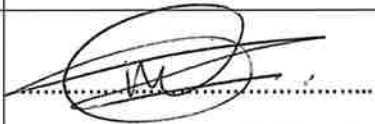


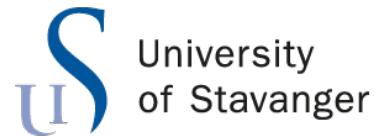


University of
Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialization: Applied Data Science	Spring semester, 2021 Open/Confidential
Author: Maalidefaa Moses Tantuoyir	 (Signature of author)
Programme coordinator: Supervisor(s): Associate Professor II Øyvind Meinich-Bache & Professor Kjersti Engan	
Title of master's thesis: Sound event detection from AED in team training situations using DNNs	
Credits: 30	
Keywords: Automated External Defibrillator, Sound Event Detection, Deep Neural Network, Convolutional Neural Network, Binary Classification, Multiclass Classification, ResNet	Number of pages: 110 + Supplemental material/other: 26 Date/year: Stavanger, 15-06-2021



Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Sound event detection from AED in team training situations using DNNs

Master's Thesis in Applied Data Science
by

MAALIDEFAA MOSES TANTUOYIR

Supervisor I

Øyvind Meinich-Bache

Supervisor II

Kjersti Engan

June 15, 2021

'Talent is universal; opportunity is not'

- Rye Barcott

Abstract

Cardiovascular diseases are the second most common cause of deaths in Norway in 2019. An increase in out-of hospital cardiac arrests cases due to an increase in reported cases in Norway between 2013 and 2018 has also been reported. The need for training health care professionals in delivering cardiopulmonary resuscitation(CPR) contributes significantly to safe care. The coordination of actions and skills during CPR impacts the survival of patients. Currently, mannequins developed by Lærdal Medical AS and used for team training events cannot detect sound events that originate from external sources.

The primary objective of this thesis is to use machine learning and deep neural networks(DNN) to automatically detect specific causes of different pauses during team training to support debrief. The automated external defibrillator(AED) audio clips are provided by Lærdal Medical AS. Background noise(audio clips) are overlaid on the AED audio clips to form close to realistic audio from team training scenario. Audio clips are converted to log-scaled mel spectrograms to form the final datasets. Four DNN architectures have been used; two convolutional neural networks, one convolutional recurrent neural network(CRNN) and pretrained ResNet18. In addition, two binary classification methods and a multiclass classification is experimented on six distinct AED audio classes.

Binary classification of AED audio using ResNet18 gave the best overall accuracy of 86,7 percent and an F1 score of 0.85. This model is further proposed in a final system for further development and implementation in the real world.

Acknowledgements

I would like to genuinely thank both my supervisors, Kjersti Engan and Øyvind Meinich-Bache for their support and guidance throughout the period of this thesis work.

I would also like to express my sincere gratitude to all colleagues and friends especially Kobina A. Quansah and Luca Tomasetti, who constantly supported me all time during the realization of this thesis. I do appreciate all the help and kind words especially in this COVID-19.

I dedicate it to my aunt, Grace Warisa, who was eagerly waiting to see me finish this work and visit home, but she unfortunately could not live to see the end of this work. Finally, a special mention goes to all members of my family, who supported me throughout my life decisions. This thesis was realized just for you, to show all my love and my appreciation that I could not demonstrate in the last 4 years 10 months, far away from home.

Ni barka yaga za!

Contents

Abstract	vi
Acknowledgements	viii
Abbreviations	xiii
List of Figures	xv
List of Tables	xix
1 Introduction	1
1.1 Motivation	1
1.2 Problem Definition	2
1.3 Outline	3
2 Background	5
2.1 Medical Background	5
2.1.1 Cardiac Arrest	5
2.1.2 AED in Team Training	6
2.2 Audio Signal Processing	7
2.2.1 Audio Signal Properties and Characteristics	7
2.2.2 Audio Signal Representations	10
2.2.2.1 Fourier Transform	10
2.2.2.2 Spectrogram	12
2.2.2.3 Mel Scale	12
2.2.3 Audio Features for Classification	13
2.2.3.1 Log-scaled Mel Spectrogram	13
2.2.3.2 Mel-Frequency Cepstral Coefficients (MFCC)	13
2.2.4 Sound Event Detection and Related Works	14
2.3 Technical Background and Deep Neural Networks	16
2.3.1 Machine Learning	16
2.3.2 Neural Networks(NNs)	16
2.3.3 Deep Learning	20
2.3.4 Deep Neural Networks(DNNs)	21
2.3.5 Convolutional Neural Network(CNN)	21
2.3.6 Recurrent Neural Network(RNN)	23

