the Fourier Transform and Fourier Series are defined as follows:

- The Fourier Transform is a mathematical method to convert the non-periodic function in the amplitude vs time domain to the amplitude vs frequency domain.
- The Fourier Series is a mathematical method to convert the periodic function in the amplitude vs time domain to the amplitude vs frequency domain.

Note the similarity between Fourier transform and Fourier series, both convert a time domain to a frequency domain, however Fourier Series is applied on periodic functions and Fourier Transform is applied on non-periodic functions. Namely, Fourier Transform only takes a single segment of the signal and applies the Fourier computation. The result is a continuous frequency value which looks the same as the discrete frequency value returned by the Fourier Series for a periodic function. Short-time Fourier transform (STFT) is a complex-valued matrix such that this matrix is the magnitude of frequency bin f at frame t. To calculate the STFT, the following parameters are then needed:

1. First, define the length of the analysis window.
2. Amount of overlap among each window.
3. The chosen windowing function in order to avoid spectral leakage. The most common functions are Hann and hamming windows.
4. Generating the window segment via multiplying window function by signal.
5. At the end, apply the Fast Fourier Transform (FFT) computation on each windowed segment.

In order to get the most dominant frequency, FFT gives a distribution over a group of different frequencies. As mentioned before, each signal has a time domain and a frequency domain which has N complex points. Time domain refers to how the signals change over time and frequency domain refers to the magnitude of the signals which lie in the frequency range. By theory (Fourier series), signals are composed of many sinusoidal signals with different frequencies, like triangle signal, it is actually composed of infinite sinusoidal signal (fundamental and odd harmonics frequencies) [Smi99]. The main parameters of the FFT are window size and FFT size. Window size defines the duration and number of samples which depends on the fundamental frequency, intensity and changes of the signal. The window size influences the representation of the signal or the frequency resolution. However, the FFT size is the number of the bins of the analysis window. By changing the FFT size, the frequency resolution can be increased and by default it gets the value with a power of 2 factor (2, 4, . . . , 512, 1024, ..) [Smi99]. Short Term Fourier Transform is a solution to capture the temporal changes of the content of the spectral. As explained before, STFT applies Fourier Transform on each short period of time. The window length determines the frequency resolution. Namely, the larger the Fourier Transform input, the higher the resolution of the frequency. After applying this computation steps, one last step is needed in order to represent the spectrogram of the given input. As mentioned before, spectrogram is a series of the consecutive magnitude Fourier Transforms on a signal. Figure 3.4 shows the spectrograms resulted after applying STFT to extract the features, using different length for time-window. By carefully examining the illustration, one can notice the bigger number of time-windows zooms out the image (outputs an image with higher resolution) and the smaller number, obviously, zooms into the image (outputs an image with lower resolution). The audio is the sound of a baby cry on a residential area as background noise. This audio is a sample from TUT rare sound event detection 2017's dataset. Library used for this feature in this work is Librosa which provides STFT as its core application.

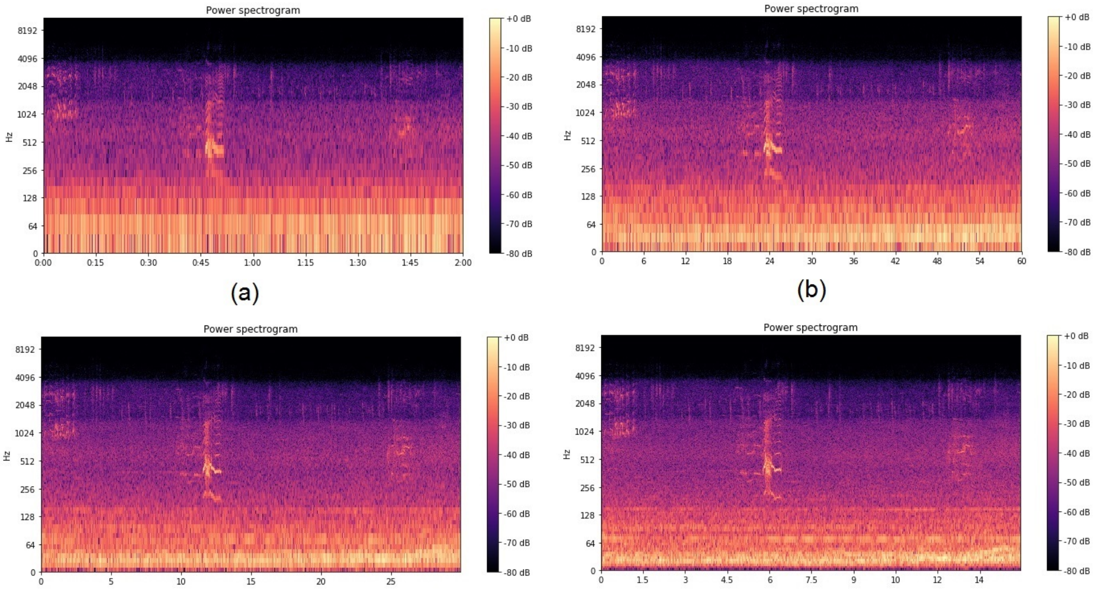


Figure 3.4:
Illustration of the different window sizes in Short Time Fourier Transform spectrograms. The larger the window size, the more zoomed out the spectrogram becomes. Sub-figure (a) has the FFT window size 512. In sub-figure (b), the FFT window size is 1014. Sub-figures (c) and (d) have the FFT window sizes 2048 and 4096.

3.1.2 Mel-Spectrograms and Mel Frequency Cepstral Coefficients

SED task mainly focuses on training the system to distinguish events occurring in an audio stream. The most common feature extraction technique used for this matter is Mel Frequency Cepstral Coefficients called MFCC which is highly effective, powerful under various conditions and is less complex to implement [PKVK13].
The Mel, comes from melody which indicates that the scale is based on pitch comparison and was named by Stevens, Volkmann, and Newman in 1937. The motivation behind the MFCC design is the knowledge of human auditory system. The step by step computation of MFCC are pre-emphasis, Framing, windowing, Fast Fourier Transform, Mel filter bank and computing DCT (Discrete Cosine Transform).

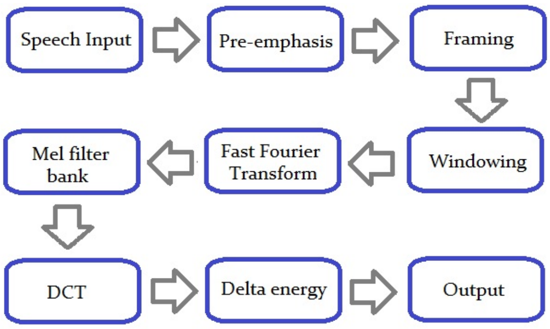


Figure 3.5:
MFCC block diagram

Pre-emphasis: In this step, the input audio signal is passing through a filter which emphasizes higher frequencies. This process increases the energy of the signal at higher frequency.

Frame Blocking: Through the process of segmentation, the input signal is divided into smaller duration called frames. Since an audio signal can be a time varying, it needs to be examined in shorter durations, therefore short time spectral analysis is required.

Hamming Windowing: Hamming window is a window function used to reduce discontinuity. In the equation below, W(n) is the window function, X(n) is the signal which they output a signal Y(n).

$$Y(n) = X(n) \cdot W(n) \tag{3.1}$$

Fast Fourier Transform: In this step, as mentioned the the STFT section, the time domain will be converted to a frequency domain. By using FFT (supported by the equation below), the magnitude frequency response of each frame can be obtained and the output is a spectrum. Equation below shows this calculation where X_p represent the

Fourier Transform of the windowed p^{th} frame of the signal $X[n]$ for each k=0,.., N-1.

$$X_p(k) = \sum n = 0^{N-1} x_p[n]\omega[n]exp^{-j\frac{f\pi kn}{N}} \tag{3.2}$$

Mel-Scaled Filter Bank: The goal of this step is to get smooth magnitude spectrum. It also reduces the size of the features involved. Using the following formula [PKVK13], the mel spectrograms for each input frequency is computed:

$$L_p(m,k) = log_{10} \sum_{K=0}^{N-1} M(m,k) \cdot |X_p(k)| \tag{3.3}$$

where m= 1, .., F and p=1, .. P. The filter bank output is the product of the Mel filter bank M and the magnitude spectrum $|X|$ and mel filter banks $M(m,k)$ are computed as

$$M(m,k) = \begin{cases} 0, & \text{for } \mathrm{l}_f(k) < l_f(m-1) \\ \dfrac{l_f(k) - l_{fc}(m-1)}{l_{fc}(m) - l_{fc}(m-1)}, & \text{for } l_{fc}(m-1) < l_f(k) \leq l_{fc}(m) \\ \dfrac{l_f(k) - l_{fc}(m+1)}{l_{fc}(m) - l_{fc}(m+1)}, & \text{for } l_{fc}(m) < l_f(k) \leq l_{fc}(m+1) \\ 0, & \text{for } l_f(k) \geq l_f(m+1) \end{cases}$$

(3.4)

Discrete Cosine Transform: In order to achieve the L mel-scale cepstral coefficient, DCT is applied on the 20 log energy obtained from the previous step.

$$\phi_p^r x[n] = \sum_{m=1}^{F} L_p(m,k) cos[\frac{r(2m-1)\pi}{2F}] \tag{3.5}$$

where r = 1, .., F and $\phi_p^r x[n]$ represent the r^{th} MFCC of the p^{th} frame in the audio signal. The overview of MFCC algorithm:

1. Segment the audio signal into short durations
2. For each frame calculate the periodogram estimate of the power spectrum.
3. Apply the mel filterbank to the power spectra, sum the energy in each filter.
4. Take the algorithm of the filter bank energies.

5. Take the DCT (Discrete Cosine Transform) of the log filterbank energies.

In order to generate the mel features, two different librosa functions, MFCC and Mel-Spectrogram were used. The first function, *librosa.feature.mfcc*, computes the Mel-frequency cepstral coefficients which outputs an mfcc matrix which is a numpy array with the size of (n_mfcc, T) which are the number of the mfccs and the track durations in frames. Figure 3.6 illustrates the spectrograms generated by MFCC feature extraction approach. As before, different numbers for MFCC to return are depicted. The audio signal is the same as before. The difference between the sub-figures with different MFCC numbers (20, 40, 80, ...) is not obvious to human eyes. However, by increasing the number of MFCCs, features with higher frequency are captured.
The function *librosa.feature.melspectrogram*, computes a mel-scaled spectrogram which outputs a numpy array matrix of mel scales. The parameters passed to this function are only the wave-data and sample rate, then its magnitude spectrogram is computed at first and it will be mapped onto the mel-scale by $mel\_f.dot(S \cdot \cdot power)$. By default, power is set to 2. Figure 3.7 illustrates the spectrograms generated by mel-spectrogram feature extraction approach. Unlike the MFCCs figure, differences in increasing the number mel-bands are depicted in each sub-figure. The larger number of bands results in capturing the high frequency features. Therefore, the babycry in this image is visibly separated from the background noise. Note that two other events are also visibly noticeable which shows their high frequency feature. These events are birds singings (two different bird types).

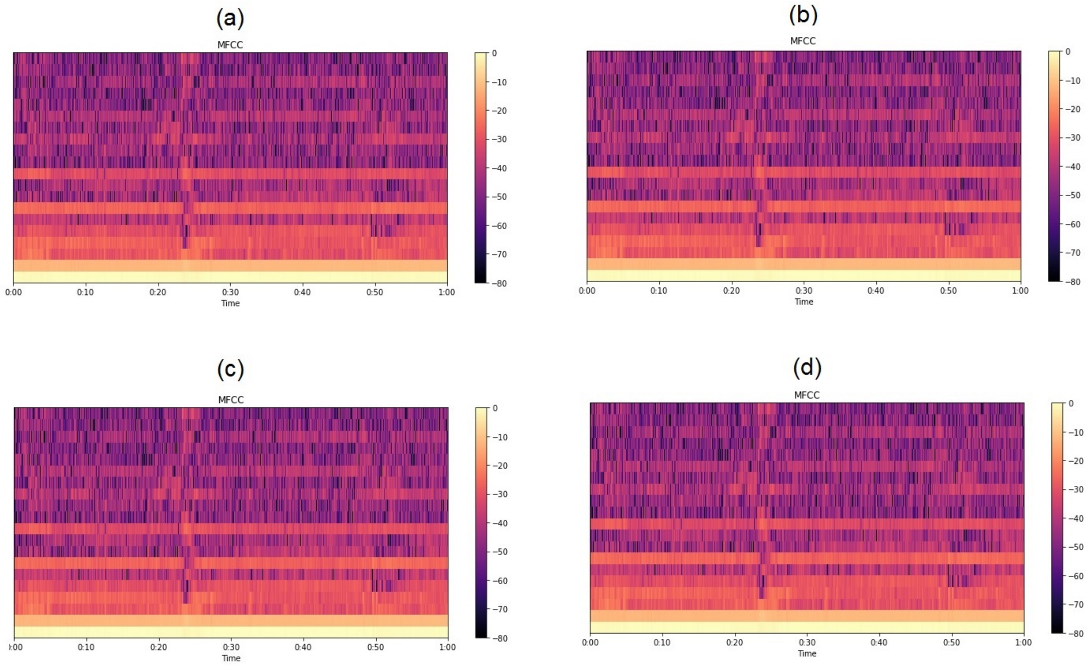


Figure 3.6:
Comparison of different mel-bands in Mel Frequency Cepstral Coefficients. (a) 20, (b), 40, (c) 80 and (d) 160 mel bands.

3.1.3 Constant-Q Transform

Constant-Q Transform (CQT) resembles human auditory system. Similar to human auditory sense, a digital computer also requires more time in order to perceive low tune frequencies. To calculate the constant Q transform of some sequence x, first it is required to examine the Fourier filter formula:

$$\sum_{n<1}^{N} x[n]e^{-2\pi inz}/N \tag{3.6}$$

This filter has the frequency to resolution ratio as $z = \frac{f_z}{\Delta_z^{ft}}$ and the bandwidth is Δ_z^{ft} which is the sampling rate divided by the number of samples N and it is independent of z. Therefore, by choosing a window of length $N_k = \Delta_k^{cq}$ can be perceived. In this equation, f_s is the sampling rate and f_k denotes the frequency of the k^{th} bin.

Note that cq-bins which are the components of the CQT will be calculated as the above filter. However, to match the properties, the appropriate values for z and window length N needs to be previously recognized. By choosing Q as z, a constant value Q for the frequency to resolution ratio of each cq-bin is achieved. Hence the Q^{th} Discrete Fourier

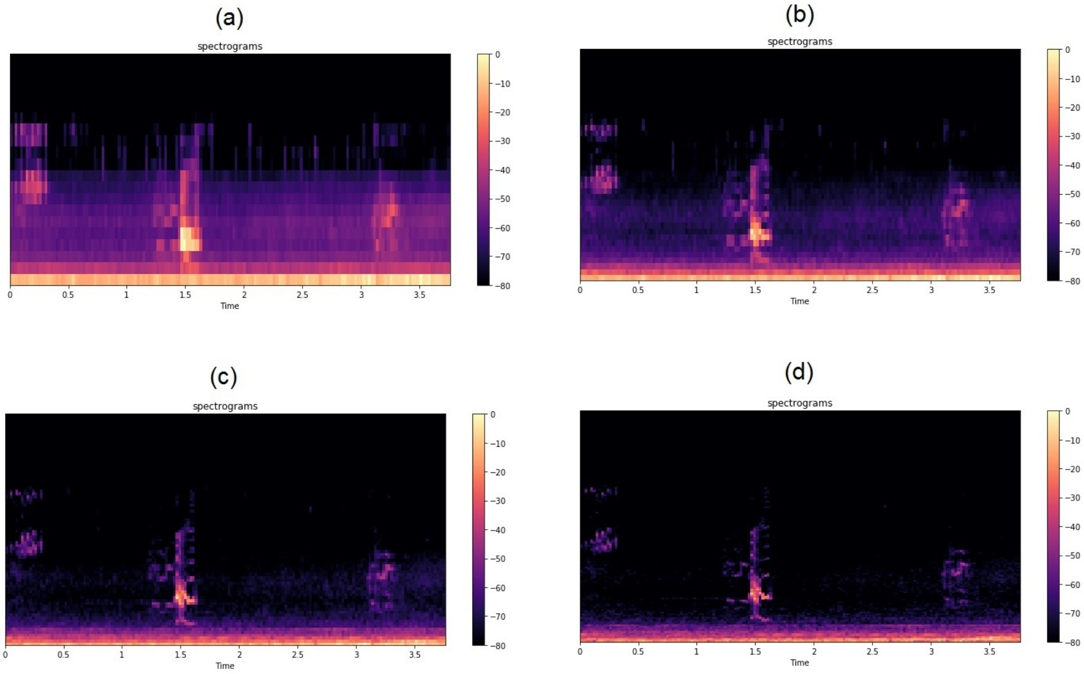


Figure 3.7:
Comparison of different mel bands in Mel spectrograms. (a) 20, (b) 40, (c) 80 and (d) 160 mel bands.

Transform (DFT)-bin with the window length $Q\frac{f_s}{f_k}$ is the integer value Q the k^{th} cq-bin. An efficient algorithm for calculating Constant Q Transform is proposed in the work of Judith C Brown and Miller S. Puckette [BP92].

A summary of the CQT calculation process, first a minimal frequency f_0 and the number of bins per octave are determined (these are according to the requirements of the application). f_{max} is only affecting the number of cq-bins which needs to be calculated. It is also recommended to use any window function such as hamming window, in order to avoid the spectral leakage.
The figure 3.8 illustrates a comparison between the CQT features and STFT features which was computed by librosa core functions. The reason we compare CQT with FFT is because of its close relation to FFT which results in bank of filters but visualization of their spectrograms are completely different.

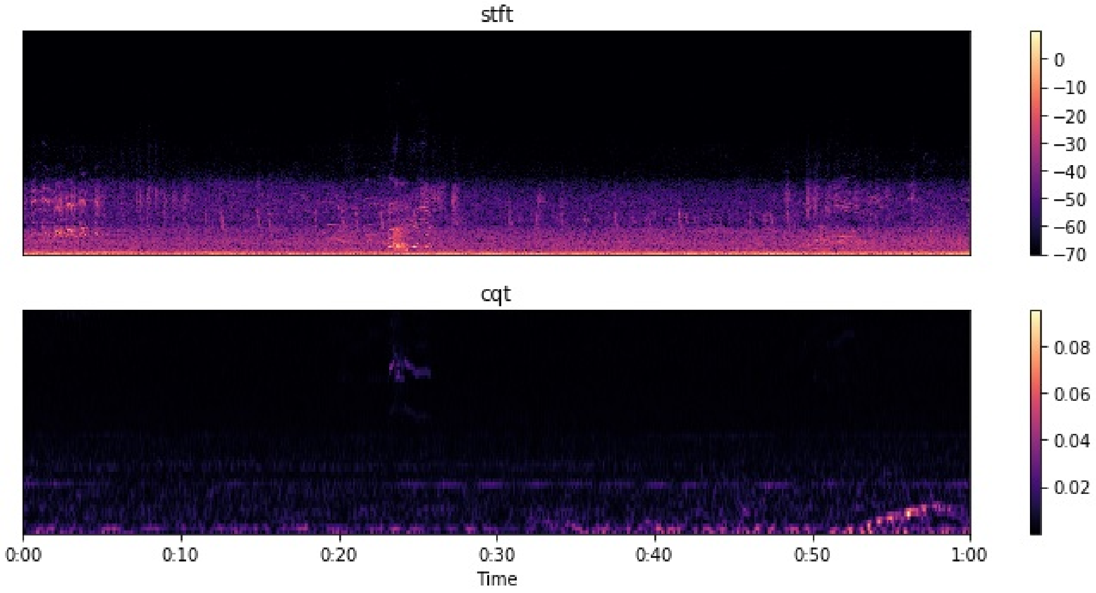


Figure 3.8:
Comparison of STFT and CQT. The top image, shows the STFT results with 4096 FFT size with 120 bins.

3.2 Data Augmentation

In order to increase the performance of the models, a large amount of training data is required. However, collecting such amount of data is expensive. The offered solution of this issue is to artificially enlarge the size of the dataset by generating manipulations of the existing data. Data augmentation enhance the performance of the models in many machine learning applications. Different variations of each label in the training data helps the deep neural network to learn the classification tasks much better. The goal of the data augmentation approaches is to improve the generalization ability of the machine learning models, accuracy in detection and classification tasks and controlling the over-fitting. Issue which occurs in this task are such as class imbalance. Class imbalance refers to the problem of having classes with large number of samples and classes with fewer samples. For this thesis, since the learning process is a supervised learning, namely the training data is labeled, the method used are some audio manipulations from librosa library [MRL$^+$15] with label preserving. Using time and frequency effects, from each audio input, two different classes of audio waves are generated. Functions used for these purposes are *librosa.effects.time_stretch* and *librosa.effects.pitch_shift*. *Time_stretch*, stretches the time in an audio file by a fixed rate and *pitch_shift*, shifts the pitch in the waveform by a given number of steps half-steps.

3.2.1 Time_Stretch

The output of this function is a time stretched waveform. The generated audio, depending on the given stretch factor, is either faster (compressed waveform) than the original audio, or slower (stretched waveform). To generate such output, first by applying the Short Time Fourier Transform, the given waveform is transformed to its frequency domain. Afterwards, using the Phase Vocoder approach presented by Laroche and Dolson in [LD99], the frequency of the wave data is stretched. Phase Vocoder is an algorithm which stretches or compresses the time-base of a spectrogram to change the temporal characteristics of an audio.
The length of the timesteps array determines the length of generated audio. The difference between each timestep in this array is equal to the stretch factor. To make it a bit more clear, given the stretch factor 2, for a waveform with length 10 seconds and 44100 sample rate (wavelength = 441000), the calculated timesteps will have the wavelength equal to 22500 (5 seconds).
After calculating the linear magnitude interpolation from the actual waveform and storing them into the output array, the Invert Short Time Fourier Transform is used to transfer the frequency domain back to the time domain and the stretched waveform is then returned as the output.
As an example, the audio can be twice as fast as its original speed (compressed) by giving the stretch factor as 2. Note that this factor should always be a positive number. An audio can be also expanded (slower than the original audio) by giving the stretch value as a real value smaller than 1. For example, given the stretch value as 0.5 creates a signal, 2 times slower than the original signal, the function will compress the audio by twice as fast as its original speed.

Figure 3.9 illustrates one of the augmented wave generated by the time_stretch function. The stretched rate for the augmented wave was set as 2. As noticed here, the augmented wave is compressed

In order to stretch an audio, the value for the stretch rate needs to be smaller than 1 and bigger than 0. Figure 3.10 shows this example which the stretch rate was chosen as 0.5. This made the augmented audio to be half the time slower than the original audio. In this image, notice the time axis, the length of the audio was stretched to twice as the original length.
When using the time_stretch function for data augmentation, given the stretch rate, the onset and offset of the labeled audio will be accordingly changed, therefore the onsets and offsets of the events in the newly generated audio are calculated as follows: The given onset and the offset of the original formula divided by the stretch factor gives the onset and offset of the target audio.

$$target\_onset(offset) = \frac{original\_onset(offset(x))}{stretch\_factor} \tag{3.7}$$

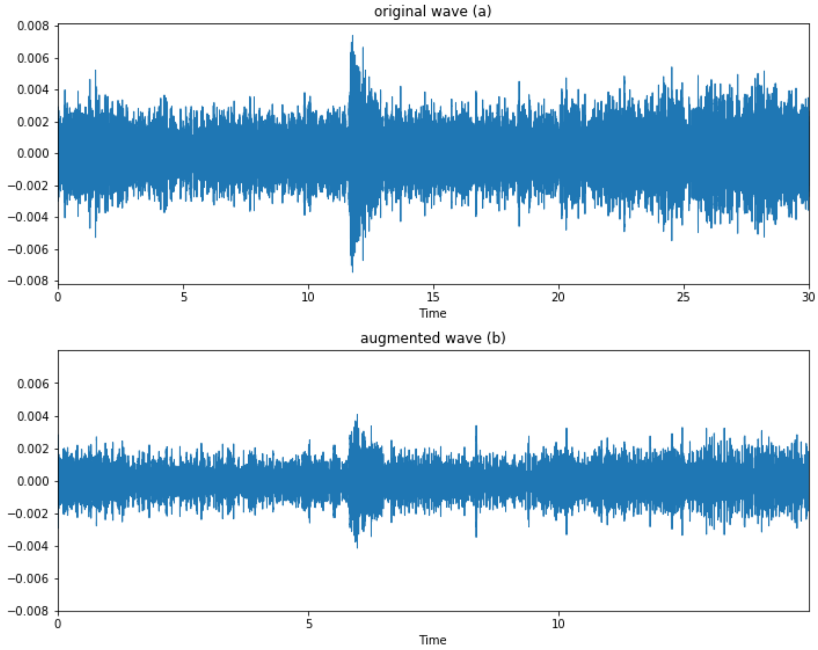


Figure 3.9: The Original waveform (a) with the length 30 seconds and the augmented waveform (b) with the stretched_factor = 2 which resulted the compressed waveform with length 15 seconds.