2.3.7	Pretrained Models and Transfer Learning	24
2.3.8	Hyperparameter Tuning	24
2.3.9	Statistical Information and Metrics	25
3	Dataset	27
3.1	Overview of Dataset	28
3.1.1	AED Audio Dataset	28
3.1.2	Background Noise Dataset	29
3.1.3	Selection of AED audio Classes	29
3.1.4	Polyphonic Audio Mixing	30
3.1.4.1	Creation of Final Dataset	31
3.2	Feature Extraction	32
3.2.1	MFCC	32
3.2.2	Log-scaled Mel Spectrogram	33
3.3	Train, Validation and Testing Set	34
3.4	Contribution to Dataset and Preprocessing	35
4	Methodology	37
4.1	Introduction	38
4.1.1	Initial Proposed System	38
4.1.2	Model Baseline Approach	39
4.2	Proposed Architectures	39
4.2.1	Architecture A	39
4.2.2	Architecture B	42
4.2.3	Architecture C	44
4.3	Transfer Learning Models	45
4.3.1	ResNet18	46
4.4	Binary Classification Methods	46
4.5	Multiclass Classification Methods	48
5	Experiments and Results	49
5.1	Introduction	50
5.1.1	Experimental Setup	50
5.1.2	Choice of Hyperparameters	51
5.1.2.1	Learning Rate	52
5.1.2.2	Dropout	52
5.2	Trial Experiment for Superior Features	52
5.3	Presentation of Results	53
5.3.1	Experiment 1: Binary Classification (1-versus-others)	53
5.3.1.1	Architecture A: Binary (1-versus-others)	53
5.3.1.2	Architecture B: Binary (1-versus-others)	55
5.3.1.3	Architecture C: Binary (1-versus-others)	58
5.3.1.4	ResNet18: Binary (1-versus-others)	61
5.3.2	Experiment 2: Binary Classification (1-versus-1)	63
5.3.2.1	Architecture A: Binary Classification (1-versus-1)	64
5.3.2.2	Architecture B: Binary Classification (1-versus-1)	64
5.3.2.3	Architecture C: Binary Classification (1-versus-1)	65

5.3.2.4	ResNet18: Binary Classification (1-versus-1)	65
5.3.3	Experiment 3: Multiclass Classification	66
5.3.3.1	Architecture A: Multiclass Classification	66
5.3.3.2	Architecture B: Multiclass Classification	68
5.3.3.3	Architecture C: Multiclass Classification	69
5.3.3.4	ResNet18: Multiclass Classification	71
5.4	Analysis of Results	72
5.4.1	Experiment 1: Binary Classification (1-versus-others)	72
5.4.2	Experiment 2: Binary Classification (1-versus-1)	75
5.4.3	Experiment 3: Multiclass Classification	75
6	Discussion and Future Works	77
6.1	Model Performance	78
6.1.1	Experiment 1: Binary Classification (1-versus-others)	78
6.1.2	Experiment 2: Binary Classification (1-versus-1)	79
6.1.3	Experiment 3: Multiclass Classification	79
6.2	Proposed System	79
6.3	Limitations	80
6.3.1	Volume Reduction	81
6.4	Future Work	81
6.4.1	Network Architectures	81
6.4.2	Realistic Data Material	81
6.4.3	More Dataset per Class	81
7	Conclusion	83
A	Appended Codes	85
A.1	Generating Audio Chunks	85
A.2	Reduce Volume of AED Audio	85
A.3	Create Polyphonic Audio Mix	86
A.4	Create Features	86
A.5	Import Dataset	86
A.6	Architectures	86
B	Plots for Binary Classification(1-versus-others)	87
B.1	Plots for Architecture A	87
B.2	Plots for Architecture B	89
B.3	Plots for Architecture C	91
B.4	Plots for ResNet18	93
C	Sample Spectrograms	97
C.1	Log-scaled mel spectrogram samples	97
C.2	MFCC spectrogram samples	98
	Bibliography	101