3.2.2 Pitch_Shift

To shift the pitch of an audio, two parameters need to be determined; the number of steps which is desired to shift the wave data and also the desired number of bins per octave which determines the number of the steps per octave and have the default value of 12 (the actual number of the steps in one octave). The function will then outputthe pitch shifted form of the original waveform. The function calculates the rate by given the mentioned parameters as:

$$rate = 2^{\frac{n\_steps}{bins\_per\_octave}} \tag{3.8}$$

Afterwards, it will stretch the wave (with the calculated rate) in time and re-sample it from the calculated sample rate to the original sample rate. At last it returns the shifted wave with the fixed length as the original wave data. Figure 3.11 and 3.13 illustrate the augmented wave data compared with the original wave data. The results is also shown as spectrograms in figures 3.12 and 3.14.
In figure 3.11 the number of steps were set as 2, resulting an audio with higher pitch than the original one and in figure 3.13 the number of steps is set to be -2 which resulted the augmented audio with lower pitch. The figure 3.12 visualizes the difference between the mel-spectrogram of the original signal and the augmented signal, using pitch shifting with 2 number of steps in the octave. Using the equation 3.7 the calculated rate for the

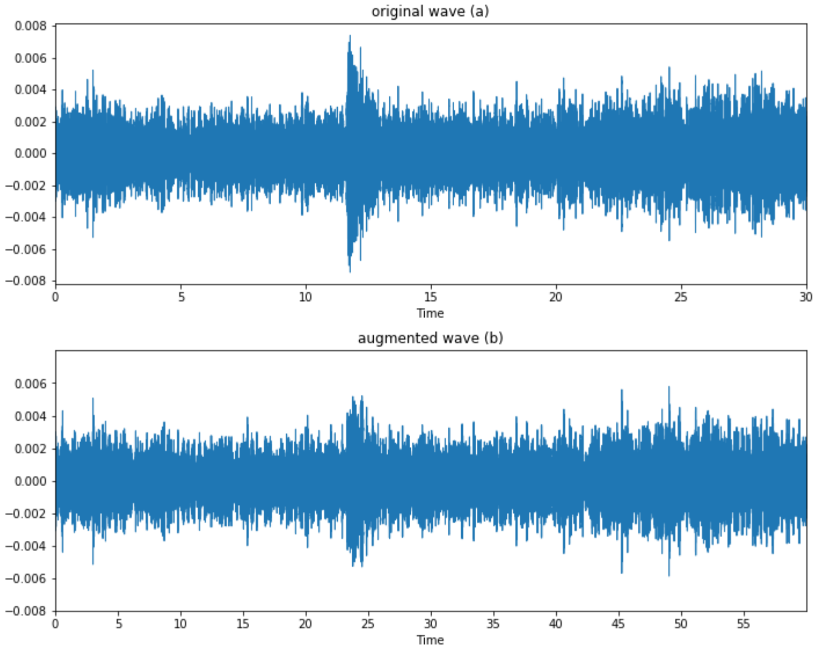


Figure 3.10: The Original waveform (a) with the length 30 seconds and the augmented waveform (b) with the stretched_rate = 0.5 which resulted the expanded waveform with length 60 seconds.

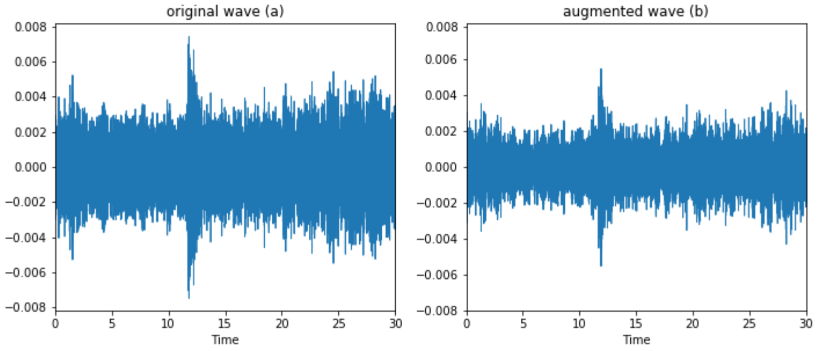


Figure 3.11: original waveform (a) and the pitch shifted waveform (b) with the number of steps = .

2 steps per octave is 1.122 which means 12% shifting up the signal's pitch. As for the figure 3.14, calculating the rate for -2 as the value of the number of steps in an octave, the value 0.8908 is resulted which is 11% shifting down the pitch in the signal. A good way to choose the steps parameter is to listen to the augmented waveform. Created audio should be noticeably different from the original audio while sounding like what is is supposed to sound i.e. to augment a baby cry, by increasing/decreasing the steps more than 8 (58% up shifting the pitch) or less than -6 (30% sown shifting the pitch), the cry

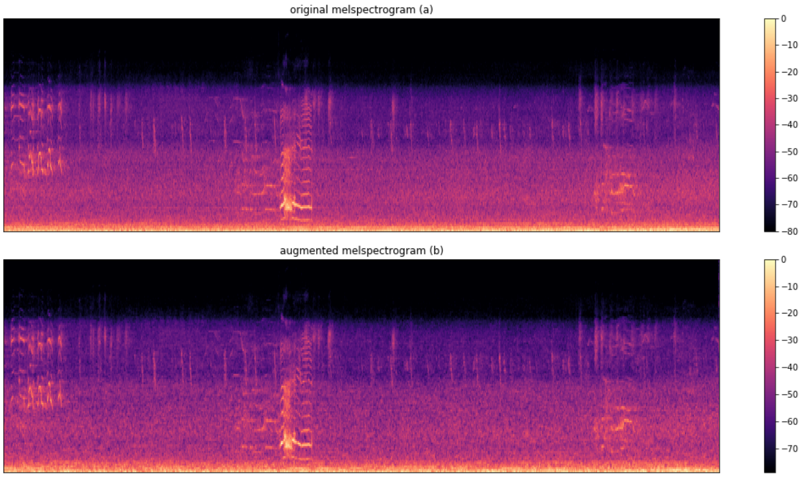


Figure 3.12: Original spectrogram (a) and the pitch shifted spectrogram (b) with the number of steps = 2. By looking carefully at the images, it is noticeable that the distance in the y axis has been slightly stretched (up-shifted pitch) which is mostly noticeable at the event area. The event shows a baby cry and the spectrogram is the results of calculated mel energies with FFT window size = 2014.

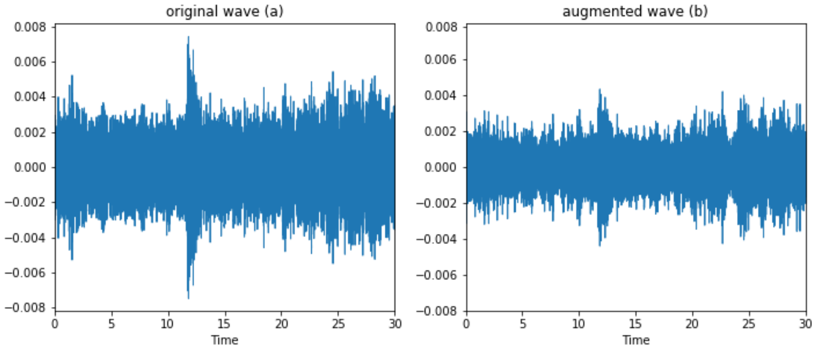


Figure 3.13: Original waveform (a) and the pitch shifted waveform (b) with the number of steps = -2.

did not sound like a babycry anymore and was an animal-like sound.

3.3 Learning Process

This section consists of the data preprocessing and architectures used for this work. Afterwards the post-processing approaches required to achieve the desired results is mentioned. Before fitting the data into the designed model, first it should be appropriately shaped to be used as the input of the model. Afterwards by setting the values for model's parameters, the model will train itself and starts the learning process on each data. All

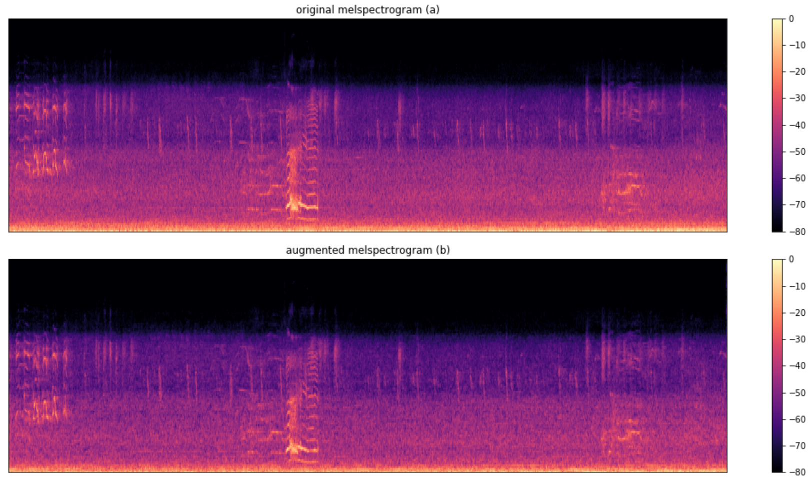


Figure 3.14: Original spectrogram (a) and the pitch shifted spectrogram (b) with the number of steps = -2. By looking carefully at the images, it is noticeable that the distance in y axis has been slightly compressed (down-shifted pitch) which is mostly noticeable at the event area. The event shows a baby cry and the spectrogram is the results of calculated mel energies with FFT window size = 2014.

these steps are explained in more detail in their corresponding sub-section.

3.3.1 Pre-processing

The aim of the data pre-processing is to reduce the complexity of the data in order to enhance the performance of the learning algorithms. Data cleaning, integration, transformation and data reduction are some of the major tasks in data pre-processing. Data cleaning refers to removing the outliers, smoothing the noisy data and filling the missing values. For pre-processing phase of this thesis i.e Z-score normalization or attribute-wise standardization, Scikit-learn library is used. Scikit-learn is a powerful machine learning library which provides pre-processing functions such as StandardScaler which standardize features by removing the mean and scaling to unit variance. Standardization re-scales the features in a way that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one.
The datasets used for this work are already cleaned and normalized by the provider (Tampere University of Technology) for the DCASE challenge. Namely, all the data are correctly labeled with their appropriate events and the continuous values for their onsets and offsets. Also the time scale is all in the same unit with the duration of 30 seconds. Each audio is labeled with either no event, babycry, glassbreak or gunshot. Having 3000 audios, 500 of them are labeled with baby cry, 500 with glassbreak and 500 with gunshot, in total there are 1500 audios which an event occurs in them and 1500 audios which no event occurs in them.

However, the third task of the DCASE 2017 do not have equally distributed labels. All the audios in the development set are labeled with the 1 or more events which have been occurred within them. To overcome this issue, data augmentation was applied on the other 5 labels (by augmenting only segments of the audio where the other 5 labels occur) in order to have normal distribution of the labels. The overall distribution of labels among all 24 audios are illustrated in the figure 3.17. As depicted below, for the label brakes, there are 52 occurrences, for cars, 304 which is the dominant event in all the audios. For children, large vehicle, people speaking and people talking, there are respectively 44, 61, 89 and 109 occurrences.

3.3.2 Deep Learning Architectures

Since the scope of signal analysis and processing research has been significantly widened, it has embraced many broad areas of information processing and machine learning has been an important area of these researches [Den12].
Since 2006, deep learning has emerged and the technique, developing from its research have affected a broad range of signal and information processing within the traditional and the new extensive scopes including artificial intelligence and machine learning [HOT06]. Deep Learning refers to a class of machine learning techniques where within a hierarchical architecture, many layers of information processing stages are extracted and as mentioned in the first chapter these techniques, in contrast with the earlier techniques such as Support Vector Machines (SVMs) have shown very good performance [MHV16b, GPMK16, GAFC$^+$16, HWT$^+$16]. There are variants of deep architectures; Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), Deep Belief Neural Networks (DBNNs) and Recurrent Neural Networks (RNNs).
In this work, chosen methods are RNNs and its extensions which have been selected based on their popularity for the task of Audio Event detection. Afterwards CNN architecture will also be tested to compare the performance of this architecture with RNNs for detection and classification tasks on time series data. Following this chapter, a brief explanation of each method is provided and afterwards the result of implementation of them and a combination of them are shown in the next chapter.

Recurrent Neural Networks

Recurrent neural networks address the issue of reasoning about previous events in order to make a decision for future events. They are networks with loops in them, allowing information to persist. Recurrent Neural Networks (RNNs) are popular models of Deep learning Networks. The internal structure of a RNN forms a directed graph which displays a dynamic temporal behavior and is called recurrent because of its ability to perform the same task for every element of the given sequence, having the output depended on the previous computations, or so to say, they are able to memorize what had been observed and calculated sofar. However there is limitation on its “memory”; only the information for just a few states before is captured. Tasks that RNN is popular for: Speech recognition, Natural Language Processing, Handwriting recognition, ...

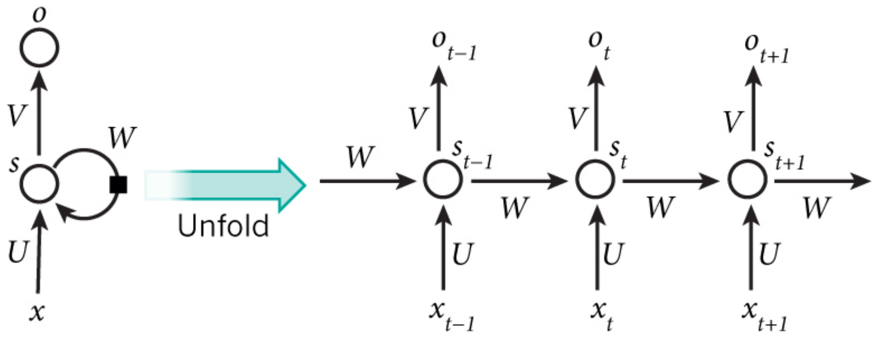


Figure 3.15:
A recurrent neural network in its forward computation process[1]

In figure 3.15, unfolding the network refers to the number of iteration process in the recurrent cell which depends on the given input. each of these iterations are called a time-step. S_t is the unit memory which captures the previously calculated information. RNN shares the same parameter at each step (W,U,V) which greatly reduces the number of parameters needed to be learned. This is an advantage of RNN in comparison with traditional DNN which used different parameters at each level. The hidden states of an RNN are its main feature which represent the information of previous steps. As extensions of RNN, Bidirectional RNNs can be mentioned. The idea behind this extension is that the output of a specific time may not only depend on the previous elements, but also the future elements. For example in an Natural Language Processing (NLP) task, to predict a missing word, the previous word and the next words need to be known. BRNN have relatively simple model. They are only two RNNs stacked on top of each other. Deep Bidirectional RNNs are another extension of RNNs (also BRNNs) which is similar to BRNNs but multiple layers per time steps are sustained.

Denoting the feature vector as $x_1, x_2, ..., x_T$, the equation below mathematically describes an RNN network. An RNN with a hidden layer output vector h_t and output layer one y_t are calculated as follows:

$$h_t = f(W_1 + W_r h_t - 1 + b_1) \tag{3.9}$$

$$y_t = g(W_2 h_t + b_t Hz \tag{3.10}$$

In the equation above, W_i represent the input weight matrix and b_i denotes the bias vector of the $i^t h$ layer, W_r represents a recurrent weight matrix. f is the hidden layer

---

[1] http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/09/rnn.jpg

[2] http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/09/rnn.jpg

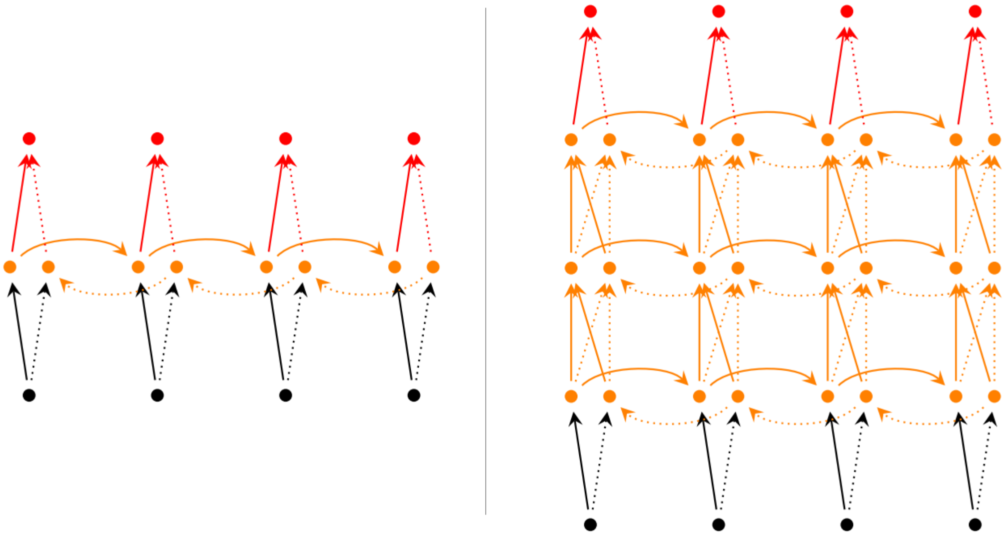


Figure 3.16:
Architecture of BRNN with one hidden layer and 3 hidden layers.[2]

function and g is the output layer function.

The Problem of Long-Term Dependencies

One major problem with RNNs is that they cannot learn context information over long stretches of time which is due to the so called vanishing gradient problem [PMB13]. The training process uses something called gradient which measures the rate at which cost changes with respect to weights or biases. While training a neural network, the cost value is constantly calculated. The cost value is the difference between the predicted output and the actual output. This value is then lowered by slightly adjusting the weights and biases over and over throughout the training process, until the lowest possible value is obtained. This process is called back propagation. Since the obtained values from back propagation gets smaller and smaller, it leads slowing down the training process. This is the issue which developers were facing while training an RNN model. The solution proposed to avoid the vanishing gradient descent are:

1. The choice of the activation function: many activation functions squash their input into a very small output range In a very nonlinear fashion i.e. sigmoid function maps a real number onto a small range between zero and one which results to mapping large regions of the input space to an extremely small range. Therefore, even a large change in the input will produce a small change in the output and hence, the gradient is small. Therefore activation functions such as rectified Linear Unit (ReLu) are strongly advised. This activation function maps the value x to $max(0, x)$.

2. LSTMs or GRUs: LSTMs (short for Long Short Term Memories) and GRUs (short for Gated Recurrent Units). These both extensions of RNN, use different approaches of gating information to avoid vanishing gradient problem. GRUs was introduced by Cho et al in 2014 [CVMBB14] and have a very similar architecture to LSTMs and is achieved by modifying the recurrent unit in an RNN network and adding a new variable to represent the memory cell. The task of this cell is to remember and forget its state based on the input signal to the unit. Each GRU thus has a reset gate and an update gate. This unit fully exposes its memory content at each time-step and balances between the previous memory content and the new memory content strictly using leaky integration, albeit with its adaptive time constant controlled by update gate. The update technique helps the GRU to capture long term dependencies. At the point where a previously detected feature, or the memory content is considered to be important for later use, the update gate will be closed to carry the current memory content across multiple time-steps. By allowing the GRU to reset itself whenever the detected feature is not of use anymore, the reset function helps to use the capacity of the cell memory efficiently [CGCB15, LD17a]. As for LSTMs, this extension is chosen to be experimented in this thesis, motivated by the work of [HWT^{+}16] explained in more details in the following section of this chapter.

Long Short Term Memories

Long Short Term Memory is a special case of deep learning RNN which solves the problem discussed before. It is very similar to RNNs but uses different methods for computing the hidden states. The memories in LSTM are referred to as cells.These cells carefully decide whether an information is needed (to keep the information) or not (to erase them). For that, gates are used to let the information in (or not) which are formulated from a sigmoid neural net layer and a point-wise multiplication operation.
Sigmoid Neural Net layer which outputs a value between 0 and 1 and represents how much of a information are allowed to go through. In other words, if the function outputs zero, the information is not needed at all (let nothing through) on the other hand the value one means all the information are important (let everything through). Point-wise multiplication Operation is used to generate a continuous value between 0 and 1, in order to determine how much of the given information is important to hold on to. Describing this approach more in detail, as mentioned before, an LSTM contains three gate layers: A forget gate layer, an input gate layer and a tanh layer.

First Layer: Forget layer is used in the first step of the LSTM which decides which information is going through and which is not.
Note that deciding which information to forget and which ones to keep is learned automatically from the data. The target labels that are used and the process of training with gradient back-propagation will accordingly adjust the parameters in order to understand the importance of the information for keeping them and in contrary forgetting the non-important information. Learning the parameters which control this decision is modeled

by the memory cell through the computation in this unit.

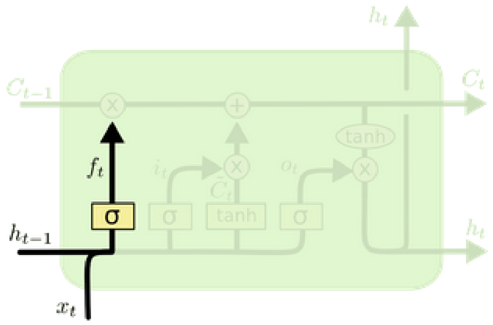


Figure 3.17:
Forget layer in the LSTM Architecture[3]

Second Layer: In this step, we need to decide what new information is needed to be stored in the cell state. This process is performed in two parts: first part is a sigmoid function which decides which values needs to be updated. Afterwards a tang layer, creates a vector of new candidate values that could be added to the states. The reason behind using the tanh as the function in this layer is its ability to sustain for a long range before going to zero.

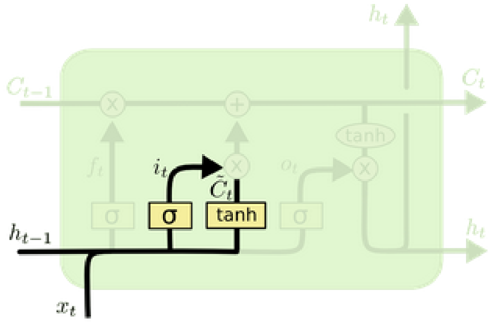


Figure 3.18:
Input layer in the LSTM Architecture[4]

Third Layer: Next, the out outputs of the previous step (tang gate and the input gate) will be combined in order to create an update for the state. Afterwards they combine the current state, the previous knowledge and the input which makes these units very efficient at capturing long terms dependencies (states). In the recurrency of LSTM, the activation function is the identity function with a derivative of 1.0. Therefore, the back-propagated

---

[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[4] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

gradient remains constant [ZSV14].

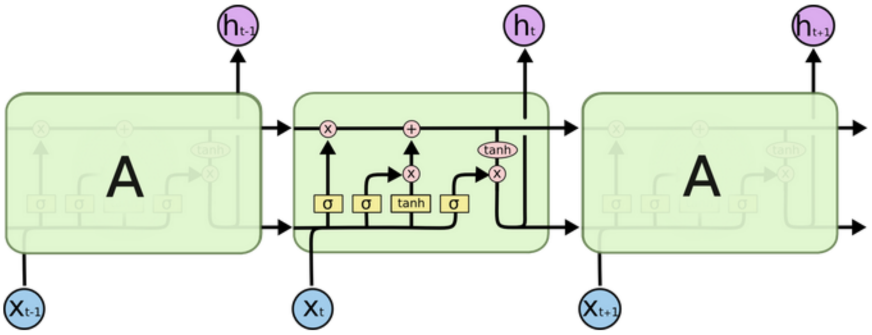


Figure 3.19:
The repeating module in an LSTM contains four interacting layers. [5]

To mathematically describe LSTM, the previous RNN formula is replaced by the following equations:

$$g_t^I = \sigma(W^I x_t + W_r^I h_{t-1} + s_{t-1}), \tag{3.11}$$

$$g_t^F = \sigma(W^F x_t + W_r^F h_{t-1} + s_{t-1}), \tag{3.12}$$

$$S_t = g_t^I \odot f(W_1 x_t + W_r h_{t-1} + b_1) + g_t^F \odot s_{t-1} \tag{3.13}$$

$$g_t^O = \sigma(W^O x_t + W_r^O h_{t-1} + s_{t-1}) \tag{3.14}$$

$$h_t = g_t^O \odot tanh(S_t) \tag{3.15}$$

In the set of equations above, 3.10 is the mathematical computation inside the input gate of LSTM where W denote input weight matrices and W_r represents recurrent weight matrices. The σ represents the logistic sigmoid function which gives this gate's output, a number between 0 and 1. 3.11 represents the mathematical calculations in the forget gate. 3.12 represents the calculation which updates the state captured information. Again, using a sigmoid function, equation 3.13 gives the memory state output based on the

---

[5] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

previously captured information, output of the previous unit and its input. The value calculated by 3.13 will be then used to calculate the final output of the LSTM unit which is represented in equation 3.14 where $\odot$ represents point-wise multiplication.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a powerful architecture motivated by variants of Multi-Layer Perceptrons (MLPs) and is emulating the behavior of a natural visual cortex. CNNs are feed-forward neural networks which contain one or more convolutional layers. Neurons in this architecture are purposely spatially arranged to form feature maps. Each of these neurons has a connection to a fixed-size local region of the input which corresponds to its position in the map and weights are shared among all neurons in the map. The output which feature map computes is interpreted as a convolution of its input with a small filter kernel followed by an element wise nonlinearity.

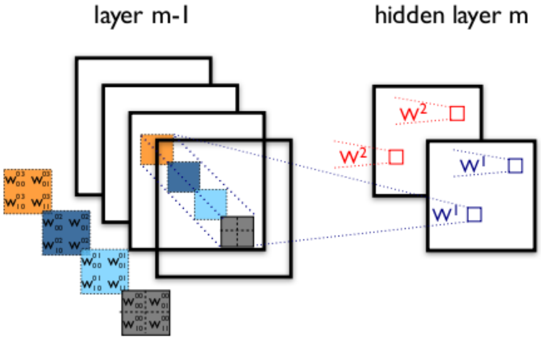


Figure 3.20:
Example of a convolutional layer.[6]

What the filter kernel does, is to match the features with each piece of the given input by comparing the pixels one by one. Compared to a fully connected layer, the outcome given by the feature map is spatial layout of the input data which has noticeably lower number of trainable parameters. A convolutional layer can be followed by a Pooling layer. This layer sub-samples each feature map by retaining e.g. only the maximum value in non-overlapping usually 2x2 or 3x3 pixel cells in order to reduce the size of the data. It also introduces some translation invariance. For the classification task, a fully-connected network is the ending of the computation chain of CNN. This part integrates information among all feature maps of CNN layers. However to use as an onset detector, giving binary labels to specify onsets and non-onsets, the CNN is trained on spectrogram excerpts centered on the frame for classification.
Figures 3.20 and 3.21 illustrates a convolutional layer and also an example of a CNN

---

[6] http://deeplearning.net/tutorial/lenet.html

model, respectively. In figure 3.21, as depicted, the input of the network is a feature matrix (here taken from a 2D spectrogram). At the end, there is a fully connected layer which maps the learned features from the convolutional layers to the out put layer.

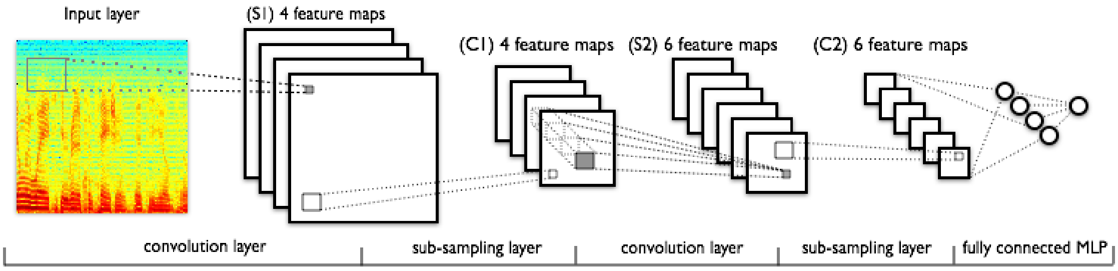


Figure 3.21:
An overview of a convolutional neural network model.[7]

In the work of [SB14] As the task of signal processing mostly involves discovering changes over time, filters wide in time and narrow in frequency was used and since results of high time resolution was required, max-pooling over only frequencies was performed. With different window sizes and same frame rate, training was on a stack of spectrograms reduced to the same number of frequency bands with logarithmic filter banks. This way, each neuron combines information of high temporal and high frequency accuracy for its location. In order to detect onsets in a signal, he spectrograms was computed in their work and was fed to the network which onset activation function over time was obtained. Using a hamming window of 5 frame, the function was smoothed and local maxima higher than a given threshold was reported as onsets.

3.3.3 Parameters

As for the parameters, in each section of the experiment using different techniques and algorithm, require different parameters to set. In this section, the parameters used for each phase of the work is briefly explained. Afterwards, in the next chapter, the values set for each parameters is specified.

Input Representation parameters

The parameters used for this phase, depending on the feature extraction method are presented as follows:

---

[7] http://deeplearning.net/tutorial/lenet.html

The fast Fourier transform size: Defines the number of bins used for dividing the window. Each bin is a spectrum sample which defines the frequency resolution of the window[8]. Therefore, the larger the FFT size, the higher the frequency.

Overlap: this parameter determines how many percentage is desired in order to hop to the next window. The hop size is then calculated as

$$hopsize = fft\_size \cdot (1 - overlap)$$

.

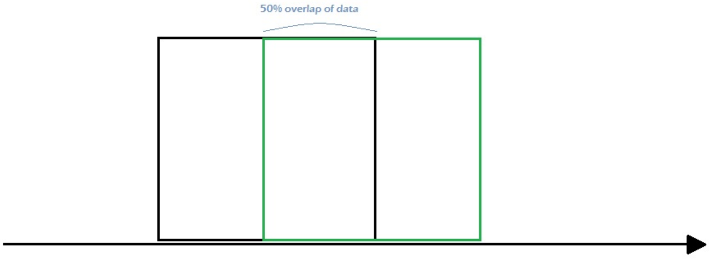


Figure 3.22:
Overlap between two windows.

In the figure 3.22, the first analysis window (shown in black) is an FFT_window which contains some information about the signal. By setting the overlap value as 0.5, we are determining the hop size as half of the FFT size, therefore the next window (shown in green) will move only half of the window size forward which leads to have half of the information in the green window to be the same as the previous window. If the value of overlap is set to be as 0.25, then the hop size is 0.75 of the FFT size which leads to have only 25% of the information in the next window to be the same as the previous window.

Model Parameters

The parameters used for building a deep learning model, compiling and training are described in the following;

Units: Neurons or units if using the RNN model, if using the CNN model is called filters in each layer and using an activation function they calculate a weighted sum of their inputs, and adds a bias to it. To choose a value for this parameter, we need to look at the size of the data. The larger the number of neurons and layers results in a larger model and more parameters to calculate. If the size of dataset is smaller than the size of the model, the model will have numbers of neurons which are not used in calculations [CGCB15].

Activation function: A neuron needs to be either activated or not. This decision relies on the value which neuron calculates. Here, the activation functions used in this work are

---

[8] http://support.ircam.fr/docs/AudioSculpt/3.0/co/FFT%20Size.html

described in the following. These function were chosen based on their popularity among the activation functions.

Sigmoid function
Sigmoid activation function is a nonlinear function. The form of this function is shown in the equation 3.15.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.16}$$

It takes a real number x as its input and returns a value between 0 and 1. Sigmoid function is a particular case of the logistic function and the output of it can be interpreted as the probability value. Sub figure *a* in figure 3.23 depicts the shape of this function.
Tanh function

The tanh activation function is a scaled sigmoid function. The equation 3.16 defines the form of this function shown in the sub figure *c* in 3.23.

$$\sigma(x) = \frac{1}{1+e^{-2x}} = 2 * sigmoid(2x) - 1 \tag{3.17}$$

This activation function is also nonlinear and it is bound to range $(-1, 1)$. The different between the tanh function and the sigmoid function is the strength of their gradient. As illustrated in the figure below, the gradient of a tanh function is steeper than the sigmoid function which depending the importance of the gradient's strength, one can decide whether to use a tanh function or a sigmoid function. However, as also mentioned in the earlier section of this chapter, where the problem of vanishing gradient was explained, both these activation functions cause the vanishing gradient.

Rectified Linear Unit (relu)
As it was mentioned in the other section 3.3.2, ReLu is one of the solution for vanishing gradient problem. The equation 3.17 shows the form of this function.

$$A(x) = max(0, x) \tag{3.18}$$

For the positive value of x, this function return x and for the negative value of x it returns 0. The shape of this function is illustrated in the sub figure b in 3.23.

Dropout: Dropout is one of the most used tools in many deep learning models as a way to avoid over-fitting [43]. The key idea behind this theory is to randomly turn off or 'drop' units along with their connections during the training process, in the layer which it is applied. This act prevents the units from co-adopting too much which leads to over-fitting. Dropout is also a form of a regularization of the model. [SHK$^+$14] depicted dropout technique as shown in the figure 3.24. Note that setting dropout to 50 means the 50% of neurons in each layer will be randomly shut down, namely, 50% of the calculated information will not go through the next layer. In this thesis, the value of the dropout is

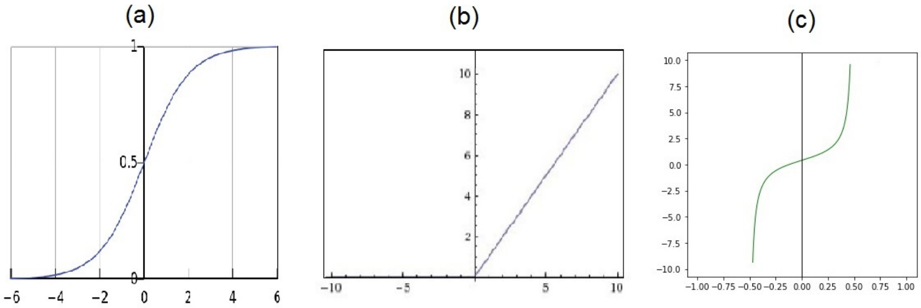


Figure 3.23:
Sigmoid activation function

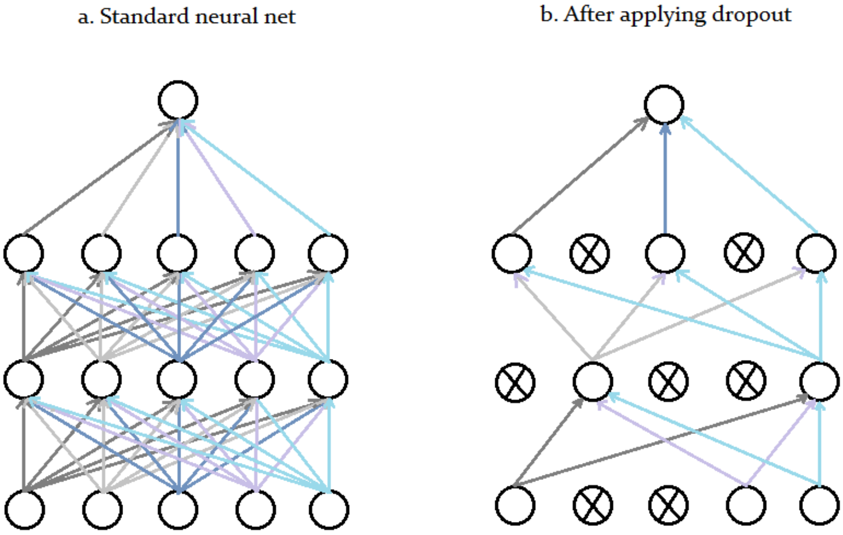


Figure 3.24:
Dropout method applied in the layers of a fully connected neural network.

set after visualizing the learning curve and the prediction results to lower the model's over learning and improving the predictions on the evaluation set.

Loss function: Loss function or cost function maps one or more values onto a real number representing some cost associated with the values. A loss function $l(\hat{y}, y)$, where $\hat{y}$ is the predicted value of an event and y the actual value of the event aims to lower the difference between the predicted output value and the actual output value. This function is used in order to compile a deep learning model. For this work, these two loss function were used: categorical cross entropy and binary cross entropy. The first one is used for categorical prediction, which in this work is task 3 of the DCASE 2017. The latter is used for binary prediction and in this work, it was applied to compile the models for the task 2 DCASE 2017.

Optimizer: As mentioned in the previous part, a loss function needs to be minimized. Therefore the usage of an optimizer is needed. The optimizers used for this work are from the category of the most popular ones and are briefly explained. The optimizer is chosen based on the experimental results. In some works optimizers such as adam and PMSprop are recommended for RNN and its extensions [CV17]

Stochastic Gradient Descent (SGD)
The gradient descent algorithm minimizes the loss function $l(\theta)$by updating the parameters as shown in the equation 3.18.

$$\theta = \theta - \alpha \nabla \theta E[J(\theta)] \tag{3.19}$$

Using the full training set, gradient descent algorithm approximates the cost and gradient. However, SGD simply computes the gradient of the parameters over only a single or a few training samples in the set. The equation 3.19 shows the form of the SGD. The pair $(x^(i), y^(i))$are the random choices from the training set.

$$\theta = \theta - \alpha \nabla \theta J(\theta; x^{(i)}, y^{(i)}) \tag{3.20}$$

RMSprop
RMSProp is the combination of Rprop and SGD. Resilient Propagation (Rprop), uses adaptive learning rate approach which is by increasing the learning rate multiplicatively for a weight only if signs of two previous gradients agree. Otherwise it decreases the learning rate multiplicatively.
RMSprop divides the learning rate by an exponentially decaying average of squared gradients and uses the equations 3.20 and 3.21 to update the parameters.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \tag{3.21}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} \tag{3.22}$$

where $E[g^2]_t$ is the exponentially decaying average of the squared parameter updates.

Adam
Adaptive moment estimation (adam) computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients [44]. In the equations 3.22 and 3.23 calculations of the first moment m_t (the mean) and the second moment v_t (the un-centered variance) of the gradients are shown.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.23}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{3.24}$$

Adam updates its parameter just as RMSprop using the following equation 3.24 as the update rule.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \tag{3.25}$$

where $\hat{v}_t$ and $\hat{m}_t$ are calculated as follows:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{3.26}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{3.27}$$

Which are the computations of bias-corrected first and second moment estimates. The default values for β_1 is 0.9, for β_2 is 0.999 and for ε is 10^{-8}. in practice, this optimizer has shown very good performances compared to other adaptive learning method algorithms such as SGD and RMSprop [KB15].

Kernel size: This parameter is only set in a convolutional layer which determines the largeness of the convolutional window. This window replicates how human eyes function when looking at an image. First a small region is picked and then the window slides to capture the information of the neighboring regions. The length of sliding is also determined by setting the strides parameter. The kernel size is normally set based on practice, however [AHMJ+14] recommended to start by small sized such as kernel size 3x3.

Max pooling layer: This parameter is also set only for CNN architectures and maps the dot products calculated by the kernel size, into one single integer which is the maximum value within the determined pooling size. The same as kernel size, the value set for this layer is determined by practice [Wu17].

Early stopping: This is a technique which is considered another solution to over learning and reduces the execution time while training a model with an iterative method. As the name describes, early stopping of the models training is applied when no change withing some pre defined number of iterations (epochs) are not observed. For this we define a patience, i.e. the number of epochs to wait before early stop if no progress on the validation set. The patience is often set somewhere between 10 and 100 (10 or 20 is more common) [PHH+17, PHH+17, LD17a].

In the next chapter, the chosen values for all the mentioned parameters is written and also explained.

---

[8] http://www.hpmemoryproject.org/an/pdf/an_243.pdf

CHAPTER 4

Experimental Results

In this chapter, steps of the practical part of this work are presented. As mentioned in the first chapter, sound event detection is a task which can be classified into two categories: multi label classification which addresses the issue of detecting overlapping sound events and multi-class detection such as addressing the issue of detecting a rare sound event. Different architectures are tested and methods such as dropout and data augmentation are applied to generalize the model and improve the performance. To evaluate the models, two datasets provided by [MHD$^+$17] have been used.
The first part of this chapter, presents these datasets. Afterwards, the systems which have been developed for this thesis as well as frameworks and libraries used for the application is introduced. Furthermore, the phases of processing the data, extending training dataset and creating different models to find the best fit for the data are described.

4.1 Datasets

The evaluation presented in this chapter is based on two experiments: experiment 1, uses the dataset for DCASE 2017 task 2 which is rare sound event detection with no multi label tagging. Experiment 2, is the task of sound event detection with overlapping sound events which the dataset for DCASE 2017 task 3 is used.

4.1.1 DCASE 2017 Task 2 dataset: Rare Sound event detection

The dataset used for the phase one of the project is given by [MHD$^+$17] for the second task. This dataset has two parts, development dataset and evaluation dataset and they both comprises of isolated sound events for each target class. These classes are babycry, glassbreak and gunshot. The recordings of everyday acoustic scenes are also part of the dataset.
Background audio material consists of recordings from 15 different audio scenes, and is

part of TUT Acoustic Scenes 2016 dataset.

The isolated sound examples are collected from the website "Freesound"[1] and the selection of the sound was based on the exact label and had sampling frequency higher than 44.1kHz. The annotation of the start and end time of isolated examples in the training set were created by a SVM-based semi supervised segmentation. Afterwards, for correctness, a manual refinement was also applied.

The event-to-background ratio (EBR) [MHD$^+$17], randomly selected positioning for the target sound and also the probability of event occurrences (there are mixtures where no event is occurred in them) are counted as the parameters controlling synthesized material. Isolated sounds and background samples are selected at random. The mixture synthesizer automatically produce annotations for the synthetic mixtures. Since the task is independent of the background acoustic scene, these annotations contain only the target sound event temporal position.

As mentioned before, there are two subsets: the evaluation and development dataset. Development dataset comprises the original background, isolated event samples and a set of generated mixture audio tracks. The background audio file in this dataset are approximately 9 hours and for each target class, around 100 isolated sound examples and 500 mixture audio samples are provided. Each of these mixtures contains zero or one target sound.

For the system's evaluation during the development phase, 500 mixtures per target class are also provided which different background audio and sound examples was used to create these mixtures. Using original audio material not distributed in development set, the evaluation set is produced the same way. As mentioned in the challenge description, the evaluation metric for the task is event-based error rate calculated from one second length segments and is used for ranking the submitted systems.

---

[1] https://freesound.org/

4.1.2 DCASE 2017 Task 3 dataset: Sound Event Detection in Real life Audio

The difference between this dataset and the one for second task is that, this dataset provides audios for sound event detection task with multiple possibly overlapping audio signals originating from different sources (e.g. baby cry, glass break and gunshot). Namely, similar to everyday life situation, the sound sources are rarely heard in isolation. Another thing which makes the third task more complicated than the second task is that there are no control over the number of overlapping sound events at each time, not in the training nor in the testing audio data. The background audio used in this dataset is part of the Acoustic scene dataset used for task 1 DCASE 2017[2] which are recordings of street acoustic scenes with various levels of traffic and other activity. The length of each audio provided in this dataset are between 3 to 5 minutes. The recording used sampling rate 44.1 kHz and 24 bit resolution. To annotate the audios, the labels were chosen freely (which resulted in a large set of raw labels) by the same person who recorded the audios. These labels are divided into two groups; the noun-labels which characterize the sound source, and verb-labels for the sound production mechanism (using a non-verb pair if possible). Annotation of the offset and onset of the events was performed manually.
By creating a mapping from the raw labels, sounds were merged into classes described by their source, i.e. “car passing by”, “car engine running”, “car idling”, etc are under the label “car”, sounds produced by buses and trucks are under the label “large vehicle”, “children yelling” and ” children talking” are under the label “children” and so on. Final event classes, selected by the providers of this dataset, are:

- brakes squeaking
- car
- children
- large vehicle
- people speaking
- people walking

Three persons other than the annotator, listened to each of the audios in this data set as well. Agreement and disagreements of their annotating process were not on the onsets and offsets, rather on the occurrences of the presence of sound in each labeled audio segment. Similar to the task 2 dataset, task 3 dataset also comprises 2 parts; development and evaluation dataset. The event subclasses in each different classes are distributed unevenly in each recordings, therefore, the partitioning of individual classes can be controlled only to a certain extent.
A four fold cross validation was provided which made each recording exactly used once as the test data and Segment-Based Error rate is used for the model evaluation.

---

[2] http://www.cs.tut.fi/sgn/arg/dcase2017/index

4.2 Framework

Over the recent, the popularity of deep learning models have increased and therefore several deep learning software frameworks have appeared to efficiently develop and implement these methods. As a number of such frameworks, Theano[3], Caffe[4], Neon[5], Tensorflow[6] and Torch[7] can be named.
The framework used to implement the SED application is chosen to be Theano and it is briefly explained. A summary of the other libraries and system language used for the practical part of the project is also given.

Theano

Theano is one of the deep learning frameworks which is used as a Python library for optimization and evaluation mathematical expressions, especially matrix-valued ones. This library has a close integration with NumPy, another python library for scientific computing. Using GPU, theano performs the computation process on large amount of data much faster than using CPU. Function derivations are with one or more inputs and it has an extensive unit-testing and self-verification[8]. For this work theano is used as backend. The motivation was to support rapid development of machine learning algorithms. The name Theano comes after a Greek mathematicians, who was possible Pythagoras' wife [BBB+10].

System Language and Libraries

The language used for this work is Python version 2:7:13 [Pyt09]. Python is an "easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object oriented programming [pyt]".
Other used libraries are listed below:

- Numpy[9]: Numpy version 1.12.1 is used for the experiments. This package is used for scientific computing. This library is also used as an efficient multi-dimensional array of generic data and creates the feature vectors from the input representations. Numpy's arrays compactness, accelerate the mathematical calculations practices on the data.

---

[3] http://deeplearning.net/software/theano/

[4] http://caffe.berkeleyvision.org/

[5] https://github.com/NervanaSystems/neon

[6] https://www.tensorflow.org/

[7] http://torch.ch/

[8] http://deeplearning.net/software/theano/

[9] http://www.numpy.org/

- Scikit-learn[10] version 0:16:1: [PVG+11] This library is a simple and yet efficient tool for data mining and analysis which is developed for Python. This package is used to scale the data and calculate the confusion matrix.

- Keras[11] version 2.0.9: An open source high level deep learning library designed for Python. This library abstracts from the underlying backends and provides simple implimentation, traing and apply models. For this thesis, the sequential keras layers were used to build the RNNs, LSTMs and Bidirectional LSTMs [HWT+16] and the convolutional layers for CNN architecture [Wu17].

- Librosa[12] 0.5.1: [MRL+15] Librosa is a python package for music and audio analysis which offers commonly used functions such as spectrogram calculations and frequency conversions to represent audio signals. For this work, the melspetrogram function, MFCC [GAFC+16] and CQT [BP92] functions of librosa were used to test and compare the different input representations.

4.3 Hardware Specifications

For all parts of the experiments that are conducted in this project, a single computer with the specifications mentioned in Table 4.1 is utilized. Thus, all represented running times of different algorithms or different runs are measured on the stated hardware system and may differ if tested on different specifications.

Component	Details
Processor	Intel(R) Xeon(R) CPU x5680
GPU	GForce GTX Titan X
Operating System	Linux Ubuntu 16.04

Table 4.1:
Hardware specifications on which the experiments of this project are conducted.

4.4 Experiment Results

In this section, the experimental results of the project are presented. In each sub-section, approaches used for signal processing are mentioned as well as the learning part along with the outcome. At the end, a comparison is made to magnify the differences among the approaches and a conclusion is made base on the analysis of the comparison.

---

[10] http://scikit-learn.org/stable/

[11] https://keras.io/

[12] https://librosa.github.io/librosa/

4.4.1 Overview of Experiments

To analyze the performance of SED systems, the input representation and the model architecture are important factors. Therefore different input representation on different architectures are tested and the result for each experiment is presented in the two following sections in more details. Due to their recent popularity [PKBGM17, GSR^{+}17], chosen models are RNN, LSTM, BLSTM and CNNs which have has been explained in the theoretical part of this thesis in 3. The sequential layers of RNNs and its extensions, LSTMs and BLSTMs give them the capability of exploiting temporal dependencies in audio which is why, they are widely used in speech recognition tasks [PKVK13].
Another network that proved robust in classification tasks is CNN architecture [SB17] which is suited to exploit image-like log-Mel spectrograms to learn and identify both high-level and low-level features. Choosing the parameters for these networks, are mostly experimental such as learning rate and optimizers. However, some parameters such as defining the size of networks with number of neurons and layers depend on the size of data [VBGS17]. As large models have larger number of parameters to compute, they do not perform well on small datasets which require smaller number of parameters to compute. therefore, to pick the size of model, parameters computed by the model should be smaller than the size of dataset. More detail explanation of parameter setting is presented in the following sub chapters, where the analysis of experimental results on each dataset are provided. The overview of these experiments in table 4.2 and 4.3 have shown that models performed noticeably better on rare sound event detection (detecting only one event on a given audio signal). One reason for that is the precise annotation of the datasets. Rare sound event dataset, each audio signal is a result of synthetic generated mixture and the detected event boundaries are therefore precise. The second dataset comprises randomly recorded audios from streets and the annotation was done using a support vector machine and then was checked manually. Also, for the second dataset, the model is trained to recognize the presence of multiple events at once. The highest performances are highlighted in each table.