Abbreviations

ACC	ACC uracy
Adam	Ad aptive M oment Estimation
AED	A utomatic E xternal D efibrillator
ANN	A rtificial N eural N etwork
AUC	A rea U der the C urve
BN	B atch N ormalization
CCE	C ategorical C ross E ntropy
CCF	C hest C ompression F raction
CD	C ompact D isk
CNN	C onvolutional N eural D isk
CPR	C ardio P ulmonary R esuscitation
CRNN	C onvolutional R ecurrent N eural N etwork
dB	deci B el
dBFS	deciBel F ull S cale
DCT	D iscrete C osine T ransform
DFT	D iscrete F ourier T ransform
DNN	D eep N eural N etwork
ESC	E nvironmental S ound C lassification
FFNN	F eed F oward N eural N etwork
FFT	F ast F ourier T ransform
FN	F alse N egative
FP	F alse P ositive
FPR	F alse P ositive R ate
GRU	G ated R ecurrent U nit
Hz	H ertz

ICD	I mplantable C ardioverter D efibrillator
ICU	I ntensive C are U nit
kB	kilo B yte
kHz	kilo H ertz
LSTM	L ong S hort T erm M emory
MFCC	M el F requency C epstral C oefficient
Nadam	N esterov Momentum into adam
NN	N eural N etwork
PPV	P ositive P recision V alue
ReLU	R ectified L inear U nit
ResNet	R esidual N etwork
RGB	R ed G reen B lue
RNN	R ecurrent N eural N etwork
ROC	R eceiver O perating C haracteristic
ROSC	R eturn to S pontaneous C irculation
S	S econds
SED	S ound E vent D etection
SGD	S tochastic G radient D escent
STFT	S hort- T erm F ourier T ransform
TN	T rue N egative
TP	T rue P ositive
TPR	T rue P ositive R ate
WCD	W earable C ardioverter D efibrillator

List of Figures

1.1	An illustration of the approach used for the thesis	1
2.1	AED device from Lærdal Medical. Reprinted with permission from Lærdal Medical AS.	6
2.2	Team training scenario. Reprinted with permission from Lærdal Medical AS	7
2.3	Figures of some properties of sound. Reprinted with permission from [1]	8
2.4	Waveform plot of 'No Shock Advised'	10
2.5	Graphical explanation of Fourier Transform. Reprinted with permission from [2]	11
2.6	Spectrum for 'no shock advised', $sample_rate = 44100$, $number_of_FFT = 69418$	11
2.7	Illustration of how spectrograms are formed and an example of a spectrogram.	12
2.8	Mel versus Log-scaled Mel Spectrograms	13
2.9	Illustration of MFCC generation	14
2.10	Examples of MFCCs from AED audio clips	15
2.11	Representation of NNs with different layers	16
2.12	Example of a CNN architecture. Reprinted with permission from [3]	21
2.13	Illustration of Max Pooling and Average Pooling. The figure shows an example of max pooling operation and average pooling with a 2x2 pixel filter size from 4x4 pixel input. At max pooling, each filter is taken the maximum value, then arranged into a new output with a size of 2x2 pixels. While the average pooling value taken is the average value of the filter size. The figure is reprinted in unaltered form from: "Using convolutional neural networks for image recognition" [4]	22
2.14	Illustration of a RNN structure.	24
2.15	Confusion Matrix	25
2.16	Illustration of ROC curve and AUC	26
3.1	An illustration of the approach used for the thesis	27
3.2	MFCC examples	33
3.3	Log-scaled Mel Spectrogram examples	34
3.4	Illustration of how the folders are arranged. For each labels folder, the training, validation and tests sets are stored in separate sub-folders.	34
4.1	An illustration of the approach used for the thesis	37
4.2	Initial system proposed for AED sound event detection in team training scenario	38
4.3	AlexNet style of CNN model architecture A	40
4.4	AlexNet style of CRNN model architecture B	42
4.5	AlexNet style of CNN model architecture C	44