As shown in table 4.2, CNN and LSTM architectures achieved lower error rate and higher F-score than the other tested architectures as well as both baseline systems. It is worth to mention that these results are after applying data augmentation and dropout technique which decreases model's over learning and helps to improve the model by providing more training samples. Through these experiments, we have observe the difficulty of detecting the gunshot which was due to different types of gun sounds. As the length of each sound event is different from one another (the average length of event baby cry is much longer than glass break and gunshot), we needed to find a comprise on the segment length in a way tat it captures information of all the events, longer segment length increased the detection accuracy of baby cry but reduced the glass break and gunshot. Smaller segments resulted in higher accuracy on detecting the glass break and lower on baby cry and gunshot. More detail description of the experiments are explained in experimental results of rare sound event detection subsection.

Overview of the RSED Experimental Results						
Models	MLP		SVM		RNN	
Classes	Err	F1	Err	F1	Err	F1
Babycry	0.67	72.00%	0.37	64.60%	0.41	77.03%
Glassbreak	0.22	88.50%	0.31	70.36%	0.38	78.91%
Gunshot	0.69	57.40%	0.63	34.43%	0.60	59.34%
Average	0.53	72.70%	0.43	54.95%	0.46	71.85%
Models	LSTM		BLSTM		CNN	
Classes	Err	F1	Err	F1	Err	F1
Babycry	0.27	77.84%	0.40	69.43%	0.24	83.17%
Glassbreak	0.34	81.05%	0.33	76.27%	0.24	84.17%
Gunshot	0.53	69.53%	0.69	41.47%	0.44	58.04%
Average	**0.38**	**76.16%**	0.47	62.34%	**0.30**	**75.12%**

Table 4.2: Rare Sound Event Detection Experiment Overview: Event-based overall metrics (onset only, t-colar=500 ms).

As mentioned before, these models are further tested on a real life street recording dataset and the results are shown in table 4.3. The overall observation on the real life street sound dataset experiments have proved the complexity of polyphonic sound event detection in comparison to the monophonic. As explained previously, the target classes on this dataset are people walking, people speaking, children, car, large vehicle and sound of brake squeaking. After multiple experiments using different set of parameters for signal processing and model's hyper parameter, we noticed the difficulty of detecting the brakes, children and people speaking. Brake squeaking and children were hardly ever detected. Car, large vehicle and people walking had better chance of recognition by the models but te were still always below average segment-based f_score of 50%. This shows that the model needs more advanced signal processing techniques to boost the presence of these events within the recordings. However, in comparison with the baseline models, the deep models had lower error rate which shows the lower number of False positives these models achieved comparing to the baseline models.

4.4.2 Experimental Results on Rare Sound Event Detection

The approach used to detect the rare sound events from an audio signal is the state of the art and commonly used method "detection by classification". Namely, by applying number of pre-processing steps on the dataset, inputs of the network are then pieces of the audio signal which the network learns which pieces contain an event and which do not contain an event. The step by step process used to run these experiments are explained in the following subsections.

content of DCASE 2017 development dataset was already explained at the beginning of this chapter. After reading all the tracks, a segmentation method on each audio signal

Overview of the real-life SED Experimental Results

Models	MLP		SVM		RNN	
Classes	Err	F1	Err	F1	Err	F1
People Walking	1.44	33.5%	1.40	13.33%	0.93	07.90%
People Speaking	1.29	3.6%	1.31	4.5%	1	0.0%
Children	2.66	0.0%	2.4	0.95%	1	0.0%
Car	0.76	65.01%	0.95	19.54%	0.8	20.52%
Large Vehicle	1.44	42.07%	0.00	7.01%	0.98	1.59%
Brake	0.98	4.1%	1.86	1.31%	1	0.0%
Average	0.93	42.08%	1.28	8.12%	0.98	22%

Models	LSTM		BLSTM		CNN	
Classes	Err	F1	Err	F1	Err	F1
People Walking	0.89	13.53%	0.91	15.94%	0.92	9.29%
People Speaking	0.92	12.21%	0.95	6%	0.97	2.71%
Children	1	0.0%	0.99	1.11%	0.98	2.88%
Car	0.63	42.35%	0.7	34.35%	0.74	28.82%
Large Vehicle	0.87	14.66%	0.91	9.71%	0.00	00.00%
Brake	1	0.0%	0.97	3.23%	1	0.0%
Average	**0.79**	**41.09%**	0.93	31.72%	**0.77**	**28.61%**

Table 4.3: Real Life Street Sound Event Detection Experiment Overview: Segment-based overall metrics.

was applied. The motivation for segmenting the signal is to reduce the size of network's analysis window which improves the accuracy [CKBK16, PHV16, SGB$^+$15].
In order to segment the audio signal, first the spectrogram of audio wave data is calculated. Then, in case an event occurred within the audio, onset and offset position of the event in spectrogram is calculated. Then each of these smaller pieces of spectrograms were divided into number of fixed size segments (the segment size refers to the number of frames in the given segment duration i.e 0.5 second as segment duration). If no event is occurred in the audio, a random piece of the spectrogram with the average (mode) length of the event is chosen and segmented. This will keep the balance between the distribution of segments with no events and segments with events in the input dataset to the deep model. Figure 4.1 depicts this process.

Figure 4.1 illustrates the segmentation process of two different spectrograms. The calculated spectrogram has the occurrence of an event "babycry". The segmentation function starts from the beginning of event and finishes at the end of event. The ending of event is always inside of the last segment. Sub-figure b illustrates a calculated spectrogram from an audio which no labeled event is occurring within it. Therefore, the segmentation function chooses a random place to start segmenting the spectrogram. The number of iteration for such spectrograms depend on mode of the event lengths. Here, most events

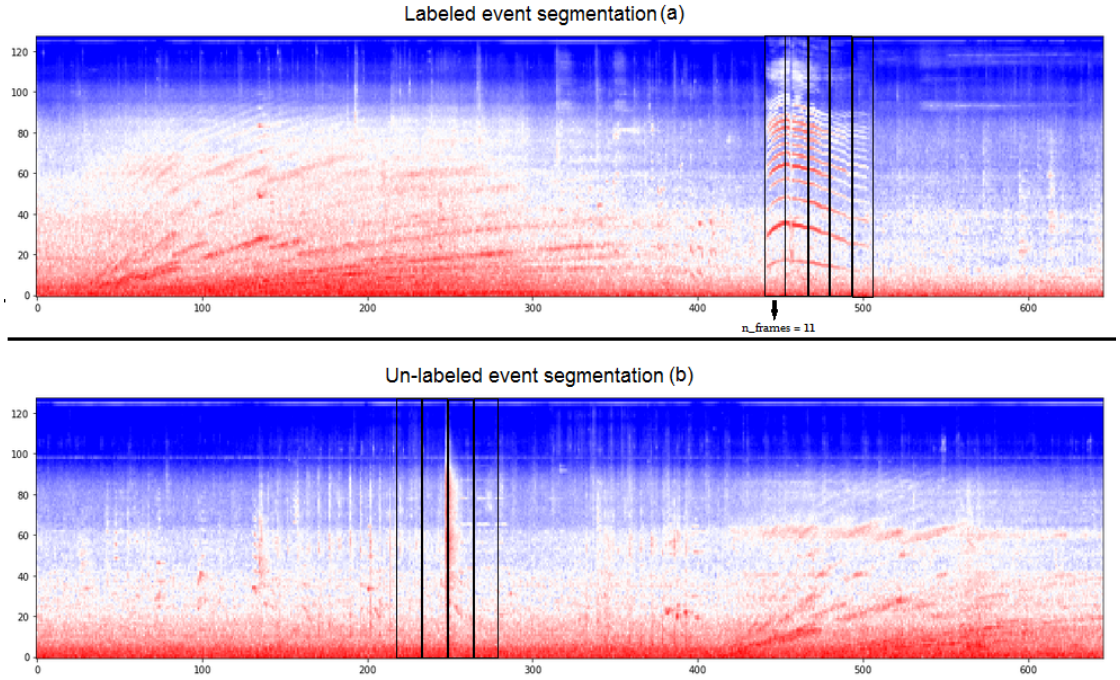


Figure 4.1:
Segmentation process of two different spectrograms.

had the length 4 seconds. therefore only 4 segments are picked from the audios with no events. In this case, the train dataset remains balanced. The size of segment here is calculated as 11 frames (0.5 second) with FFT_size as 4096 with 50% overlapping. Using the scikit-learn package, the segmented spectrograms are then standardized to zero mean and unit variance. To prepare the data as a valid feature matrix for RNN networks, the data is then a Numpy array with the shape (samples, timesteps , features). In the following of this section the different parameters such as FFT_size and mel bands are tested and analyzed.

Post-processing

In order to detect the onset and offset of the event, model predictions need to be processed. Therefore, all the probability values of each segment, needs to be listed to complete the whole audio signal again. Afterwards, where the value of the probability is higher than the defined threshold, the segment will be considered an active event, and where the value of probability is below the threshold, the segment will be labeled as no event. The figure 4.2 shows an example of a model's prediction the predicted probability of segments are listed next to each other to create the audio recording and where the probability is higher than 0.5, presence of an event (in this case a baby cry) is detected.

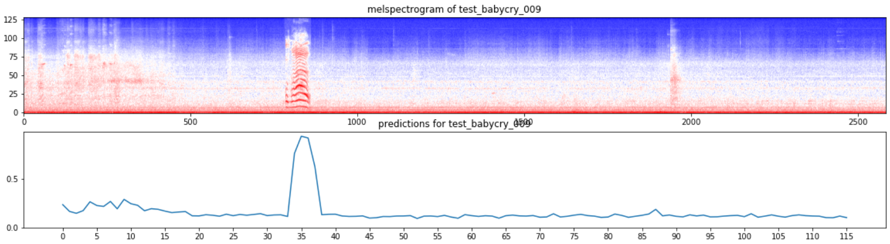


Figure 4.2:
Probability distribution through an audio signal.

Model Evaluation

To Evaluate the performance of deep learning architectures for SED task, evaluation metrics event-based error rate and f-score are used. The system ranking for the challenge is based on the event based error rate. The calculation of these metrics are presented in [MHV16a] where the f-score is calculated for all the detected events and is shown in the equation 4.1 and error rate calculation is shown in the equation 4.2. TP and FP refer to True and False Positives, TN and FN refer to True and False Negatives.

$$f1 = \frac{2 \cdot TPs}{2 \cdot TPs + FNs + FPs} \tag{4.1}$$

$$error_rate = \frac{S + D + I}{N} \tag{4.2}$$

where SS is the sum of substitution errors $(min(FP, FN))$, I is the sum of insertion $(max(0, FPs - FNs))$, D is the sum of deletions $(max(0, FNs - FPs))$ and N is the number of active events in the audios.

Baseline Systems

In order to evaluate the deep learning models, a baseline system is needed to determine the performance of these deep models in comparison to the state of the art machine learning algorithms. for that two different baselines were chosen; one is a Multilayer Perceptron model, provided by the DCASE challenge 2017 team [MHD$^+$17] and the other method is a support vector machine. The detail of these both system's results are in the following. DCASE 2017 Base line system:
The base line system which was provided by the DACSE 2017 challenge is a Multilayer Perceptron (MLP) model and provides a simple entry-level approach but still close to the state of the art systems to give a reasonable performance for the rare sound event detection task. In this approach Log Mel Band energies in a 40 ms window with 20 ms hop size were extracted from the audio signals. The MLP model has 2 layers with 50 neurons in each and 20% of dropout between each layer. A model is provided to detect three different events (babycry, glassbreak and gunshot) separately at a time. The results are shown in the table 4.4 below.

Support Vector Machine as baseline System: For this work, a conventional machine learning algorithm for classification tasks called support vector machines is chosen to compare the deep learning models' results with. The approach in DCASE 2017 base line system [MHD$^+$17] gave the motivation to also use the SVM models to train each event separately and then take the average results for the event detection. This experiment

DCASE baseline results		
	Error-Rate	F-score
Babycry	0.67	72.00%
Glassbreak	0.22	88.50%
Gunshot	0.69	57.40%
Average	0.53	72.70%

Table 4.4:
DCASE 2017 Rare Sound Event Detection baseline results: Event-based overall metrics (onset only, t-colar=500 ms).

is also only performed on the given trained set and tested on the provided test set (the same as DCASE2017 baseline system) to be used as a baseline system.
The hyper parameters used to train the SVM classifier are the kernel with default value of "rbf" (radial basis function), penalty parameter C of the error term with the default value of 0.1, Gamma with the value of "Auto", decision_function_shape with the default value of "ovr", degree with default value of 3, and tolerance with default value of 0.001. Previous studies showed that using MFCC as features results in better performance of the SVM algorithm [CDY10, CGO06]. The same as DCASE2017 baseline system, the same parameters for features extraction is used for SVM model. The results of SVM baseline system are shown in the table 4.5.
First experiment started with the default SVM parameters. The model achieved average error rate of 0.77 and f-score of 27.91%. The segment classification results with 10 fold cross validations for babycry achieved the 0.7 error rate and 37.41% f-score. The low error rate is because of the very high number False Positives (348/500) which cause a very low precision of 0.23. The class gunshot and glass break had much lower f-score and higher error rate than the MLP baseline system. This is also because of the very large number of False Positives (431/500 FP for gunshot and 360/500 FP for glassbreak) which resulted in very low precision for detected events.
Since the results of SVM on MFCC were not very high, the input representation mel-spectrogram was also tested to observe the performance of SVM on the not so conventional feature representation for onset detection. the table 4.5 showed that this classifier (with its default parameters) performed much better on mel spectrogram with the same feature extraction parameters used for MFCC. This improvement is due to low number of False Positives which for MFCC proved to be very high. For baby cry, the falsely detections were decreased from 348 to 203. For glassbreak and gunshots it was decreased from 360 to 281 and 431 to 324 respectively.
Afterwards, to find the best set of parameters for SVM, a grid search was applied. The chosen set of parameters are the penalty parameter C as 10, 0.0001 gamma and 'rbf' kernel. The results were the average error rate of 0.48 and f-score of 54.95%

SVM baseline results						
	MFCC		Mel-Spectrogram		mel grid-search	
	Err	F-score	Err	F-score	Err	F-score
Babycry	0.70	37.41%	0.53	58.17%	0.43	64.60%
Glassbreak	0.73	31.63%	0.41	70.95%	0.37	70.36%
Gunshot	0.88	14.70%	0.71	31.35%	0.70	34.43%
Average	0.77	27.91%	0.55	53.49%	0.48	54.95%

Table 4.5: Rare Sound Event Detection SVM baseline results: Event-based overall metrics (onset only, t-colar=500 ms).

RNNs

RNNs are meant to learn the temporal behavior of time series data. therefore, this architecture is tested with different parameters. RNN models are trained separately on each event. That is each time, the RNN model will focus only on one event and learn the features of only that event. Then the model is reset and learns the feature of the next event. This approach is motivated by [MHD$^+$17] which transfers a multi-class classification into a binary classification. As mentioned in the third chapter of this thesis, different input representations such as MFCC, Mel-spectrogram and CQT is used as the input of models. The result of onset detection and analysis are shown in the following of this part.
As for mel-spectrograms, in the following experiments, these features were calculated from each audio. Afterwards the segmentation was applied and the models were trained using 4 fold cross validation approach. the rest of parameters are determined in the following table.

The results provided in 4.6 shows the performance of RNNs in 4 experiments, each changing some hyper parameters to observe and compare the results. Chosen values for hyper parameter in these experiments are based on the size of dataset, observations of learning curves and model results.
Hyper parameters such as number of layers, neurons and batch size are directly depended on the size of dataset [VBGS17]. The number of neurons and layers determine the size of model. Therefore, for a small dataset, a larger model is not appropriate as it will require a large number of parameters to calculate which is not matched with the size of dataset. The other hyper parameters such as segment duration (how long each segment should be), size of the FFT-window, optimizer, epoch and learning rate are chosen after number of experiments and comparison of their results.
The value of dropout is chosen based on the number of neurons in each layer. For instance, if the number of neurons are 32, randomly shutting off 20% of them means the information of only 25 neurons are passed on to the next layer. Number of epoch is also chosen after observing the learning rate. If the value for epoch is too low, the model will not improve on learning the data. if it is too high, the model reaches to a point where is does not learn any further, therefore will not improve anymore and it will just slows

RNN results on mel-spectrogram					
Parameters		Exp-1	Exp-2	Exp-3	Exp-4
Augmentation		False	False	True	True
n_fft		2048	4096	4096	4096
overlap		50%	50%	50%	50%
segment duration		0.4 s	0.4 s	0.4 s	0.4 s
n_mels		128	128	128	128
n_layers		2	2	2	3
n_units		32	8	8	8
dropout		False	20%	20%	20%
loss-function		BCE	BCE	BCE	BCE
activation		sigmoid	sigmoid	sigmoid	sigmoid
optimizer		rmsprop	rmsprop	rmsprop	rmsprop
lr		0.0001	0.0001	0.0001	0.0001
batchsize		128	128	128	128
epochs		100	100	200	200
Babycry	err	0.64	0.45	0.41	0.54
	fscore	68.24%	74.06%	77.03%	73.05%
Glassbreak	err	0.56	0.47	0.38	0.31
	fscore	70.56%	70.54%	78.91%	82.58%
Gunshot	err	0.83	0.58	0.60	0.56
	fscore	38.09%	37.50%	59.34%	58.08%
Average	err	0.67	0.50	0.46	0.45
	fscore	58.96%	60.70%	71.85%	71.05%

Table 4.6: RNN results on DCASE 2017 Rare Sound Event Detection evaluation dataset, using mel-spectrograms as input representation.

down the training process.

However setting the early stopping fixes this issue, as it will stop the training when it does not see any changes in the loss or accuracy of the model or any degradation in these parameters after some specific epochs.

The experiments in this table are chosen to show the effect of dropout and data augmentation in learning process of a model. To Augment the data, different shift steps and stretch factors were tested and values which sounded different but still realistic to the audio were chosen and applied on the training set on each fold of the cross validation. Stretch factors were chosen from $\pm 0.5, \pm 0.7$ and Shift steps where chosen from $\pm 2, \pm 4$ to create an augmented dataset with different sizes (up to 8 times larger than the original dataset). In experiment 1, 128 mel bands from the Audio with window length as 2048 is extracted and the log amplitude of these bands is calculated. In order to avoid over fitting, early stopping with choosing patience as 10 on the validation loss value is applied. For this experiment, the dropout parameter is not set so that the result can be compared

with other experiments where the dropout is set with a value. The result of this experiment has a lower error rate of 0.51 than the baseline system which is 0.53. however, the F-score metric here is 58.96%, also lower than the baseline which is 72.70%. This is due to the low precision of this model in comparison with the baseline system. Details of experiment 1 is as follows; Using 4 fold cross validation, the model is each time trained on 3/4 of the dataset and evaluated on the remaining 1/4 to provide reproducibility of the experimental results.
Model precision and recall for baby cry detection are respectively 0.53 and 0.93. This is calculated based on the number of correct and wrong predictions which are 187 (out of 250) true positives and 139 (out of 250) true negatives for correct predictions and 12 false negatives and 162 false positives for wrong predictions. Other than the recordings which contain no labeled events, false positives also contain those wrong predictions where the onset of event is predicted after/before than the tolerance rate which is half a second offset from the beginning of event.
Figure 4.3 shows the prediction results for the experiment 1 of the table 4.6. The model actually detected each occurrence of the events very good (both for babycry and glassbreak, not for gunshot).

The table 4.7 shows the effect of shifting the tolerance each time by 0.25 seconds on the results for the events. Note that, other than the recordings which do not contain any labeled events, false positives also contain the prediction of wrong onsets. This means, the ones that the onset was shifted before/after the tolerance (500 ms) and also the ons which another sound was detected as the event.
The abbreviations TN, FN, TP, FP in the table 4.7, refer to true negatives, false negatives, true positives and false positives relatively. WO stands for number of detecting the wrong onset. Prcs and Rcl refer to precision[13] and recall[14] respectively.

After analyzing the results, it came to notice that the system detected the sound of a rooster, bird singing and running tap water in the kitchen as babycry, therefore in audio tracks where no event was happening but the mixture was created either in a park or in a kitchen, the system detected the mentioned sounds as the crying baby. by visualizing the spectrograms of such signals, the bird sounds had similar features to the babycry, however, the running tab water was not much similar. The similarity of the rooster sound is shown in the figure 4.4.

For glass break, most of the time which the sound of cutlery in a restaurant was occurring in the audio, the model mistakenly detected it as the glassbreak. On the other hand the model could not learn the gunshot features in a louder background environment very well. As illustrated in 4.3 through out the length of the audio, the model detected couple of times gunshot occurrences.
The figure 4.5 illustrates the learning curve of experiment 1 for each events and each folds. At the x-axis, the number of epochs are shown. Note that the experiment had 100 number of epochs. however due to early stopping, the learning process was stopped

---

[13] Precision = TP/(TP+FP)

[14] Recall = TP/(TP+FN)

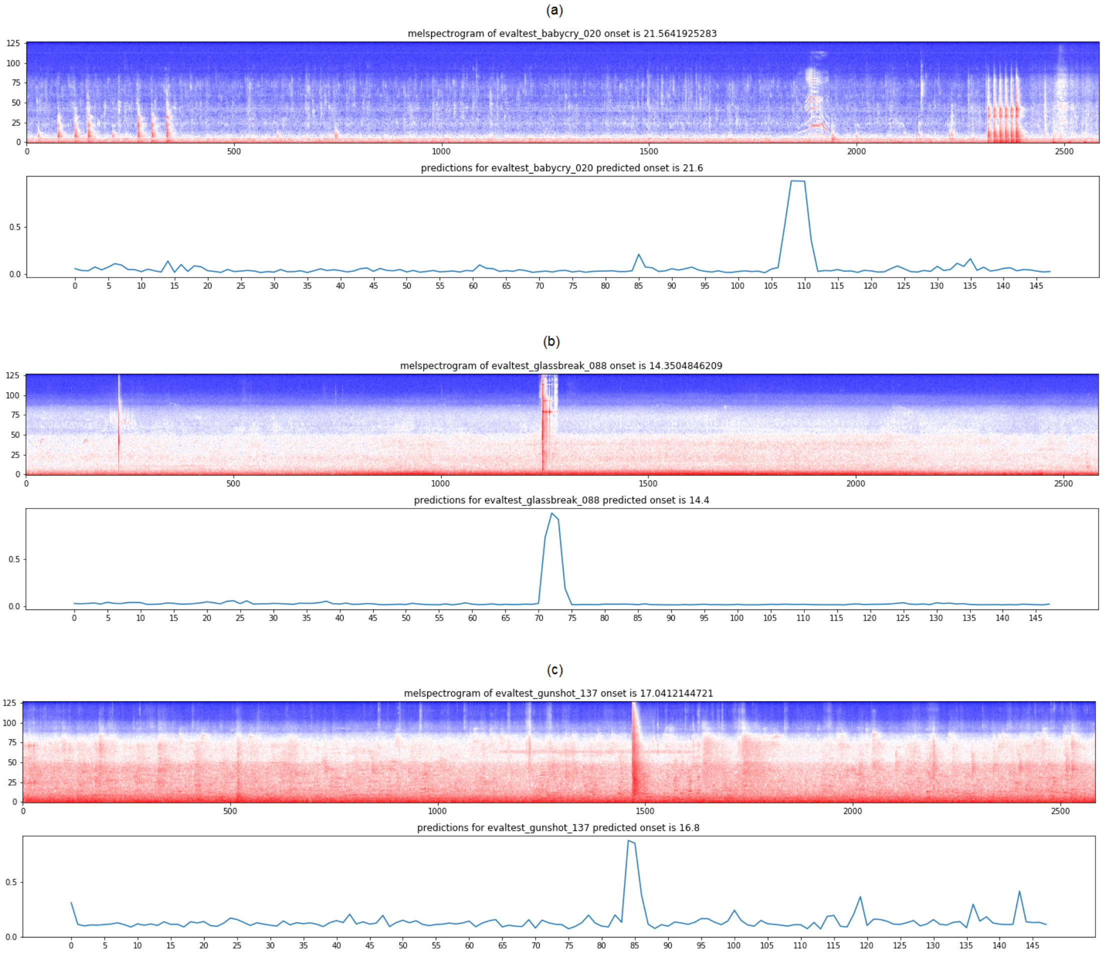


Figure 4.3:
The prediction visualization for RNN experiment 1.