4.6	Summary of ResNet18 architecture	46
4.7	Illustration of binary classification methods	47
4.8	Illustration of multiclass classification method using 6 classes	48
5.1	An illustration of the approach used for the thesis	49
5.2	Flow diagram of experimental setup	51
5.3	Epochs versus Accuracy Graphs. The graphs show two features used for training and validation of P17 class	52
5.4	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background10db. Early stopping can be applied after 40 epochs to avoid overfitting for all the classes.	54
5.5	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background15db. Using Reduced15db, the model runs for longer epochs before overfitting starts to show. Early stopping after 65 epochs can be applied.	55
5.6	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background10db	57
5.7	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(right hand side) and epochs versus loss(left hand side) of the training and validation of P03, P17 and background15db	58
5.8	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background10db	59
5.9	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background15db	60
5.10	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background10db	62
5.11	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background15db	63
5.12	Comparison of binary classification of P13 and P17 using architecture A. The graphs display epochs versus accuracy and loss graphs for Reduced15db	64
5.13	Comparison of binary classification of P13 and P17 using architecture B. The graphs display epochs versus accuracy and loss graphs for Reduced15db	65
5.14	Comparison of binary classification of P13 and P17 using architecture C. The graphs display epochs versus accuracy and loss graphs	65
5.15	Comparison of binary classification of P13 and P17 using ResNet18. The graphs display epochs versus accuracy and loss graphs	66
5.16	Architecture A training and validation plots for multiclass classification. The graphs display epochs versus accuracy and loss graphs	67
5.17	Architecture B training and validation plots for multiclass classification. The graphs display epochs versus accuracy and loss graphs	69

5.18	Architecture C training and validation plots for multiclass classification. The graphs display epochs versus accuracy and loss graphs	70
5.19	ResNet18 training and validation plots for multiclass classification. The graphs display epochs versus accuracy and loss graphs	71
5.20	Analysis of mean test and F1 Scores for experiment 1	73
5.21	Analysis of Mean Test and F1 Scores for all architectures used in experiment 1	74
6.1	An illustration of the approach used for this chapter of the thesis	77
6.2	Model versus average time per epoch graph. The average training time for Reduced10db is higher than in Reduced15db. Architecture C trains the fastest while ResNet18 is the slowest training model.	78
6.3	Diagram of the proposed system for sound event detection in team training. The system takes in real time audio from the team training event and divide the audio into chunks as a function of time(t). The audio chunks pass through a preprocessing stage where they are converted into log-scaled mel spectrograms. The spectrograms are then serve as an input into the model. The final predictions are interpreted as a function of time.	80
B.1	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25	88
B.2	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25	89
B.3	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25	90
B.4	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25	91
B.5	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25	92
B.6	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25	93
B.7	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25	94
B.8	Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25	95
C.1	Log-scaled mel spectrograms. These are examples of log-scaled mel spectrograms used for the experimentation	98
C.2	MFCC spectrograms. These are examples of MFCC spectrograms used for the experimentation	99

List of Tables

3.1	AED Audio Event Properties	28
3.2	Distribution of background noise dataset	29
3.3	Reduced10db outlook	31
3.4	Reduced15db outlook	32
4.1	Summary of layers of architecture A	41
4.2	Summary of layers of architecture B	43
4.3	Summary of layers of architecture C	45
5.1	Binary classification results for architecture A	53
5.2	Binary classification results for architecture B	56
5.3	Binary classification results for architecture C	60
5.4	Binary classification results for ResNet18	61
5.5	Model performance results for architectures A, B, C and ResNet18. The table shows the results obtained by binary classification of two similar classes using Reduced15db dataset	66
5.6	Multi-class classification metrics for architecture A	67
5.7	Multi-class Classification Report: Architecture A	68
5.8	Multiclass classification metrics for architecture B	68
5.9	Multiclass Classification Report: Architecture B	68
5.10	Multiclass classification metrics for architecture C	69
5.11	Multiclass Classification Report: Architecture C	70
5.12	Multiclass classification metrics for ResNet18	71
5.13	Multi-class Classification Report: ResNet18	72
5.14	Mean Test Accuracy and F1 Score Results. The results compare the mean test accuracy and f1 score between the classes used in experiment 1	72
5.15	Mean Test Accuracy and F1 Score Results. The results compare the mean test accuracy and F1 score between the architectures used in experiment 1	74