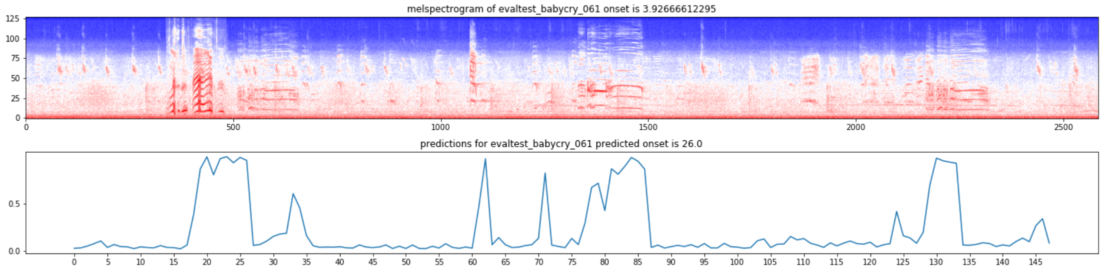


Figure 4.4:
The wrong onset prediction visualization for RNN experiment 1. The model confused the sound of baby cry with a rooster. This is due to the similarity of their features.

babycry	TN	TP	FN	FP	WO	Prcs	Rcl	ERR	F1
0.5 s	139	187	12	162	51	0.5358	0.9396	0.64	68.24%
0.75 s	139	193	12	156	45	0.5530	0.9414	0.62	69.67%
1 s	139	195	12	154	43	0.5587	0.9420	0.61	70.14%
glassbreak									
0.5 s	157	187	15	141	48	0.5701	0.9257	0.56	70.56%
0.75 s	157	190	15	138	45	0.5792	0.9268	0.55	71.29%
1 s	157	191	15	137	44	0.5823	0.9271	0.54	71.53%
gunshot									
0.5 s	126	88	77	209	85	0.2962	0.5333	0.83	38.09%
0.75 s	126	91	77	206	82	0.3063	0.5416	0.82	39.13%
1 s	126	93	77	204	89	0.3131	0.5470	0.81	39.82%

Table 4.7:
Testing the Onset detection results with different tolerances for the experiment 1.

before that. By observing the model's training log, it came to understanding that for glassbreak and babycry, the validation loss did not change (shown with the red color) and for gunshot, after the 10th time reduction increase in validation loss, the condition met and the learning was stopped. Also, the over learning in this image can be observed by looking at the sudden jump within the first 10 epochs. After the jump the value of accuracy did not have a significant change. However, this sudden jump is not observed in the learning curve of the second and third fold of the model, training on the babycry features.

The second experiment in the table 4.6, zooming into the spectrogram (FFT size as 4096, 2 times more than the FFT size in experiment 1) and setting the dropout value to 20%, the model achieved better result than the one with smaller window size. As noticed in the results zooming in the spectrogram had a noticeable effect on babycry and gunshot results but lower on glass break where lowered the error rate which improved the error rate 31% compared to experiment 1 and 5% compared to the baseline system.
The third and forth experiments used similar model as the second one but data augmentation was applied here and which increased the size of dataset to 2 times more (using both pitch shifting with 2 steps and time stretching with factor 0.5). The forth experiment has 1 layer more than the third experiment which yield a slightly different result than the third experiment. By adding the third layer, the detection od glassbreak and gunshot slightly improved. However the babycry weakened. Average of result in forth experiment turned out to be very close/slightly less than the third experiment, in terms of f-score. In the end, result of the model trained with data augmentation was noticeably better than the one with less data to train. Table 4.8 shows the confusion matrix of the experiment 3, with different tolerance of offset.

Figure 4.6 illustrates the learning curve for experiment 3 with 4 fold cross validation.

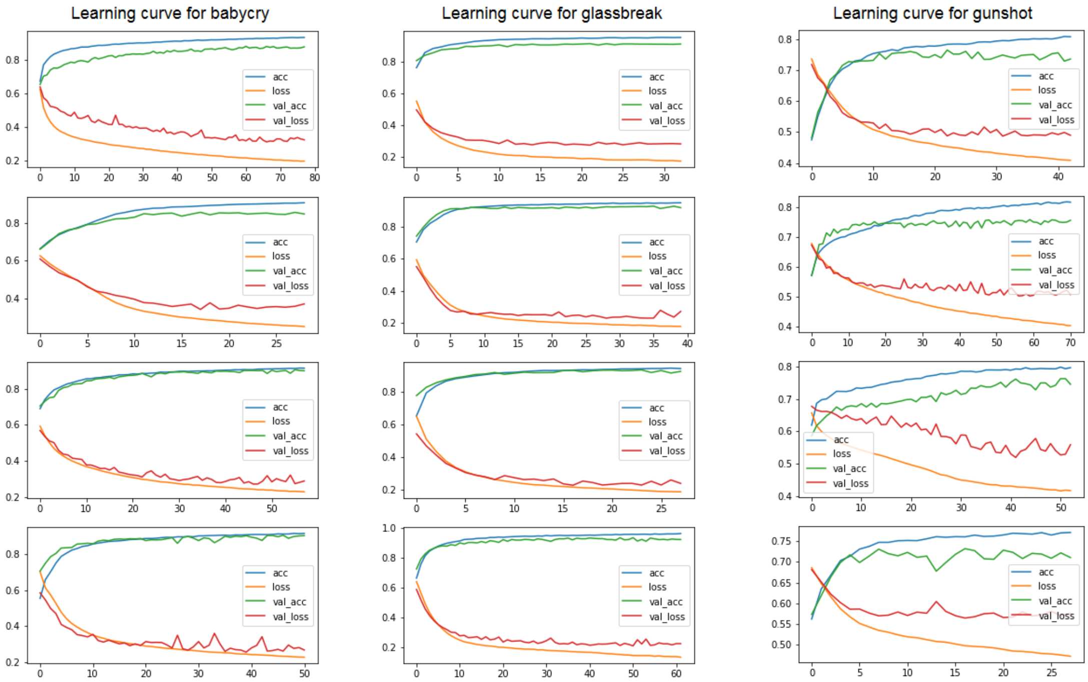


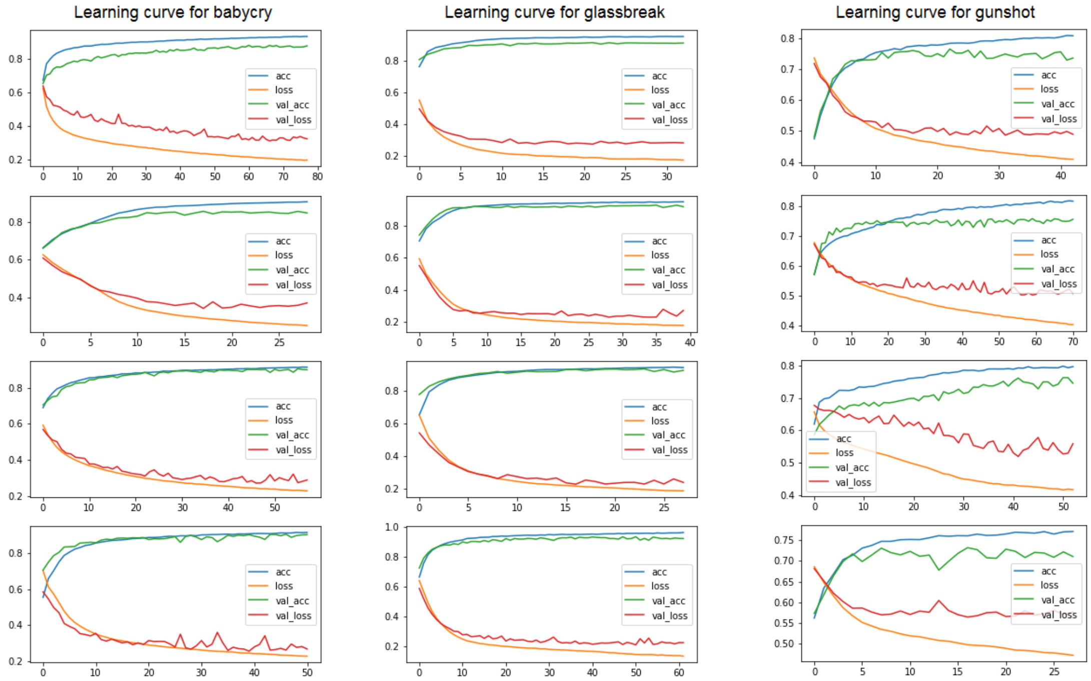
Figure 4.5:
Visualization of the RNN- 4 fold cross validation- for each label in experiment 1.

babycry	TN	TP	FN	FP	WO	Prcs	Rcl	ERR	F1
0.5 s	181	201	15	103	34	0.6611	0.9303	0.41	77.30%
0.75 s	181	208	15	96	27	0.6842	0.9327	0.38	78.93%
1 s	181	210	15	94	25	0.6907	0.9333	0.37	79.39%
glassbreak									
0.5 s	187	204	13	96	33	0.68	0.94	0.38	78.91%
0.75 s	187	206	13	94	31	0.6866	0.9406	0.37	79.38%
1 s	187	208	13	92	29	0.6933	0.9411	0.36	79.84%
gunshot									
0.5 s	126	88	77	209	85	0.2962	0.5333	0.83	38.09%
0.75 s	126	91	77	206	82	0.3063	0.5416	0.82	39.13%
1 s	126	93	77	204	89	0.3131	0.5470	0.81	39.82%

Table 4.8:
Testing the Onset detection results with different tolerances for the experiment 3.

The sudden jump here still exists, However, the curve has a smaller slope than the one in 4.5. By changing the learning rate from 0.0001 (the default value for RMSProp) to

0.001 and 0.00001 the result got worsen. Therefore other optimization functions which were discussed in chapter 3, such as SGD and Adam with different learning rate was also tested. However the result of those experiments where also worsen than the one illustrated in the table 4.6.
Choosing the SGD optimization function for the same model and, same parameters and also data augmentation resulted the average result of 0.72 for error rate and 65.29% for f-score which was caused by very high false positive rate. Choosing the adam optimization function resulted 0.91 for average error rate and 6.03% f-score which was caused by very large number of falsely negative detection. Also different window sizes as well as segment length was tested. The 40 ms segment length and 4096 window size gave the best result among the tested ones. smaller segment size improved the glass break but could not capture enough information on baby cry and gunshot class where have relatively longer duration than glass break.


Learning curve for babycry

Learning curve for glassbreak

Learning curve for gunshot

Figure 4.6:
Visualization of the RNN- 4 fold cross validation- for each label in experiment 1.

As shown in the table 4.6, a simple RNN method with no data augmentation or any noise reduction already resulted lower error rate than the baseline system. Error rate in baseline system provided by the DCASE 2017 is 0.67, 0.22 and 0.69 for babycry, glassbreak and shotgun respectively. The RNN architecture used in the table 4.6 improved the results to

0.41, 0.38 and 0.60 error rate for the classes babycry, glassbreak and gunshot respectively. However, the precision of the RNN models are lower than the baseline system which is 72.70% average of f-score as the best result achieved by the RNNs is 71.05%.

Different tolerances (0.75 and 1 second) also did not noticeably change the overall result of the experiment with tolerance of half a second. For instance, after analyzing the result for babycry, the number of correctly detected onsets (True Positives) within 1 second interval after and before the actual onset changed from 201 to 210 (out of 250 samples).

After changing other parameters such as the segment size, window size, overlap size, and also the model parameters (larger number of neurons, deeper layers), the model still has difficulties to learn the gunshot features.

Different numbers of mels was also tested which the higher the number improved the accuracy of the onset detection. The reason was mentioned in [PHV16] which is the need to look for higher frequency in tasks such as SED. Therefore choosing higher value for mel, captures the high frequency information and improves the event detection. Here 128 was chosen as the mel band value which is also used in [PHV16].

To compare the performance of the RNN with different parameters, the confusion matrix of all the experiments done for this model was compared to see how each set of parameters effect on the prediction. There were some audio tracks which the event was always correctly detected. By listening to those tracks, it was understood that they all had a similar condition, that is, the background noise was very quiet (mostly recordings in a quiet room or office or street). On the other hand there where some tracks which had louder backgrounds in which the detection of the events was much more difficult which changing the parameter caused false positive or wrong onset. This means, either another sound was detected as the event, or the event was detected but was outside of the offset tolerance (half a second). The most effective parameters which had influence on the prediction accuracy where the FFT window size, number of mel-bands, optimization functions and the learning rate.

Another observation is the size of model, the higher number of neurons or deeper number of layers did not improve the results.

Other input representations such as MFCCs and CQTs are also tested and the best results are presented here and compared to mel-spectrogram input representation results. These experiments and their details are illustrated in the table 4.9 and 4.10.

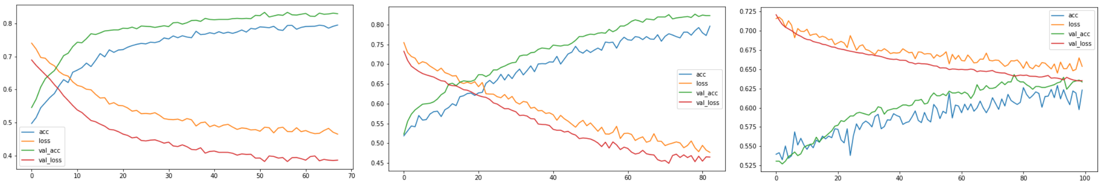
MFCC and simple-RNN Results for RSED			
Parameters		Exp-1	Exp-2
n_fft		2048	2048
overlap		50%	50%
MFCCs		40	128
layers		1	1
units		32	32
activation		sigmoid	sigmoid
dropout		20%	20%
loss_function		BCE	BCE
optimizer		rmsprop (.0001)	rmsprop (.0001)
batchsize		128	128
epoch		100	100
Babycry	err	0.55	0.82
	fscore	53.44%	30.07%
Glassbreak	err	0.54	0.87
	fscore	57.43%	56.76%
Gunshot	err	0.62	0.55
	fscore	25.05%	18.07%
Average	err	0.58	0.62
	fscore	45.30%	50.20%

Table 4.9: RNN results on DCASE 2017 Rare Sound Event Detection Development Dataset, using MFCC as input representation.

As shown in the tables 4.9 and 4.10, both MFCC and CQT as input representation did not catch the important information for the RNN model to learn the events features. After observing and analyzing the results, it came to notice that any loud noise in the audio track was detected as a glassbreak and resulted a very high number of false positives. Another notable point which is observed, is the learning curve of the model on CQT, over learn on the input data and hence the predictions were poor. This over learning can be observed within the first 10 epochs of training for glassbreak. Figures 4.7 and 4.8 illustrate the learning rate of the models using MFCC and CQT respectively as input representation.

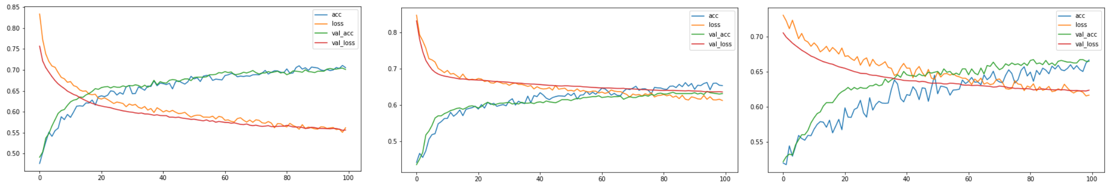
CQT and simple-RNN Results for RSED			
Parameters		Exp-1	Exp-2
n_cqt		120	60
layers		1	1
units		32	32
activation		sigmoid	sigmoid
dropout		20%	20%
loss_function		BCE	BCE
optimizer		rmsprop (.0001)	rmsprop (.0001)
batchsize		128	128
epoch		100	100
Babycry	err	0.52	0.71
	fscore	66.19%	56.56%
Glassbreak	err	0.81	0.73
	fscore	4.26%	13.71%
Gunshot	err	0.72	0.85
	fscore	34.72%	38.58%
Average	err	0.68	0.56
	fscore	35.05%	36.28%

Table 4.10:
RNN results on DCASE 2017 Rare Sound Event Detection Development Dataset, using CQT as input representation.



Babycry Glassbreak Gunshot

Figure 4.7:
Illustration of the learning curves of table 4.9 experiment 1, using MFCC as input representation.



Babycry Glassbreak Gunshot

Figure 4.8:
Illustration of the learning curves of table 4.10 experiment 1, using CQT as input representation.

LSTMs

As mentioned in the third chapter, to improve the performance of RNNs, an extension called Long Short Term Memories (LSTMs) was proposed. In this subsection, the feed forward LSTM architecture and the bidirectional LSTMs are tested and compared. The results of the feed forward LSTMs are shown in the table 4.11.
This table illustrated the onset detection results are using mel-spectrograms as input representation method.

A simple LSTM model already performed better than the simple RNN model with no augmentations (look at the RNN table of results 4.6). Total average of the error rate and F-score of a simple RNN model were 0.67 and 58.96% respectively, while the simple LSTM resulted 0.61 and 65.43%. The lowest error rate achieved in RNN experiments is 0.45 (and 71.05% f-score) as the LSTMs yield 0.38 (and 76.16% f-score). This improvement is mostly because of the class babycry.
Figure 4.9 illustrates the prediction results of experiment 4 in table 4.11. This recordings are the same ones in RNN prediction results 4.3. This visualizations only shows the similarity between the model predictions. However, there is a noise in the background of glassbreak event (the middle picture) which the LSTM also detected it as glassbreak.

Figure 4.10 illustrates the prediction results of experiment 4 in table 4.11 for the same mistake that RNNs did on detecting the sound of a rooster as babycry event shown in the figure 4.4. LSTMs also did detect the rooster as the babycry. However, the probability of its detection was lower than the RNNs and therefore the babycry is correctly detected.

Observations of confusion matrix in the experiment 4 in table 4.11 are shown in the table 4.12. Comparing this table with the RNNs confusion matrix on different offset tolerance in table 4.8 shows that the LSTMs result improvements was because of the lower False Positive rate and False negative rate for all events, especially the gunshot event which had the most impact on the lower error rate and higher f-score in comparison with these metrics achieved by RNNs.

Figure 4.11 illustrates the learning curve of the experiment 4 in table 4.11 for a 4 fold

LSTM results on mel-spectrogram					
Parameters		Exp-1	Exp-2	Exp-3	Exp-4
Augmentation		False	False	False	True
n_fft		2048	2048	1024	1024
overlap		50%	50%	50%	50%
segment duration		0.5 s	0.5 s	0.4 s	0.4 s
n_mels		128	128	128	128
n_layers		1	2	2	3
n_units		32	32	32	32
dropout		False	20%	30%	20%
loss-function		BCE	BCE	BCE	BCE
activation		sigmoid	sigmoid	sigmoid	sigmoid
optimizer		rmsprop	rmsprop	rmsprop	rmsprop
lr		0.0001	0.0001	0.0001	0.0001
batchsize		128	128	128	128
epochs		100	100	100	100
Babycry	err	0.65	0.30	0.28	0.27
	fscore	64.24%	72.27%	77.35%	77.84%
Glassbreak	err	0.46	0.45	0.45	0.34
	fscore	74.68%	78.54%	79.43%	81.05%
Gunshot	err	0.72	0.60	0.54	0.53
	fscore	63.37%	67.69%	68.52%	69.53%
Average	err	0.61	0.45	0.42	0.38
	fscore	65.43%	72.83%	75.10%	76.16%

Table 4.11:
LSTM results on DCASE 2017 Rare Sound Event Detection evaluation dataset, using mel-spectrograms as input representation.

cross validation. What can be observed in this image is that the gunshot and babycry events have relatively similar learning curv and that for gunshot, the model have relatively lower accuracy at the beginning but it learns the features of the event relatively fast within the first 10 epochs and afterwards it slowly goes higher, namely, higher accuracy and lower loss. The difference between validation accuracy and evaluation accuracy is not much which shows the similarity between the training batch and the evaluation batch in each fold.

Another interesting observation of learning curve for these experiments on LSTMs is that they all learned the glass break features relatively fast and the accuracy of the model prediction at the first epoch was much higher than the gunshot and baby cry events. Also notice that none of the learning process in each folds yield the complete 100 epochs due to the early stopping technique. This is because the validation loss did not change after 10 observations.

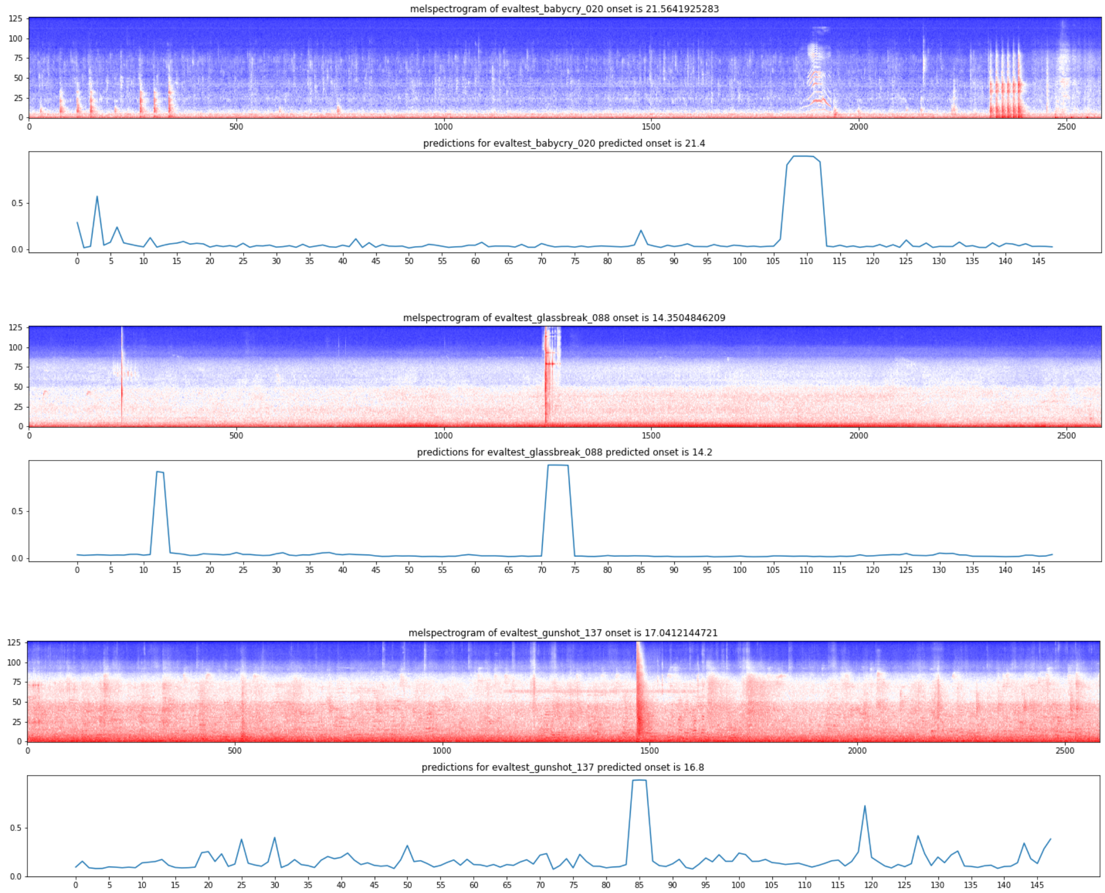


Figure 4.9: LSTM learning curve in experiment 4.

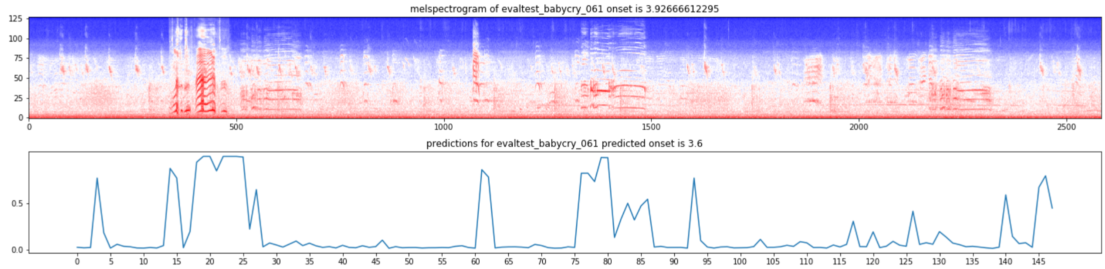


Figure 4.10:
Wrong onset prediction visualization for LSTM experiment 1. The model predicted the onset of the babycry correctly. However, still confused the sound of the cry with sound of a rooster.

To test the performance of LSTMs when using MFCCs and CQTs as input representation, more experiments were also performed and the results are written in the following

babycry	TN	TP	FN	FP	WO	Prcs	Rcl	ERR	F1
0.5 s	213	181	36	70	33	0.7211	0.8341	0.28	77.35%
0.75 s	213	188	36	63	26	0.7490	0.8392	0.25	79.15%
1 s	213	190	36	61	24	0.7569	0.8407	0.24	79.66%
glassbreak									
0.5 s	160	224	2	114	24	0.6627	0.9911	0.45	79.43%
0.75 s	160	225	2	113	23	0.6656	0.9911	0.45	79.64%
1 s	160	227	2	111	21	0.6715	0.9912	0.44	80.07%
gunshot									
0.5 s	170	172	22	136	56	0.5584	0.8865	0.54	68.53%
0.75 s	170	173	22	135	55	0.5615	0.8871	0.54	68.78%
1 s	170	174	22	134	54	0.5649	0.8887	0.53	69.04%

Table 4.12:
Testing the Onset detection results with different tolerances for the experiment 3.