1

Introduction

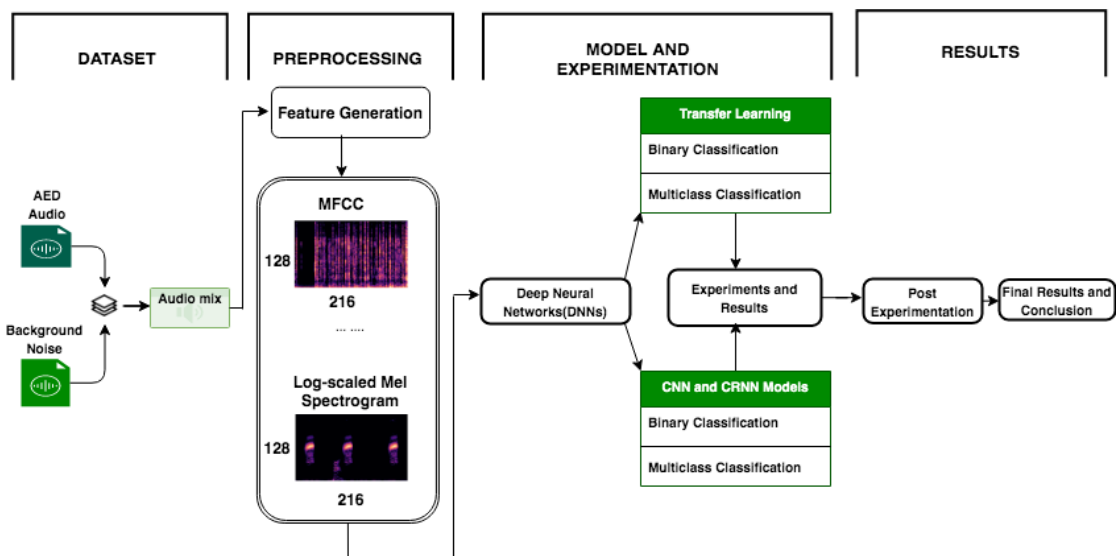


Figure 1.1: An illustration of the approach used for the thesis

1.1 Motivation

Cardiac arrest is the abrupt loss of heart function in a person who may or may not have been diagnosed with heart disease [5]. This is caused by different factors which

include; heart attack, overdose, cardiomyopathy, electrocution etc. The resultant effect of a cardiac arrest inadvertently leads to problems with memory, permanent brain damage, death and in many cases high healthcare cost to the patient and society. Cardiopulmonary resuscitation (CPR) is needed to keep blood and oxygen circulating to the brain and through the body when there is a cardiac arrest. A defibrillator will deliver a controlled electric shock, as an attempt to get the heart beating normally again. For defibrillation to be effective, the heart needs to be oxygenated and CPR is usually crucial prior to shock. The coordination of actions and skills during CPR are huge factors that impact survival, as the chest compression fraction (CCF) (time you spend compressing) is directly connected with the probability of return to spontaneous circulation (ROSC) of cardiac arrest. The ability to resuscitate is also directly related to the time, type and sequence of therapy provided to the patient [6].

An increase in out-of-hospital cardiac arrests cases have been reported between 2013 and 2018 in Norway. The number of cases jumped from 1101 to 3400 per year [7] due to the fact, there is an increase in reporting of cardiac arrests. In 2019, cardiovascular diseases (of which cardiac arrest falls under) was the second most common cause of deaths in Norway. 31.8% of global deaths in 2017 was caused by heart related diseases [8].

Training of health care professionals in delivering CPR helps to increase their awareness of various clinical practices, motivates learning and most importantly contributes to safe care [9]. Abella et al. [10] further empathised on the need for effective and quality training for healthcare professionals.

1.2 Problem Definition

In order to effectively train high performance CPR to improve CCF, it is important to detect actions around the most common pauses, which are usually related to the use of a defibrillator. During a cardiac arrest event, healthcare providers may use an Automated External Defibrillator (AED) or a manual defibrillator. In both cases, there are pauses associated to the use of the device that can be identified by the sounds of it. Events like 'analyzing', 'charging' and 'shock' have specific sounds or voice prompts. The current mannequins developed by Lærdal Medical AS cannot detect these events as they are directly coming from external devices, thus the use of machine learning in a form of sound event detection (SED), to identify specific causes of different pauses during team training would better support debrief. This also creates an opportunity to support therapy cases in the near future.

In the literature, various solutions have been proposed to solve SED problems [11] [12] [13] [14]. This thesis explores the use of convolutional neural networks(CNNs)and convolutional recurrent neural networks(CRNNs) to detect AED sound events in noisy background environments that mimic a realistic case in medical team-training scenarios. A CNN and CRNN architectures are proposed for both binary and multiclass classification of some chosen AED sound events. Firstly, the AED audio clips are overlaid on sourced background noises and then, the resultant audio mixture is converted to log-mel spectrograms and mel-frequency cepstral coefficients(MFCCs). These image features are then used as the dataset for model training, validation and testing. Also, some pre-trained models available are explored by transfer learning to solve the proposed problem.