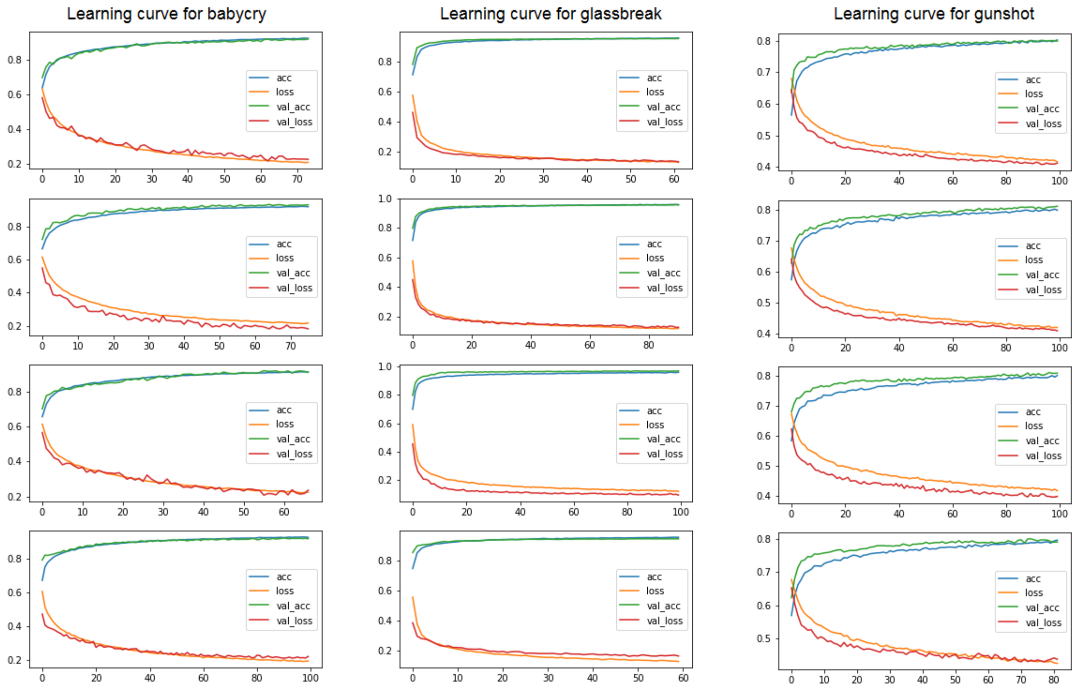


Figure 4.11: LSTM learning curve in experiment 4.

tables 4.13 and 4.14, respectively. To compare LSTMs with RNNs on MFCCs, the same parameters were used for both models and the feature extraction. Average results for experiment 1, using an RNN model was 0.58 error rate and 45.30% f-score which LSTMs outperformed RNNs by achieving 0.53 and 68.33% for error rate and f-score value. By analyzing the result of confusion matrix, it came to notice that RNN models had much difficulties in distinguishing background noise from the gunshot and glass break feature which caused very high false positive rate. LSTMs on the other side could detect these events more robust than RNNs and therefore had lower false positive rate. Also, LSTMs could often detect the events which lead to higher true positive rte than RNNs and therefore higher f-score. Still, compare to model's performance on mel-spectrograms as input representation, MFCC yield poorer results.

MFCC and LSTM Results for RSED			
Parameters		Exp-1	Exp-2
n_fft		2048	2048
overlap		50%	50%
MFCCs		40	128
layers		1	3
units		32	32
activation		sigmoid	sigmoid
dropout		20%	20%
loss_function		BCE	BCE
optimizer		rmsprop (.0001)	rmsprop (.0001)
batchsize		128	128
epoch		100	100
Babycry	err	0.39	0.47
	fscore	76.22%	72.83%
Glassbreak	err	0.74	0.67
	fscore	67.37%	70.19%
Gunshot	err	0.46	0.39
	fscore	61.40%	52.40%
Average	err	0.53	0.57
	fscore	68.33%	65.14%

Table 4.13: LSTM results on DCASE 2017 Rare Sound Event Detection Development Dataset, using MFCC as input representation.

CQT and LSTM Results for RSED			
Parameters		Exp-1	Exp-2
n_cqt		120	60
layers		1	1
units		32	32
activation		sigmoid	sigmoid
dropout		20%	20%
loss_function		BCE	BCE
optimizer		rmsprop (.0001)	rmsprop (.0001)
batchsize		128	128
epoch		100	100
Babycry	err	0.27	0.80
	fscore	73.26%	3.36%
Glassbreak	err	0.71	0.88
	fscore	29.48%	4.35%
Gunshot	err	0.65	0.70
	fscore	44.44%	33.07%
Average	err	0.54	0.39
	fscore	49.06%	52.40%

Table 4.14:
LSTM results on DCASE 2017 Rare Sound Event Detection Development Dataset, using CQT as input representation.

The visualizations of MFCC experiment 1 are depicted in Figure 4.12 which shows the difference between false and true positive detections for each class. In sub-figure (a) the audio mixture with park recording as background where no baby cry event was placed, the model detected the constant bird sounds throughout the audio as baby cry. However, in sub-figure (b) where the baby cry was placed on a office background noise, the event was correctly distinguished. Sub-figure (c) has street noise in the background where the model confused the sound of a bus with glass break. However, in sub-figure (d), where the background noise is in a room with air conditioner running ans people speaking, the glass break was detected correctly. Sub-figure (e) shows the MFCC representation of a beach recording as noise background where the sound of seagull was detected as gunshot. Sub-figure (f) has a quite room in the background where the gunshot sound is clearly detected by the model. In general, the model successfully detected the gunshot event, where no louder noise in the background was active. This problem was solved by using mel-spectrogram as input representation which lowered the false positive rate for all classes.

After investigating the learning process and results of the feed forward LSTM model, the work of [HWT$^+$16] gave the motivation to use the BLSTM model which had successfully learned the features of events using the information from previous and the feature time

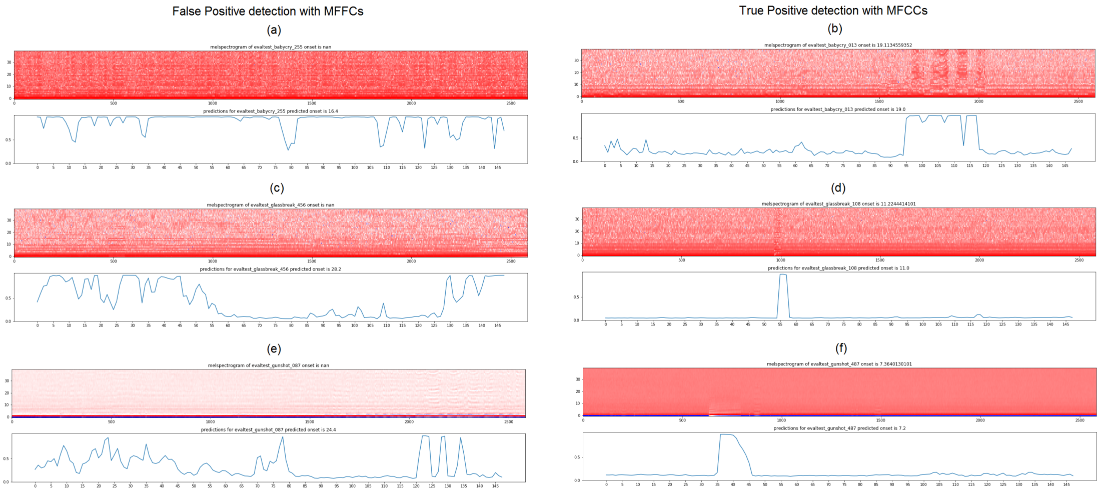


Figure 4.12: Comparison of false and true detection in an LSTM-MFCC based model.

steps. The results of the experiments are shown in the table 4.15.

Results of this model compare to the feed forward LSTMs and RNNs is very low and caused relatively large number of False Positives compare to LSTMs and RNNs. At first smaller model was used to train the model on non-augmented data. Then dropout technique was applied which avoided model's over fitting and showed improvement on model's performance on the evaluation dataset. The results are presented in table 4.15, experiment 1 and 2. Afterwards we augmented the data as explained before, both pitch shifting and time stretching manipulations on the signals which and achieved a dataset 2 times larger than the original dataset. By largening the model, the gunshot and glass break detection got improved but baby cry had very large number of false positive rate which caused a larger error rare. Overall, the learning rate achieved by the model in exp-4 was lower among the other experiments in the table 4.15 with error rate of 0.5 and f-score of 59.44% which in comparison with LSTMs is still low. Another set of BLSTM experiments is provided in table 4.16 to show the effect of using different optimization functions and learning rate for this model.
The parameters used for this table remained the same as the table 4.15 and only the optimization functions and learning rates are changed. The reason is because the remaining parameters showed to have the best of results among other sets of parameters which were tested. Experiment 5 in this table shows that by decreasing the learning rate of RMSProp from the default value 0.0001 to 0.00001 result of the BLSTM model improved from 0.5 error rate (and 59.44% f-core) to 0.48 error rate (and 61.74% f-score). This decrease of learning rate is due to observing high fluctuations in the learning curve for this experiment which showed that the learning rate is probably high and the step towards minimizing the cost function is large. Lowering the learning rate fixed those fluctuations and improved the results.

BLSTM results on mel-spectrogram					
Parameters		Exp-1	Exp-2	Exp-3	Exp-4
Augmentation		False	False	True	True
n_fft		2048	2048	2048	2048
overlap		50%	50%	50%	50%
segment duration		0.4 s	0.4 s	0.4 s	0.4 s
n_mels		128	128	128	128
n_layers		1	2	3	3
n_units		32	32	32	64
dropout		False	20%	30%	25%
loss-function		BCE	BCE	BCE	BCE
activation		sigmoid	sigmoid	sigmoid	sigmoid
optimizer		rmsprop	rmsprop	rmsprop	rmsprop
lr		0.0001	0.0001	0.0001	0.0001
batchsize		128	128	128	128
epochs		100	100	100	100
Babycry	err	0.51	0.55	0.56	0.58
	fscore	60.86%	59.32%	57.79%	56.24%
Glassbreak	err	0.38	0.39	0.39	0.29
	fscore	71.32%	69.20%	68.68%	78.43%
Gunshot	err	0.71	0.81	0.84	0.65
	fscore	40.14%	36.23%	34.32%	43.67%
Average	err	0.53	0.58	0.58	0.50
	fscore	57.44%	54.91%	53.53%	59.44%

Table 4.15:
BLSTM results on DCASE 2017 Rare Sound Event Detection evaluation dataset, using mel-spectrograms as input representation.

As it shows in the experiment 6, unlike the RNNs and feed forward LSTMs, Adam as optimization with 0.001 learning rate function improved the results for BLSTMs. This optimizer was also used for the experiments done by [HWT$^+$16] for BLSTM model on sound event detection. However, increasing the learning rate for this optimizer noticeably dropped down the precision and increased the error rate of the models prediction.
In the experiment 8, SGD optimizer with learning rate 0.0001 is used which did not outperformed the error rate using Adam optimizer for BLSTMs. Again, this was unlike the LSTMs and RNNs which SGD did not prove appropriate to use. Note that the model in experiment 8 has equal error rate with the one in experiment 6 but the model in experiment 6 has higher precision than the one in experiment 8 therefore higher f-score (62.34% compared to 60.44%). This is because of the lower number of False positives achieved in experiment 6.

BLSTM results on mel-spectrogram					
Parameters		Exp-5	Exp-6	Exp-7	Exp-8
Augmentation		True	True	True	True
n_fft		2048	2048	2048	2048
overlap		50%	50%	50%	50%
segment duration		0.4 s	0.4 s	0.4 s	0.4 s
n_mels		128	128	128	128
n_layers		1	1	1	1
n_units		32	32	32	32
dropout		20	20%	20%	20%
loss-function		BCE	BCE	BCE	BCE
activation		sigmoid	sigmoid	sigmoid	sigmoid
optimizer		rmsprop	Adam	Adam	SGD
lr		0.00001	0.0001	0.001	0.0001
batchsize		128	128	128	128
epochs		100	100	100	100
Babycry	err	0.42	0.40	0.56	0.32
	fscore	67.51%	69.43%	54.41%	73.51%
Glassbreak	err	0.34	0.33	0.45	0.39
	fscore	75.31%	76.27%	66.54%	70.76%
Gunshot	err	0.68	0.69	0.73	0.70
	fscore	42.40%	41.47%	39.14%	37.06%
Average	err	0.48	0.47	0.58	0.47
	fscore	61.74%	62.34%	53.37%	60.44%

Table 4.16:
BLSTM results on DCASE 2017 Rare Sound Event Detection evaluation dataset, using mel-spectrograms as input representation.

CNNs

The motivation behind using the recurrent neural network for the task of sound event detection is because of its ability to capture the temporal behavior of time series data. As this task is also similar to a computer vision task which is labeling images (spectrogram segments) with one or more classes, CNNs proved to excel at this and is widely used in image classification [SZ14, LSD15, OBLS14] and audio classification and recognition tasks [Pic15, PJAM15]. Therefore, this architecture is also tested and the results are compared with the recurrent neural networks.

As mentioned in the theoretical part of the thesis, CNNs are stacks of fully connected layers which each neurons in these layers are connected to all the pixels and performs dot products on an input weight matrix. The three main parameters which are set for CNNs in this work are the number of filters (which are the number of neurons), kernel-size in which determines a window which the network can "see" the image through it, similar to

the way that human eyes see an image. Max-pooling layer is used to down-sample the feature maps and only captures the maximum case of informations. By increasing the pooling filter size, the resolution is decreased even further and more information is lost [HZRS15, PHMM16].
The parameters chosen for these parameters are set by practice and performance observation. However, it is recommended to start from a small kernel size [AHMJ+14]. The first experiments using CNNs, a kernel with size 3*3 with 32 filters and a max-pooling layer with the size 2*2 were chosen, following 2 fully connected layers which achieved 0.32 average error rate and 75.04% f-score. As the size of the dataset was small (no augmentation was applied yet), enlarging the model by adding more layers to it, decreased the model's performance. In order to improve the model's precision, data augmentation was applied which 500 more samples of each event was created and using dropout method, 30% of the filters where randomly shutdown at each layer. Throughout the experiments, it came to notice that increasing the max-pooling filter size to 5*5 (and the same as the kernel size) improved the gunshot detection by reducing the error rate down to 0.35 and f-score to 63%. But reduced the average error rate of all the events to 0.41 error rate and 68.45% f-score which was caused by higher error rate for glass break and baby cry (0.38 and 0.54 respectively). Like the other model investigations in this work, only the experiments which had the best results among all the others are chosen and mentioned in the table 4.17.

Compared to the RNN model and its extensions, feed forward LSTMs and Bidirectional LSTMs, the CNNs performed better in the sound event detection task. However, the best performance which is shown in the experiment 4 is only slightly better than LSTMs. In experiment 1, no augmentation is applied and yet the lowest error rate for babycry and glassbreak events is achieved which yield over all the lowest error rate among all the other models and experiments. However, in terms of f-score (model's precision and recall), the experiment 4 had slightly better results than the one in first experiment (achieving 75.12% compared to 75.04% f-score in experiment 1). As for the optimization function, unlike RNNs and LSTMs, CNN performed much better on the inputs with SGD and Adam. Also, zooming into the spectrogram, lowered the accuracy of the segment classification. Therefore smaller values (1024 and 2048) for FFT size have been chosen. Another interesting observation here is the effect of data augmentation which for the CNN model with the same set of parameters resulted worse prediction on babycry but much better prediction for glassbreak. This is shown in experiments 2 and 3. The detection of gunshot changed slightly lower in terms of both f-score and error rate. By adding more convolutional layers from 2 to 3, the result got much closer to the results in experiment 2.The glass break was detected much better (having 0.25 error rate and 87.20% f-score). But since the babaycry and gunshot did not improve much (for babycry the deeper model achieved the error rate of 0.32 and f-score of 79.80% and for gunshot 0.42 and 66.15% error rate and f-score respectively), the average result of the detection was lower than the ones mentioned in the table 4.17.

CNN results on mel-spectrogram					
Parameters		Exp-1	Exp-2	Exp-3	Exp-4
Augmentation		False	False	True	True
n_fft		1024	2048	2048	1024
overlap		50%	50%	50%	50%
segment duration		0.4 s	0.4 s	0.4 s	0.4 s
n_mels		128	128	128	128
n_layers		1	2	2	1
n_filters		32	32	32	32
kernel_size		5x5	5x5	5x5	5x5
max_pooling		5x5	3x3	5x5	5x5
dropout		20%	30%	30%	30%
loss-function		BCE	BCE	BCE	BCE
activation		sigmoid	RELU	RELU	RELU
optimizer		sgd	sgd	sgd	adam
lr		0.001	0.001	0.001	0.0001
batchsize		128	128	128	128
epochs		100	100	100	100
Babycry	err	0.18	0.52	0.66	0.24
	fscore	80.70%	79.71%	71.67%	83.47%
Glassbreak	err	0.22	0.33	0.26	0.24
	fscore	82.28%	85%	79.43%	83.59%
Gunshot	err	0.57	0.42	0.38	0.44
	fscore	62.14%	67.22%	66.81%	57.34%
Average	err	0.32	0.42	0.43	0.30
	fscore	75.04%	76%	74.49%	75.03%

Table 4.17:
CNN results on DCASE 2017 Rare Sound Event Detection evaluation dataset, using mel-spectrograms as input representation.

Figure 4.13 illustrates the prediction results of experiment 4 in table 4.17. These recordings are the same as in RNN 4.3 and LSTM prediction results 4.9, except the babycry recording which was predicted as False Negative by CNNs. The noise in background of glassbreak recording (the middle picture) is also confused as glassbreak which is the same mistake in both RNNs and LSTMs (both feed forward and bidirectional).

Figure 4.14 illustrates the prediction results of experiment 4 in table 4.17 for the same mistake that RNNs and LSTMs did on detecting a bird sound as babycry event shown in the figure 4.4 and 4.10. CNNs also did detect the bird as the babycry. Also, even though that the babycry is detected here, it fell outside of the offset tolerance of the onset detection, namely, more than half a second further from the actual onset.

The same as other investigations, observations of confusion matrix in the experiment 4 in

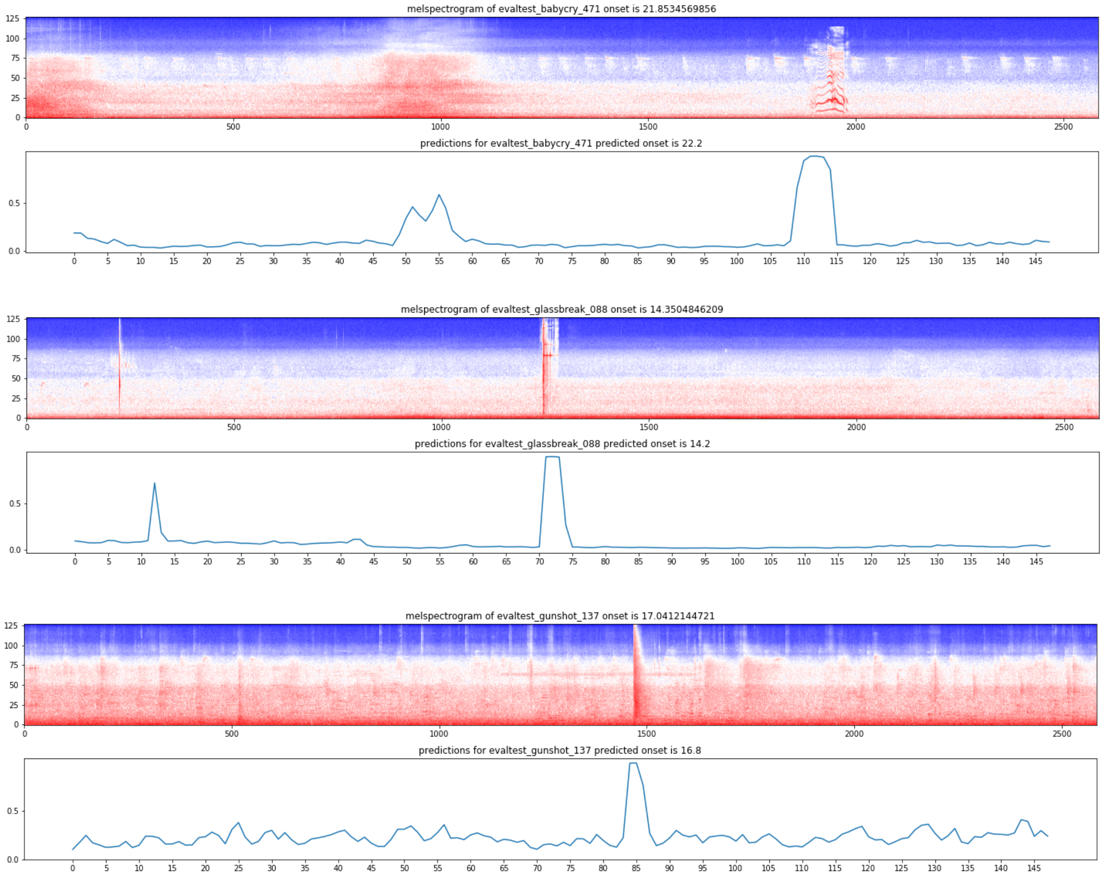


Figure 4.13: CNN learning curve in experiment 4.

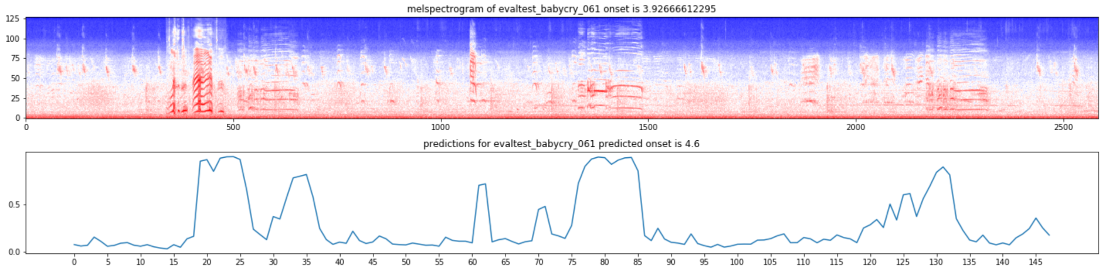


Figure 4.14:
Wrong onset prediction visualization for CNN experiment 4. The model, the same as RNN and LSTMs, confused the sound of baby cry with sound of a rooster.

table 4.17 are shown in the table 4.18. Comparing CNNs confusion matrix with RNNs 4.8 and LSTMs 4.12 shows that the CCNs have higher precision than the LSTMs and RNNs but also lower recall. This is because of the improvement in number of True Positives

and also increasing in number of False Negatives. As shown in this table, changing the tolerance affected only on the baby cry which reduced the shifted detected onsets from 37 to 30. By listening to those audio and visualizing their spectrograms it came to notice that the background noise in those audios where high when the event was occurring. For example a restaurant background noise, sound of cutleries caused the model not to be able to detect the start of cry.
However, changing the offset tolerance did not effect the glassbreak and gunshot event at all. for gunshot, the wrong detected onset was mostly because of the other events occurring in the recording which were more bold than the gunshot (such as smashing the door, tapping on the table, tapping on the pot while cooking). For glass break, there where only 2 recording which were detected as False Negative. By listening to the audios, it came to understanding that the glass break was very short (almost half a second) and the model did predict it, but it did not go above the determined threshold to recognize is as the glassbreak.

babycry	TN	TP	FN	FP	WO	Prcs	Rcl	ERR	F1
0.5 s	218	202	19	61	29	0.7680	0.9140	0.24	83.47%
0.75 s	218	208	19	55	23	0.7908	0.9162	0.22	84.89%
1 s	218	209	19	54	22	0.7946	0.9166	0.21	85.13%
glassbreak									
0.5 s	241	186	61	12	3	0.9393	0.7530	0.24	83.59%
0.75 s	241	186	61	12	3	0.9393	0.7530	0.24	83.59%
1 s	241	186	61	12	3	0.9393	0.7530	0.24	83.59%
gunshot									
0.5 s	187	132	73	108	46	0.55	0.6439	0.43	59.32%
0.75 s	187	132	73	108	46	0.55	0.6439	0.43	59.32%
1 s	187	132	73	108	45	0.55	0.6439	0.43	59.32%

Table 4.18:
Testing the Onset detection results with different tolerances for the experiment 4.

In figure 4.15, the learning cure of the experiment 4 in table 4.17 is illustrated. Like the other models, CNN results where also tested with 4 fold cross validation. The model learned event features relatively fast and the learning improvement slowly increased. Within the first epochs, the model rapidly increased its accuracy for all the events. But this pace for glassbreak was even higher. The learning curve for train and the evaluation sets are very similar which could show the great similarity between the sets.