1.3 Outline

Figure 1.1, presents a general overview of how the thesis work is structured and actualised. With the exception of this chapter which provides a brief introduction of the thesis, there are 6 chapters. The following is a short description of each chapter:

Chapter 2, introduces a general background of polyphonic SED, the methods that have been used to solve similar tasks, technical background of deep neural networks and a description of the transfer learning methods explored in this work.

Chapter 3, contains the description of the dataset used for the thesis work. It describes the AED audio clips obtained from Lærdal Medical AS, background noise data gathered, polyphonic sound mixture and my contribution to the data gathering process.

Chapter 4, outlines binary and multiclass classification methods and a detailed algorithmic flow of the CNN and CRNN models, pretrained models and outline of model architectures.

Chapter 5, presents the experiments conducted and the results achieved from these experiments and analysis of the results.

Chapter 6, discusses the results and possible future works. Also, a final system is proposed in this chapter.

Chapter 7, concludes the thesis and contains the reflections of the overall results obtained.

2

Background

This chapter gives a brief explanation of AED team training, audio signal processing, SED and related works, and technical background of DNNs; they are useful to understand the reasons why the proposed methods are chosen and how they might be necessary for future applications in mannequins. More detailed information of the arguments covered in this chapter can be found in the bibliography 6.

2.1 Medical Background

This section explains briefly the medical background of the thesis work.

2.1.1 Cardiac Arrest

Cardiac arrests occurs when the heart is not able to pump blood to the body. This could be caused by chaotic electrical activity that does not make the heart pump blood, referred to as ventricular fibrillation. An electrical shock might be able to get the heart beat back in a pulse-giving rhythm again. If ventricular fibrillation is untreated, it can result into asystoly(no electrical activity) and the consequence is death [15][16]. A defibrillator is needed for immediate treatment in situations when the heart does not spontaneously

return to pulse rhythm. This device is used to send electric shocks to the heart which in-turn, could bring the heart to a normal heartbeat. The main types of defibrillators that are used are; AED, Implantable Cardioverter Defibrillator (ICD), Wearable Cardioverter Defibrillator (WCD). The picture below(Figure 2.1) is an example of an AED device.



Figure 2.1: AED device from Lærdal Medical. Reprinted with permission from Lærdal Medical AS.

2.1.2 AED in Team Training

The development of mannequin simulators for education, training and research purposes, is of immense importance to the medical sector[17]. Team training significantly improves healthcare processes and patient outcomes[18][19]. Timely access to AEDs at training, sporting competitions, military etc. permits effective management of sudden cardiac arrests and the prevention of sudden cardiac death[20][21][22].

Research has also shown that, training medical professionals, emergency response teams etc., has the potential to increase survival rates from out-of-hospital cardiac arrest[23]. Woollard et. al [23] and Bircken et. al [24] went further on to recommend routine training of care providers after they detected a decline in their basic skill competences after 4-6 months of initial training. Facilitating or guiding a reflection in the cycle of an experimental learning is known as debriefing. Fanning et. al [25] generalised that, debriefing is the fundamental essence of simulation experience. There is also an increasing body of work exploring the role and effectiveness of debriefing in an objective manner

in the learning process. This thesis work proposes a way to make a debriefing process efficient detecting sound events from an AED device from a phone while recording in a team-training simulation. The advantage is that, it can improve insights into strategies and processes that lead to good results and also to clarify learning points. Figure 2.2 shows an example of a team training simulation with an AED device.



Figure 2.2: Team training scenario. Reprinted with permission from Lærdal Medical AS

2.2 Audio Signal Processing

Audio is the reproduction of sound. Sound is a variation of a quantity in air pressure over time. Audio signal is sound measured as an electrical signal in volt or milli-amperes[mA]. A digital audio signal is a sampled and quantified version of audio signal represented by bits per sample. Human sense of hearing provides rich information about the environment with respect to the characteristics and locations of the source of the sound[26]. There is the need for segregation and classification of this information whether by humans (auditory scene analysis) or by machines.

There are three main technological application areas in digital audio signal processing. They include: audio data compression, audio effects synthesis and audio classification. In the following subsections, we will review some methods for signal processing of audio in relation to audio classification.

2.2.1 Audio Signal Properties and Characteristics

Audio signal properties (Figure 2.3) include:

- Wavelength
- Amplitude
- Frequency
- Period
- Speed

Wavelength is the distance between repeating units of a wave pattern.

Amplitude is the strength or power of a wave signal. It can be seen as the 'height' of a wave when visualised as a graph. Higher amplitudes are normally interpreted as a higher volume.

Frequency is the number of times a wavelength occurs in one second. It is measured in Hertz(Hz)/ cycles per second. The faster the vibration from the source of sound, the higher the frequency and the higher the pitch.

Speed is the product of the frequency and wavelength.

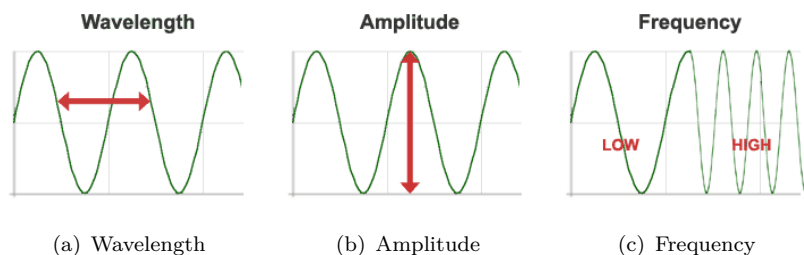


Figure 2.3: Figures of some properties of sound. Reprinted with permission from [1]

Our auditory system response to frequencies from the range of 20 [Hz] to 20 [kHz] which represents the number of sound vibrations in one second. The more amplitude a sound has, the louder it is. The lowest sound one can hear is of 0 [dB] and above a sound intensity of 120 [dB], it is inaudible for a human being to comprehend. In real life comparison, it could translate to hearing the crushing sound of a leaf on the ground to the high intensity sound of jet fighter flying just above a person.

Sound is recorded by sampling and quantisation of electrical outputs of the microphone. Most recorded sounds have a sample rate of 44,1 [kHz], which makes it possible to synchronize the audio with video data. Sampling at a frequency above 40 [kHz] is adequate to capture the full range of audible frequencies because according to the sampling theorem, we can reproduce frequencies up to half of the sampling frequency. Therefore, since we can hear 20 [kHz], we need to sample at $2 \times 20 = 40$ [27]. The following

are some parameters to consider for recorded sound and also, to understand the audio clips which would be used in this thesis.

Volume: is the intensity of sound waves, or how loud a sound is. It is measured in decibels[dB]. dB indicates a logarithmic ratio of a physical quantity. This can be intensity, pressure on the air, power etc. Since we are using recorded audio, the volume referred to in this thesis work is in decibel full scale[dBFS].dBFS is used in digital signals to indicate the limit before the sound waves get smashed due to clipping. 0 [dBFS] is the limit where a sound wave can be fully reproduced before it is being distorted due to clipping. The lowest point is when when the volume is so low that, there is no difference with the noise produced by every device. This level is called 'noise floor' and it is dependant on the bitrate of the audio. For example, the AED audio clips have 16 bits per sample. Therefore they will have 'noise floor' at -96 [dBFS], meaning that from -96[dBFS] to 0[dBFS] you will manage to get clear audio, and the 96 [dB] are called the dynamic range. Below -96 [dBFS], there will be noise, while above 0 [dBFS] will lead to distorted audio due to clipping.

Sample rate: This is the number of sample points per second, in the unit of Hertz [Hz]. A higher sample rate indicate better sound quality, but the storage space is also bigger. Commonly used sample rates are listed next:

- 8 [kHz]: Voice quality for phones and toys.
- 16 [kHz]: Commonly used for speech recognition.
- 44.1 [kHz]: CD quality

Bit resolution: The number of bits used to represent each sample point of audio signals. Commonly used bit resolutions are:

- 8-bit: The corresponding range is 0 255 or -128 127.
- 16-bit: The corresponding range is -32768 32767.

Channels: We have mono for single channel and stereo for double channels.

In this thesis, the AED audio samples used have a sample rate of 44100 [Hz], a bit resolution of 16 bits(2 bytes) and mono(single) channel. The average time duration is 1.5 seconds[s]. For example, 'no shock advised' audio has a sample rate of 44100 [Hz], 16 bits of bit resolution and sample points of 69440. Therefore the time duration is $69440/16000 = 1.574$ [s]. The file size is $69440 \times 2 = 138880$ bytes which is equivalent to 139 kilobytes

[KB]. Figure 2.4 shows the amplitude versus time plot of the above mentioned audio signal.