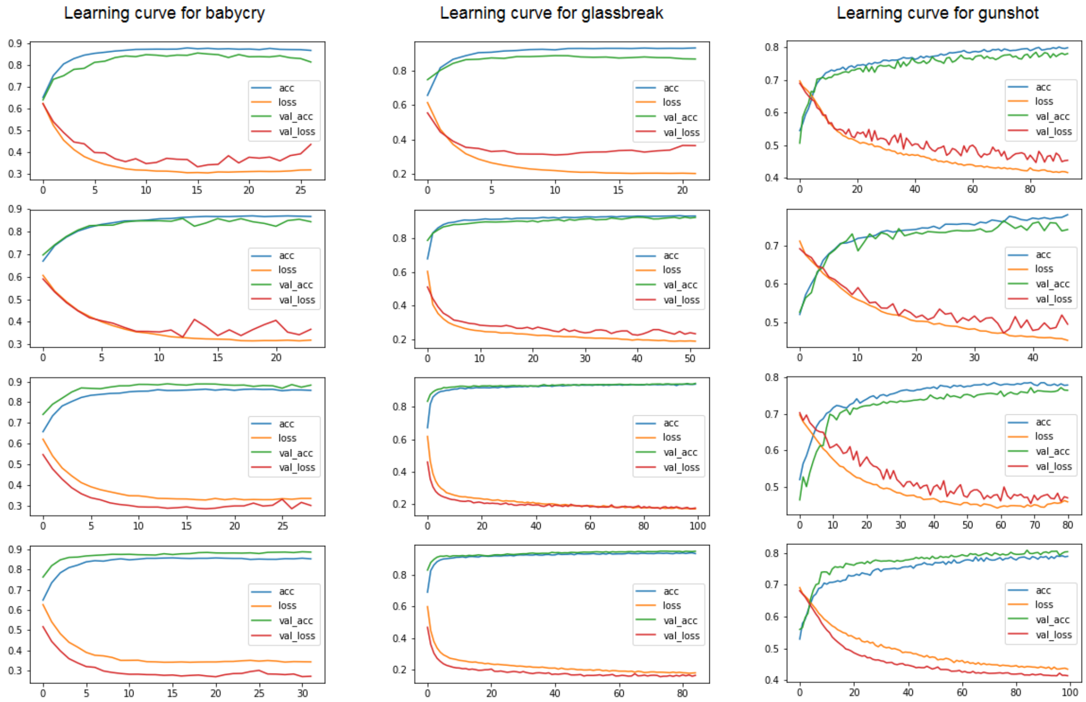


Figure 4.15: CNN learning curve in experiment 4.

4.4.3 Experimental Results on Sound Event Detection

As explained in the first chapter, SED is a task of detecting the onset and offset of multiple overlapping events. The audios provided by the DCASE dataset are 2-channel audios. The figure 4.16 illustrates polyphonic audio and its monophonic version which is achieved by averaging both channels into one.

To create an input dataset for the SED system, each audio signal is divided into smaller duration. Afterwards, as the CQT and MFCCs in the previous sub-section proved not to be suitable for this task, mel-spectrograms are used as input representation where the mel-bands are extracted from each of these short duration. Then each of these feature segments are fed into the neural network architecture which will map each of these segments into its corresponding audio event label. The same baseline system as for the RSED task was used for SED task provided by the DCASE2017 challenge [MHD$^+$17]. To Augment the data, the same as rare sound events experiments, we have tested different shift steps and stretch factors. Afterwards, the values which sounded different but still realistic to the audio was chosen and applied on the training set on each fold of the cross validation. Stretch factors were chosen from $\pm 0.5, \pm 0.7$ and Shift steps where chosen from $\pm 2, \pm 3$ to create an augmented dataset with different sizes (from 2 to 8 times larger than the original dataset).

In the following subsections, results of the experiments and their discussions are provided.

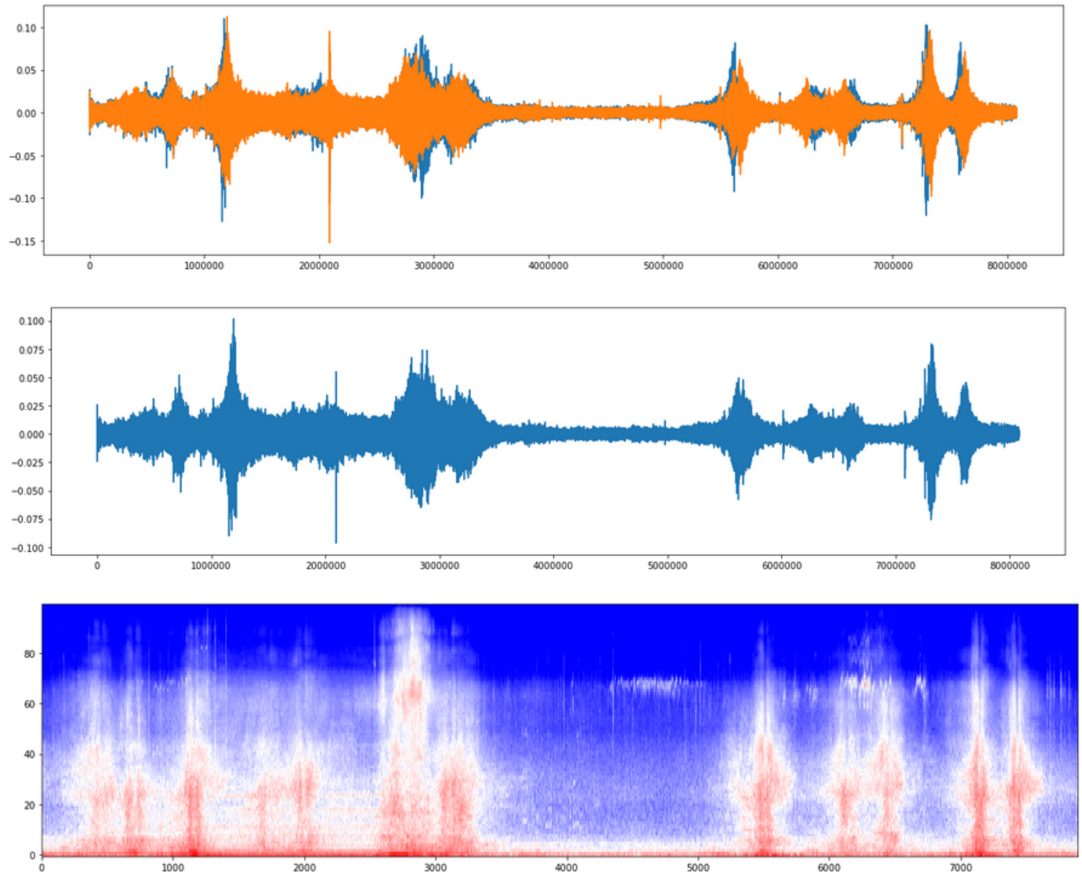


Figure 4.16: Illustration of a polyphonic and monophonic version of the an audio. The top image depicts a polyphonic audio with occurrence of multiple events such as car passing and people walking and speaking at the same time. The Image in the middle is the monophonic version of the same audio which was produced by calculating the mean of the both channels. The image bellow illustrates the extracted 100 mel-spectrograms of the mono channel Audio.

Pre-processing

In the figure 4.17, the segmentation process of an audio signal is depicted. Each signal is divided into smaller parts which then the frame wise multi-labels, based on the given annotation, are calculated as a binary string with 6 digits (each represents one event).

Figure 4.17 illustrates the segmentation process for SED system. Since the events are distributed all over the audio and there is no control on how many times they appear, each audio will be divided into a number of 40 ms segments with 20 ms overlap. Based on the experiments done in this thesis, this size have shown to be the most effective as an analysis window for the SED system. These segments are then used as the input. The

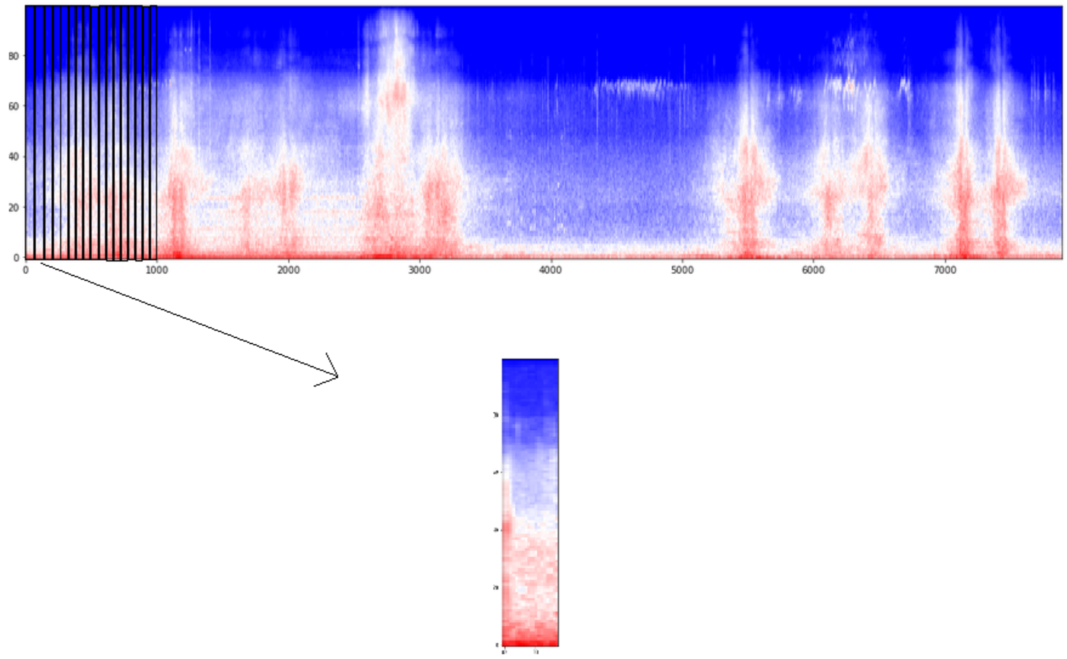


Figure 4.17: Segmentation procedure for SED task. Each audio is divided into smaller segments and these segments and their corresponding labels are then used as inputs for the SED system to be analyzed. The duration of each segment is 40 ms and they have a 20 ms overlap.

same as for the Rare Sound Event Detection, there are multiple experiments applied to study the behavior of recurrent neural network architecture and are compared with the state of the art baseline systems used for this work.

Model Evaluation

To Evaluate the behavior of deep learning architectures used as SED systems, two different systems are used as baseline. Multiple parameters was tested to study the performance of these models and table of results are provided in the following of this subsection. The evaluation metric used for comparison is the segment-based error rate. However, for further evaluation metrics such as accuracy and f-score are also provided. The calculation of segment-based error rate and f-score are presented in [MHV16a] where for each one second segment the f-score is calculated as shown in the equation 4.3 and error rate calculation is shown in the equation 4.4.

$$f1 = \frac{2 \cdot TPs}{2 \cdot TPs + FNs + FPs} \tag{4.3}$$

$$error_rate = \frac{S + D + I}{N} \qquad (4.4)$$

where SS is the sum of substitution errors ($min(FP, FN)$), I is the sum of insertion ($max(0, FPs - FNs)$), D is the sum of deletions ($max(0, FNs - FPs)$) and N is the number of active events in the segment.

Baseline Systems

As mentioned before, there are two different baseline systems. One system is provided by the DCASE challenge and the second is a state of the art machine learning algorithm, Support Vector Machine. The results of the deep Learning architectures is compared to both baseline systems.

DCASE 2017 Baseline system:
The baseline system for this task is also the same as the one used for RSED task, a 2 layered MLP system with 50 units in each layer and 20% dropout. The number of epochs was set to 200 (with early stopping which started the monitoring process at the epoch 100 with 10 observing epochs).The learning rate 0.001 and the activation function in the output layer, a sigmoid function. The experiment was trained and tested on each full audio. The average results are shown in the table 4.19 Note that the results of each class are the average of segment-wise f_score and error rate of one second long segments per each class and the average results which is then compared to the baseline system is the segment-wise average of all segments.

DCASE baseline results		
	Error-Rate	F-score
People Walking	1.44	33.5%
People Speaking	1.29	3.6%
Children	2.66	0.0%
Car	0.76	65.01%
Large Vehicle	1.44%	42.07%
Break	0.98%	4.1%
Average:	0.93	42.8 %

Table 4.19: DCASE 2017 Sound Event Detection baseline results: Segment-based overall metrics.

Support Vector Machine as baseline System:
A simple SVM was used as the second baseline system. The results of MFCC-based SVM, are shown in the table 4.20. We have trained the classifier separately for each class. Afterwards the segment based f-score and error rate for each class was calculated and the segment_wise average of error rate and f-score of taken as SVM-baseline results.
To set a SVM baseline, 2 different experiments were applied. As we observed from the previous subsection, mel-spectrogram as input representation achieved better results than the MFCCs. Therefore, here we also tested both MFCCs and Mel-spectrogram, applying a grid search on SVM with a given set of values for each hyper parameter. After comparing the results of both mel-spectrogram and MFCCs by applying the best parameter set for SVM model, we have observed that for this task, MFCCs have achieved better results than mel spectrogram. This is because of the high error rate which was caused due to large false positive rate which resulted 8% f-score and 1.61 error rate. As a consequence of having too little samples of Children, Large Vehicle and Brake, these labels had a large error rate which was due to large false negative and false positive rate. The detection of Car passing was less challenging (still very low precision) in comparison to the other labels.
Following of this section contains subsection for each model and explains each deep learning architectures performance with different set of parameters. As the RNN model results were poor in previous dataset and also the first tries on this dataset (achieving segment base error rate with augmentation 0.98 and f_score below 22%), we continue the experiments by focusing on LSTMs and its extension Bi-LSTMs and CNNs. The result of RNN experiment is shown in the overview table 4.3

SVM baseline results		
	Error-Rate	F-score
People Walking	1.40	13.33%
People Speaking	1.31	4.5%
Children	2.4	0.95%
Car	0.98	19.54%
Large Vehicle	1.74%	7.01%
Break	1.86%	1.31%
Average	1.28	8.12%

Table 4.20: Sound Event Detection SVM-MFFC based baseline results: Segment-based overall metrics.

LSTMs

Following the work of Adavanne et al. [APP+16] on DCASE 2016 Sound Event detection and Classification, a two layer LSTM with 32 hidden unit was used on the real life sound event detection dataset. Table 4.21 shows the result of multiple experiments motivated by their work. The first experiment uses the same signal processing parameters for audio segmentation which are 2048 FFT window size on 40 ms audio segments with

50% overlap. Afterwards, 40 mel-spectrograms where calculated from each of the audio segments which were then used as the LSTM input.
The same as previous dataset, we also trained the model using a 4 fold cross validation. Variance of classification accuracy among all the folds is very low which means the training folds where not diverse.

LSTM results on mel-spectrogram					
Parameters		Exp-1	Exp-2	Exp-3	Exp-4
Augmentation		False	False	True	True
n_fft		2048	2048	2048	2048
overlap		50%	50%	50%	50%
segment duration		0.4 s	0.4 s	0.4 s	0.4 s
n_mels		40	80	80	80
n_layers		2	2	3	3
n_units		32	32	64	64
dropout		20%	20%	25%	25%
loss-function		BCE	CCE	CCE	CCE
activation		sigmoid	sigmoid	sigmoid	sigmoid
optimizer		RMSprop	RMSprop	RMSprop	RMSprop
lr		0.0001	0.0001	0.0001	0.0001
batchsize		64	64	128	128
epochs		200	200	200	200
People Walking	err	0.92	0.92	0.90	0.89
	fscore	8.34%	8.21%	10.13%	13.53%
People Speaking	err	0.99	0.98	0.96	0.92
	fscore	0.44%	1.37%	4.35%	12.21%
Children	err	1	1	1	1
	fscore	0.0%	0.0%	0.0%	0.0%
Car	err	0.8	0.76	0.7	0.63
	fscore	22.33%	27.09%	33.14%	42.35%
Large Vehicle	err	0.81	0.76	0.82	0.87
	fscore	20.08%	26.29%	20.61%	14.66%
Brake	err	1	1	1	1
	fscore	0.0%	0.0%	0.0%	0.0%
Seg-Wise Average	err	0.82	0.83	0.81	0.79
	fscore	27.40%	30.11%	33.21%	41.02%

Table 4.21:
LSTM results on DCASE 2017 Real Life Sound Event Detection evaluation dataset, using mel-spectrograms as input representation.

As shown in the table, in all the experiments, squeaking sound of the car brake, people speaking and children where never detected. For each experiment, different set of

parameters for input representation is tested to see if we can improve the results. As it was mentioned in multiple works [TGPVG16] [PHH$^+$17], choosing a higher value for mel-spectrogram extract high frequency components from the audio [HWT$^+$16]. Therefore the 80 mel bands were extracted from the audio signals which resulted a higher precision but also slightly increased the segment based error rate in comparison with experiment 1. This is due to higher precision on detecting the car and large vehicle sound events. Even though the error rate is lower than the baseline system, the model has lower true positives (correct detection) and therefore is not as precise. In order to increase the True Positive rate, we have applied data augmentation which was mentioned before and increased the size of the dataset 4 times more and carefully used thresholding to reduce the false positive by setting the threshold from 0.5 to 0.7. The larger value for resulted in increasing the number of false negative and therefore 0.7 proved more appropriate. Experiments 3 and 4 in the table 4.21 provide the highest results of the model with data augmentation. The combination of both shifted-pitch and stretched audios using a deeper LSTM model resulted in increasing the f-score (by improving the number of correct detections). In experiment 3, size of the dataset is 4 times bigger which is by augmenting the segments using pitch shifting ($\pm 2 steps$) and time stretching ($\pm 0.5 factors$). For experiment 4, we increased the dataset 8 times more also using both pitch shifting ($\pm 2, \pm 3$) and time stretching ($\pm 0.5, \pm 0.7$). As shown in the table of results, the augmentation improved the detection of classes "People Walking", "People Speaking" and "Car" noticeably but decreased the prediction of "Large Vehicle" where the model failed to predict any active event in the segment and resulted high False Negative rate for large vehicle event.
We further tried experiments by zooming in and zooming out the segments (choosing 4098 and 1024 FFT size respectively). However, the FFT size 2048 proved better than 1024 and 4096 in terms of detection with size 1024 False negative rate and with 4096 False positive rate increased. Additionally, we applied CQTs and MFCCs as input representation which MFCC resulted in very high error rate (1.45) and very low (12.08%) proved to be not suitable. CQT also achieved lower results compared to LSTMs but still higher than MFCCs (26.41% f-score and 1.26 error rate). Different input representation also did not help the model in detecting the children and brake squeaking sound events.

Following the work of [HWT$^+$16], we further tested Bidirectional-LSTMs on this dataset. The results of this experiments are provided in table 4.22. The BLSTM model performed better with 40 mel bands extracted features and FFT size 4096 (2 times more than the window size used for LSTMs) which shows that zooming in the spectrogram and concatenating informations from both direction helped the model to be able to make more correct predictions. The f-score result is almost the same as LSTM but with slightly lower error rate (0.2 decreasing in error rate). The BLSTM detected passing cars better than the LSTMs but performed poorly on detecting the large vehicle. By increasing the number of extracted mel-bands, the model has improved in detecting large vehicles but precision of car sound event noticeably decreased.

We have compared the performance of LSTM with BLSTM, by observing the best

BLSTM results on mel-spectrogram					
Parameters		Exp-1	Exp-2	Exp-3	Exp-4
Augmentation		False	False	True	True
n_fft		4096	4096	4096	4096
overlap		50%	50%	50%	50%
segment duration		0.4 s	0.4 s	0.4 s	0.4 s
n_mels		40	80	40	40
n_layers		2	2	3	3
n_units		32	32	64	128
dropout		20%	20%	25%	30%
loss-function		BCE	CCE	CCE	CCE
activation		sigmoid	sigmoid	sigmoid	sigmoid
optimizer		RMSprop	RMSprop	RMSprop	RMSprop
lr		0.0001	0.0001	0.0001	0.0001
batchsize		64	64	128	128
epochs		200	200	200	200
People Walking	err	0.95	0.93	0.9	0.91
	fscore	5.28%	6.99%	11.94%	15.94%
People Speaking	err	0.97	0.81	0.95	0.95
	fscore	2.60%	2.86%	5.06%	6%
Children	err	1	1	1	0.99
	fscore	0.0%	0.0%	0.0%	1.11%
Car	err	0.70	0.72	0.72	0.7
	fscore	34.39%	26.96%	31.50%	34.35%
Large Vehicle	err	0.86	0.71	0.91	0.91
	fscore	14.45%	16.35%	9.54%	9.71%
Brake	err	1	1	1	0.97
	fscore	0.0%	0.0%	0.0%	3.28%
Seg-Wise Average	err	0.80	0.89	0.90	0.93
	fscore	31.00%	24.63%	27.88%	31.72%

Table 4.22:
BLSTM results on DCASE 2017 Real Life Sound Event Detection evaluation dataset, using mel-spectrograms as input representation.

prediction results of these models (for LSTM, experiment 2 in table 4.21 and for BLSTM, experiment 2 in table 4.22). As also provided in the table of results, the LSTM model performed better than the BLSTM model. One reason could be the information of both direction which confused the BLSTM model and reduced its precision in event detection. However, unlike LSTMs, BLSTM was able to detect the children and brake squeaking sound events (referring to the Exp-4 in the table of results) but had a poor prediction on large vehicle where LSTMs performed better. Compared to BLSTMs, prediction results in LSTM had higher True positives (also False positive). This observation is illustrated in figure 4.18 which is the prediction results of one of the evaluation set audios. As also depicted in the figure, both models failed to detect the children sound and the break squeaking.

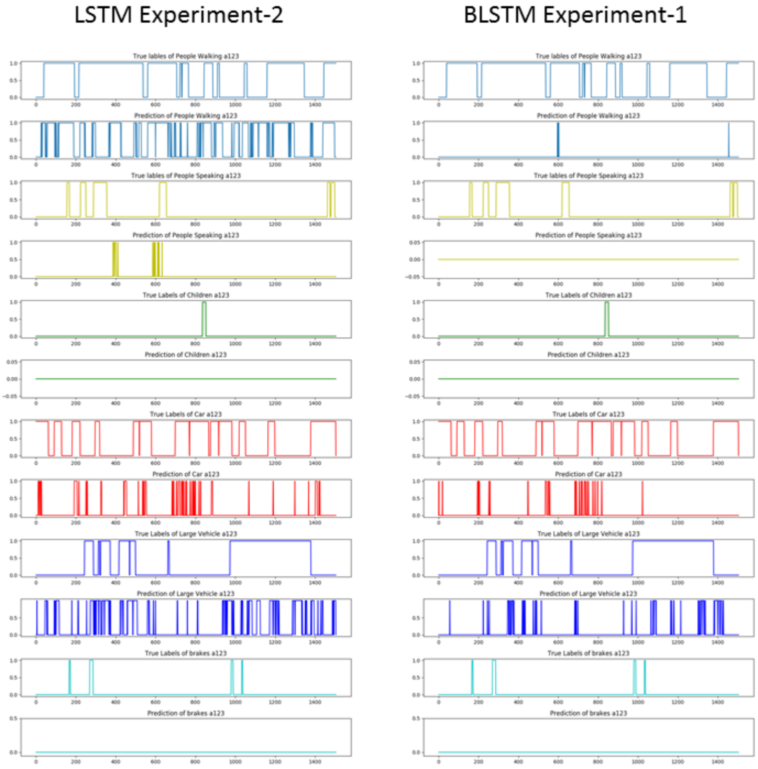


Figure 4.18: Comparison of LSTM (Experiment 2 in 4.21) prediction results with BLSTM (Experiment 1 in 4.22)

CNNs

In this subsection, we followed the work of [GMS16b], we used 1 Convolutional layer consists of 80 filters with 3*60 kernel size followed by a 4x3 max-pooling layer and 2 fully connected layers. We trained the network using Adam (0.001 learning rate) optimizing cross-entropy loss function. We stop the training using the early stopping technique by 10 epoch patience on validation loss. Like the other experiments, 20% dropout is also applied on each layer and the model is tested on 4 fold cross validation. Afterwards, to observe the changes in results and improving the prediction scores, we applied more experiments which 4 of the top results are provided in table 4.23. Experiment 1 refers to the first experiment which was motivated by [GMS16b]. Comparing to BLSTM and LSTMs, CNN had lower f-score and higher error rate which shows that the LSTMs (and B-LSTMs), this model is less suited for detecting the overlapping sounds and capturing the temporal behavior of the events. We have also tried smaller kernel sizes (as it was recommended by [AHMJ^{+}14]) which reduced the error rate and increased the f-score. However, the score is still less than the ones provided by the baseline which is due to low number of correctly detected events (True Positive rate). We further more applied data augmentation which the experiments 3 and 4 in the table 4.23 refer to two of the highest results achieved by this model. We have compared the prediction results of CNNs with LSTMs which is illustrated in figure 4.19 where we can see the lower number of false predictions by CNN in comparison with LSTMs but also lower number of correctly predicted segments which resulted in lower f-score for the CNN model. CNN model in exp-3 was able to predict children and brake sqeaking as well as the BLSTM in exp-4 4.22 which shows that augmenting the data helped these two models on discovering these difficult events.

Throughout these experiments we noted the difficulty in detecting the street sound events in comparison with the synthetic sound events where the presence of each classes using the extracted mel-bands where more clear to the model. As it is also stated in [PKK^{+}17], audio event detection is a more complex task than classification and needs further investigation in order to achieve accurate models. Moreover, complex signal processing techniques are required to separate the source of each sound in order to capture their characteristic and properly identify the presence of events in the audio signals.

CNN results on mel-spectrogram				
Parameters	Exp-1	Exp-2	Exp-3	Exp-4
Augmentation	False	False	True	True
n_fft	2048	1024	1024	1024
overlap	50%	50%	50%	50%
segment duration	0.4 s	0.4 s	0.4 s	0.4 s
n_mels	60	40	80	128
n_layers	2	2	2	2
n_filters	80	64	64	64
kernel_size	60	5	5	7
max_pooling	4	3	3	5
dropout	20%	20%	30%	25%
loss-function	BCE	CCE	CCE	CCE
activation	sigmoid	sigmoid	sigmoid	sigmoid
optimizer	SGD	Adam	SGD	SGD
lr	0.001	0.001	0.001	0.001
batchsize	64	64	128	128
epochs	200	200	200	200

People Walking	err	0.95	0.96	0.94	0.92
	fscore	5.33%	4.62%	5.89%	9.29%
People Speaking	err	1	1	0.98	0.97
	fscore	0.0%	0.0%	3.65%	2.71%
Children	err	1	1	0.97	0.98
	fscore	0.0%	0.0%	3.14%	2.88%
Car	err	0.88	0.80	0.8	0.74
	fscore	14.03%	22.02%	23.48%	28.82%
Large Vehicle	err	0.84	0.86	0.87	0.84
	fscore	16.38%	14.25%	13.82%	16.38%
Brake	err	1	1	0.80	1
	fscore	0.0%	0.0%	23.69%	0.0%
Seg-Wise Average	err	0.93	0.84	0.89	0.77
	fscore	18.61%	21.11%	23.63%	28.61%

Table 4.23:
CNN results on DCASE 2017 Real Life Sound Event Detection evaluation dataset, using mel-spectrograms as input representation.

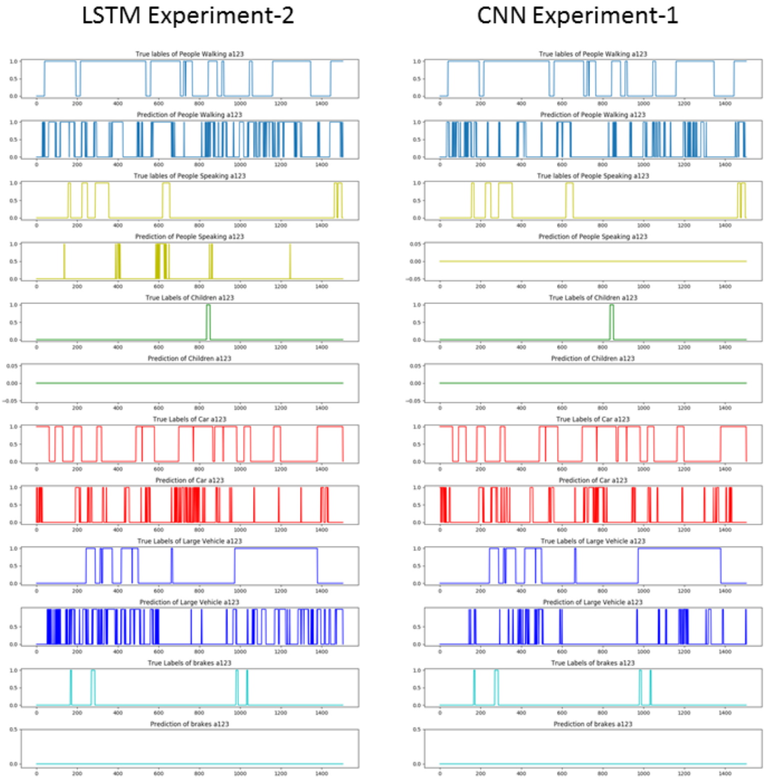


Figure 4.19: Comparison of LSTM (Experiment 2 in 4.21) prediction results with CNN (Experiment 1 in 4.23)

CHAPTER 5

Conclusion and Future Work

In this chapter, a conclusion is made over the entire project and steps which have been taken to achieve these goals, and the outcomes, in Section 5.1. Furthermore, Section 5.2 sketches some possible ways to continue and expand this project toward further optimizations and improvements of the solution to the problem which was stated in Section 1.1.

5.1 Conclusion

In this thesis we have systematically investigated the characteristics of two most popular architectures of deep neural networks, Convolutional Neural Networks and Recurrent Neural Networks (with its extensions, Bidirectional network and long Short term Memories). To answer the first scientific question of this thesis, we have studied the performance of deep models by applying different techniques such as MFCCs, Log-Amplitude Mel-spectrograms and CQTs which Log-Amplitude Mel-spectrograms proved to be more appropriate for these models.

To answer the second scientific question of the thesis contribution, we have utilized techniques such as dropout which avoids model's over fitting by randomly blocking informations from passing to the next layer and cross validation for reproducibility. To achieve model's convergence and generalizability, we have applied data augmentation which was done by manipulating the audio signals pitch and stretching the time axis.

Furthermore, we answered the final scientific question, by investigating and optimizing deep models such as LSTMs, BLSTMs and CNNs in which LSTMs and CNNs outperformed both baseline systems (MLP and SVM) and other models (RNNs and BLSTMs) and showed their robustness in learning the characteristics of different events. We have Also noticed that augmenting the training set noticeably reduces the False positive rate which leads to higher model precision (70.02%) and therefore higher F-score (76.24%).

We have evaluated the models on two different datasets, synthetic rare sound event detection (RSED) and real life sound event detection (SED) where the model performance on synthetic RSED dataset exceeded the performance on the real life SED dataset. This can be due to reasons such as the accuracy in defining the onset and offset of the events. For synthetic RSED, we observe that the model had difficulties to distinguish gunshot from louder background noise (e.g. dropping objects on the floor, smashing the door) and therefore had lowest precision among the other two events (57.17%). The similarity between features of some singing birds and baby cry also resulted in falsely detecting a cry event.

In the second dataset, real life SED, the model had difficulties to detect the brakes squeaking, children (which sometimes was just a scream and sometimes talking and laughing), and people speaking. On the other hand, car, large vehicle and people walking had higher chances of detection. However they were often below 50% f_score which shows the low precision of the model on detecting these street sounds and need for an approach to clarify the presence of these events in the audio.

5.2 Future Work

There number of works which can be further studied towards advanced optimization and improvements of the sound event detection task are investigating the hybrid models such as CRNN which have shown very good performance, GRUS which was introduced later than LSTMs but achieved the same performance with less computational complexity [GSR+17]. Another study which is considered for improvement on this project is polyphonic SED where both channels of audio signal (if it is a two channel audio) is used separately to train the model and then the results are concatenated for the prediction. This approach mimics the performance of our ear where the audio is heard from two different inputs and was examined in the work of Adavanne [AV17] which achieved the first place in DCASE challenge 2017. Also, applying an attention layer on top of neural networks, increases the detection accuracy which was successfully tested in the work of Xu [XKH+17]. To better understand the behavior of deep neural network and be able to justify the predictions which they are making, investigation through its deep layers using methods such as visualization and mathematical models have been investigated [SWM17] which is considered to be a further study of this project as well.

List of Figures

1.1 An overview of a sound event detection 2

2.1 Monophonic SED vs Polyphonic SED 10
2.2 A chart of published work on SED using deep learning 10
2.3 Effects of EMDA and VTLP (data augmentation) methods with enlarging number of augmented data. 11
2.4 Flowchart of the sound event detection system 12

3.1 Sub-figure(a) illustrates the waveform and sub figure (b) is the calculated spectrogram of waveform, representing a baby cry 16
3.2 Sub-figure(a) illustrates the waveform and subfigure (b) illustrates the spectrogram of the waveform, representing a glass break 16
3.3 Sub-figure(a) illustrates the waveform and subfigure (b) illustrates the spectrogram of the waveform, representing a gunshot 16
3.4 Illustration of multiple FFT window sizes 18
3.5 MFCC block diagram 19
3.6 Comparison of different mel-bands in MFCCs 21
3.7 Comparison of different mel bands in Mel spectrograms 22
3.8 Comparison of STFT and CQT 23
3.9 Data Augmentation: Compressing the Audio Signal. 25
3.10 Data Augmentation: Expanding the Audio Signal. 26
3.11 Comparison of the pitch shifted audio with an original audio wave: step = 2 27
3.12 Comparison of the pitch shifted spectrogram with an original spectrogram: step = 2 27
3.13 Comparison of the pitch shifted waveform with an original waveform: step = -2 28
3.14 Comparison of the pitch shifted spectrogram with an original spectrogram: step = -2 28
3.15 A recurrent neural network in its forward computation process 30
3.16 Architecture of BRNN 31
3.17 Forget layer in the LSTM Architecture 33
3.18 Input layer in the LSTM Architecture 34
3.19 The repeating module in an LSTM contains four interacting layers 34

3.20 Example of a convolutional layer 36
3.21 An overview of a convolutional neural network model 36
3.22 Overlap between two windows 38
3.23 Sigmoid activation function 39
3.24 Dropout method applied in the layers of a fully connected neural network 40

4.1 Segmentation process for rare Sound Event Detection 51
4.2 Probability distribution through an audio signal 52
4.3 The prediction visualization: RNN model for Rare Sound Event Detection 58
4.4 The wrong onset prediction visualization: RNN for rare Sound Event Detection 58
4.5 Visualization of the RNN- 4 fold cross validation on rare Sound Event Detection 60
4.6 Visualization of the RNN- 4 fold cross validation on Rare Sound Event Detection (2) 61
4.7 Illustration of the learning curves for 4 fold cross validation (1) 64
4.8 Illustration of the learning curves for 4 fold cross validation (2) 65
4.9 LSTM learning curve in experiment 4. 67
4.10 Wrong onset prediction visualization for LSTM 67
4.11 LSTM learning curve in experiment 4. 68
4.12 LSTM-MFCC based: comparison of FP and TP detections 71
4.13 CNN learning curve in experiment 4. 76
4.14 Wrong onset prediction visualization for CNN 76
4.15 CNN learning curve in experiment 4. 78
4.16 Illustration of a polyphonic and monophonic version of the an audio . . . 79
4.17 Segmentation procedure for SED task 80
4.18 Comparison of LSTM and BLSTM outputs 86
4.19 Comparison of LSTM and CNN outputs 89

List of Tables

2.1 A tabular summarization of the similar works on sound event detection. 14

4.1 Hardware specifications 47
4.2 Rare Sound Event Detection Experiment Overview 49
4.3 Real Life Street Sound Event Detection Experiment Overview 50
4.4 DCASE 2017 Rare Sound Event Detection baseline results 54
4.5 Rare Sound Event Detection SVM baseline results 55
4.6 RNN-mel based results on DCASE 2017 Rare Sound Event Detection: . . 56
4.7 Onset detection results with different tolerance: RNN-1 59
4.8 Onset detection results with different tolerance: RNN-3 60
4.9 RNN-MFFC based results on DCASE 2017 Rare Sound Event Detection . 63
4.10 RNN-CQT based results on DCASE 2017 Rare Sound Event Detection . 64
4.11 LSTM results on DCASE 2017 Rare Sound Event Detection 66
4.12 Onset detection results with different tolerance: LSTMs-3 68
4.13 LSTM-MFCC based results on DCASE 2017 Rare Sound Event Detection 69
4.14 LSTM-CQT based results on DCASE 2017 Rare Sound Event Detection . 70
4.15 BLSTM results on DCASE 2017 Rare Sound Event Detection 72
4.16 BLSTM results on DCASE 2017 Rare Sound Event Detection evaluation (2) 73
4.17 CNN results on DCASE 2017 Rare Sound Event Detection 75
4.18 Onset detection results with different tolerance: CNN-4 77
4.19 DCASE 2017 Sound Event Detection baseline results 81
4.20 Sound Event Detection SVM-MFFC based baseline results: Segment-based overall metrics. 82
4.21 LSTM results on DCASE 2017 Real Life Sound Event Detection 83
4.22 BLSTM results on DCASE 2017 Real Life Sound Event Detection 85
4.23 CNN results on DCASE 2017 Real Life Sound Event Detection 88

Bibliography

[AHMJ+14] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[All77] Jonathan Allen. Short term spectral analysis, synthesis, and modification by discrete fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(3):235–238, 1977.

[APP+16] Sharath Adavanne, Giambattista Parascandolo, Pasi Pertilä, Toni Heittola, and Tuomas Virtanen. Sound event detection in multichannel audio using spatial and harmonic features. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016.

[AV17] Sharath Adavanne and Tuomas Virtanen. Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017.

[BBB+10] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7, 2010.

[BP92] Judith C Brown and Miller S Puckette. An efficient algorithm for the calculation of a constant q transform. *The Journal of the Acoustical Society of America*, 92(5):2698–2701, 1992.

[BPT+09] Miguel Bugalho, José Portelo, Isabel Trancoso, Thomas Pellegrini, and Alberto Abad. Detecting audio events for semantic video search. In *Tenth Annual Conference of the International Speech Communication Association*, pages 1151–1154, 2009.

[CDY10] Yashpalsing Chavhan, ML Dhore, and Pallavi Yesaware. Speech emotion recognition using support vector machine. *International Journal of Computer Applications*, 1(20):6–9, 2010.

[CGCB15] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

[CGK15] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(9):1469–1477, 2015.

[CGO06] Lei Chen, Sule Gunduz, and M Tamer Ozsu. Mixed type audio classification with support vector machine. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 781–784. IEEE, 2006.

[CHHV15] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Polyphonic sound event detection using multi label deep neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015.

[CKBK16] Inkyu Choi, Kisoo Kwon, Soo Hyun Bae, and Nam Soo Kim. Dnn-based sound event detection with exemplar-based approach for noise reduction. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pages 16–19, 2016.

[CNK09] Selina Chu, Shrikanth Narayanan, and C-C Jay Kuo. Environmental sound recognition with time–frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1142–1158, 2009.

[CNKM06] Selina Chu, Shrikanth Narayanan, C-C Jay Kuo, and Maja J Mataric. Where am i? scene recognition for mobile robots using audio features. In *IEEE International Conference on Multimedia and Expo*, pages 885–888. IEEE, 2006.

[CV17] Emre Cakir and Tuomas Virtanen. Convolutional recurrent neural networks for rare sound event detection. September 2017.

[CVMBB14] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[Den12] Li Deng. Three classes of deep learning architectures and their applications: a tutorial survey. *Proceedings of Asia Pacific Signal and Information Processing Association (APSIPA) transactions on signal and information*, 2012.

[DHV13] Aleksandr Diment, Toni Heittola, and Tuomas Virtanen. Sound event detection for office live and office synthetic aasp challenge. *Proceedings of the IEEE Audio and Acoustic Signal Processing (AASP) Challenge on Detection and Classification of Acoust Scenes Events (WASPAA)*, 2013.

[dOVG$^+$15] Allan G de Oliveira, Thiago M Ventura, Todor D Ganchev, Josiel M de Figueiredo, Olaf Jahn, Marinez I Marques, and Karl-L Schuchmann. Bird acoustic activity detection based on morphological filtering of the spectrogram. *Applied Acoustics*, 98:34–42, 2015.

[DTC13] Jonathan Dennis, Huy Dat Tran, and Eng Siong Chng. Overlapping sound event recognition using local spectrogram features and the generalised hough transform. *Pattern Recognition Letters*, 34(9):1085–1093, 2013.

[GAFC$^+$16] JM Gutierrez-Arriola, R Fraile, A Camacho, T Durand, JL Jarrin, and SR Mendoza. Synthetic sound event detection based on mfcc. *Integration*, 1000(1):30–34, 2016.

[GMS16a] Arseniy Gorin, Nurtas Makhazhanov, and Nickolay Shmyrev. DCASE 2016 sound event detection system based on convolutional neural network. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016.

[GMS16b] Arseniy Gorin, Nurtas Makhazhanov, and Nickolay Shmyrev. Dcase 2016 sound event detection system based on convolutional neural network. *IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events*, 2016.

[GPMK16] Panagiotis Giannoulis, Gerasimos Potamianos, Petros Maragos, and Athanasios Katsamanis. Improved dictionary selection and detection schemes in sparse-cnmf-based overlapping acoustic event detection. *IEEE Audio and Acoustic Signal Processing (AASP) Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.

[GSR$^+$17] Shabnam Ghaffarzadegan, Asif Salekin, Anirudh Ravichandran, Samarjit Das, and Zhe Feng. Detection and classification of acoustic scenes and events 2017 bosch rare sound events detection systems for dcase2017 challenge. 2017.

[HMEV13] Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen. Context-dependent sound event detection. *The European Association for Signal Processing (EURASIP) Journal on Audio, Speech, and Music*, pages 1–13, 2013.

[HMS05] Aki Harma, Martin F McKinney, and Janto Skowronek. Automatic surveillance of the acoustic activity in our living environment. In *IEEE International Conference on Multimedia and Expo, 2005. ICME.*, pages 4–pp. IEEE, 2005.

[HMVG13] Toni Heittola, Annamaria Mesaros, Tuomas Virtanen, and Moncef Gabbouj. Supervised model training for overlapping sound events based on unsupervised source separation. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8677–8681, 2013.

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[HWT+16] Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, Takaaki Hori, Jonathan Le Roux, and Kazuya Takeda. Bidirectional lstm-hmm hybrid system for polyphonic sound event detection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pages 35–39, 2016.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[JLHL17] Il-Young Jeong, Subin Lee, Yoonchang Han, and Kyogu Lee. Audio event detection using multiple-input convolutional neural network. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017.

[KB15] Diederik P Kingma and Lei Ba. J. adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[KLB17] Wang Kaiwu, Yang Liping, and Yang Bin. Audio events detection and classification using extended R-FCN approach. Technical report, Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017), September 2017.

[KŁC11] Józef Kotus, Kuba Łopatka, and Andrzej Cżyzewski. Detection and localization of selected acoustic events in 3d acoustic field for smart surveillance applications. *Multimedia Communications, Services and Security*, pages 55–63, 2011.

[KSWP16a] Qiuqiang Kong, Iwnoa Sobieraj, Wenwu Wang, and Mark D Plumbley. Deep neural network baseline for dcase challenge 2016. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pages 50–54, 2016.

[KSWP16b] Qiuqiang Kong, Iwona Sobieraj, Wenwu Wang, and Mark Plumbley. Deep neural network baseline for DCASE challenge 2016. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016.

[LD99] Jean Laroche and Mark Dolson. New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects. In *1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 91–94. IEEE, 1999.

[LD17a] Rui Lu and Zhiyao Duan. Bidirectional gru for sound event detection. *Detection and Classification of Acoustic Scenes and Events*, 2017.

[LD17b] Rui Lu and Zhiyao Duan. Detection and classification of acoustic scenes and events 2017 bidirectional gru for sound event etection. 2017.

[LL17] Yanxiong Li and Xianku Li. The SEIE-SCUT systems for IEEE AASP challenge on DCASE 2017: Deep learning techniques for audio representation and classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2017.

[LPLH17] Hyungui Lim, Jeongsoo Park, Kyogu Lee, and Yoonchang Han. Rare sound event detection using 1d convolutional recurrent neural networks. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[MHD$^+$17] Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Benjamin Elizalde, Ankit Shah, Emmanuel Vincent, Bhiksha Raj, and Tuomas Virtanen. Dcase 2017 challenge setup: Tasks, datasets and baseline system. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017.

[MHV16a] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162, 2016.

[MHV16b] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Tut database for acoustic scene classification and sound event detection. In *24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132. IEEE, 2016.

[MRL$^+$15] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[MTX$^+$16] Erik Marchi, Dario Tonelli, Xinzhou Xu, Fabien Ringeval, Jun Deng, Stefano Squartini, and Björn Schuller. Pairwise decomposition with deep neural networks and multiscale kernel subspace learning for acoustic scene classification. In *24th Acoustic Scene Classification Workshop 2016 European Signal Processing Conference (EUSIPCO)*, pages 65–69, 2016.

[OBLS14] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[PHH+17] Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, Tuomas Virtanen, et al. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.

[PHMM16] Huy Phan, Lars Hertel, Marco Maass, and Alfred Mertins. Robust audio event recognition with 1-max pooling convolutional neural networks. *arXiv preprint arXiv:1604.06338*, 2016.

[PHV16] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Recurrent neural networks for polyphonic sound event detection in real life recordings. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6440–6444. IEEE, 2016.

[Pic15] Karol J Piczak. Environmental sound classification with convolutional neural networks. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, pages 1–6. IEEE, 2015.

[PJAM15] Gerald Bradley Penn, Hui Jiang, Ossama Abdelhamid Mohamed Abdelhamid, and Abdel-rahman Samir Abdel-rahman Mohamed. System and method for applying a convolutional neural network to speech recognition, November 17 2015. US Patent 9,190,053.

[PKBGM17] Huy Phan, Martin Krawczyk-Becker, Timo Gerkmann, and Alfred Mertins. DNN and CNN with weighted and multi-task loss functions for audio event detection. 2017.

[PKK+17] Huy Phan, Philipp Koch, Fabrice Katzberg, Marco Maass, Radoslaw Mazur, Ian McLoughlin, and Alfred Mertins. What makes audio event detection harder than classification? In *Signal Processing Conference (EUSIPCO), 2017 25th European*, pages 2739–2743. IEEE, 2017.

[PKVK13] C Poonkuzhali, R Karthiprakash, S Valarmathy, and M Kalamani. An approach to feature selection algorithm based on ant colony optimization for automatic speech recognition. *International journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 11 (2)*, 2013.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[PT17] Poorva G Parande and TG Thomas. A study of the cocktail party problem. In *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pages 1–5. IEEE, 2017.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[pyt] The python tutorial. `https://docs.python.org/2/tutorial`. "Accessed: 2018-05-01".

[Pyt09] January Python. Python (programming language). *Python (programming Language) 1 CPython 13 Python Software Foundation 15*, page 1, 2009.

[RD17] Anravich Ravichandran and Samarjit Das. Bosch rare sound events detection systems for DCASE2017 challenge. Technical report, Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017), September 2017.

[SB14] Jan Schluter and Sebastian Bock. Improved musical onset detection with convolutional neural networks. In *2014 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*, pages 6979–6983. IEEE, 2014.

[SB17] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[SG15] Jan Schlüter and Thomas Grill. Exploring data augmentation for improved singing voice detection with neural networks. *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 121–126, 2015.

[SGB+15] Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D Plumbley. Detection and classification of acoustic scenes and events. *IEEE Transactions on Multimedia*, 17(10):1733–1746, 2015.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[Smi99] Steven W.. Smith. *The scientist and engineer's guide to digital signal processing.* California Technical Publication, 1999.

[SWM17] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[TGPVG16] Naoya Takahashi, Michael Gygli, Beat Pfister, and Luc Van Gool. Deep convolutional neural networks and data augmentation for acoustic event detection. *arXiv preprint arXiv:1604.07160*, 2016.

[TMZ+06] Andrey Temko, Robert Malkin, Christian Zieger, Dušan Macho, Climent Nadeu, and Maurizio Omologo. Clear evaluation of acoustic event detection and classification systems. In *International Evaluation Workshop on Classification of Events, Activities and Relationships*, pages 311–322. Springer, 2006.

[VBGS17] Rene Vidal, Joan Bruna, Raja Giryes, and Stefano Soatto. Mathematics of deep learning. *arXiv preprint arXiv:1712.04741*, 2017.

[vHA09] PWJ van Hengel and Jörn Anemüller. Audio event detection for in-home care. In *International Conference on Acoustics (NAG/DAGA)*, pages 618–620, 2009.

[VW16] Toan H. Vu and Jia-Ching Wang. Acoustic scene and event recognition using recurrent neural networks. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016.

[WL17] Jun Wang and Shengchen Li. Multi-frame concatenation for detection of rare sound events based on deep neural network. Technical report, Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017), September 2017.

[Wu17] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 2017.

[XKH+17] Yong Xu, Qiuqiang Kong, Qiang Huang, Wenwu Wang, and Mark D Plumbley. Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging. *arXiv preprint arXiv:1703.06052*, 2017.

[Zho17] Jianchao Zhou. Sound event detection in multichannel audio LSTM network. Technical report, Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016), September 2017.

[ZSV14] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.