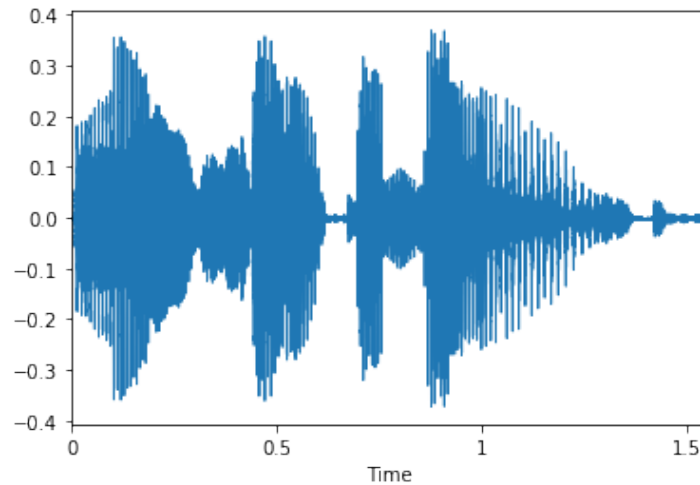


Figure 2.4: Waveform plot of 'No Shock Advised'

2.2.2 Audio Signal Representations

The digital signal in Figure 2.4 can be interpreted, modified and analyzed further for machine-based interpretation and audio classification. For the purpose of this thesis, we will focus on the how the signals are transformed from the time domain into the frequency domain through the Fourier Transform.

2.2.2.1 Fourier Transform

The Fourier transform is a mathematical formula that decomposes a signal from the time domain into individual frequencies and the frequency's amplitude (frequency domain) [28]. This is possible because every signal can be decomposed into a set of sine and cosine waves that add up to the original signal. The result is called a frequency spectrum. Figure 2.5 shows a graphical explanation of the Fourier transform.

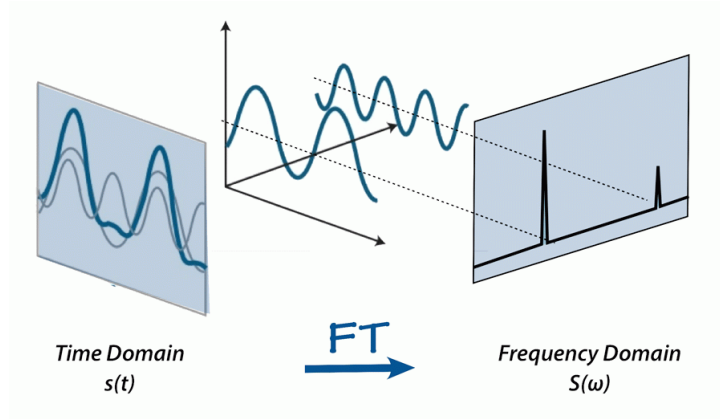


Figure 2.5: Graphical explanation of Fourier Transform. Reprinted with permission from [2]

In signal processing, there is an efficient way to compute the Fourier transform of an audio. This is referred to as fast-Fourier Transform(FFT) [29]. Given a discrete signal $x[n]$ where N is the size of the domain, we can first find the discrete Fourier Transform(DFT) by multiplying each of its value by an exponent(e) raised to some function n . The result is the sum for a given n . This is illustrated in the formula below:

$$x[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{N}}$$

FFT is an algorithm that determines the DFT of an input significantly faster than computing it directly. FFT reduces the number of computations needed for a problem of size N from $O(N^2)$ to $O(N \log N)$. Further details of DFT and FFT can be referred to in [29][30][31]. Figure 2.6 shows a spectrum for 'no shock advised.wav' file with $number_of_FFT(n_fft = 69418, sample_rate = 44100)$.

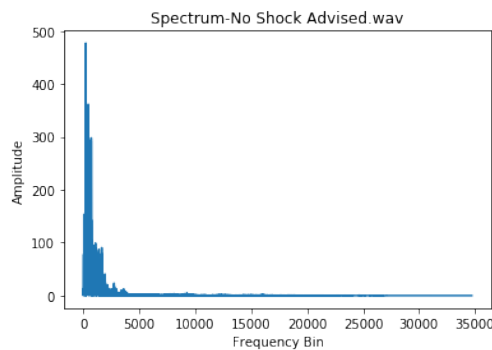
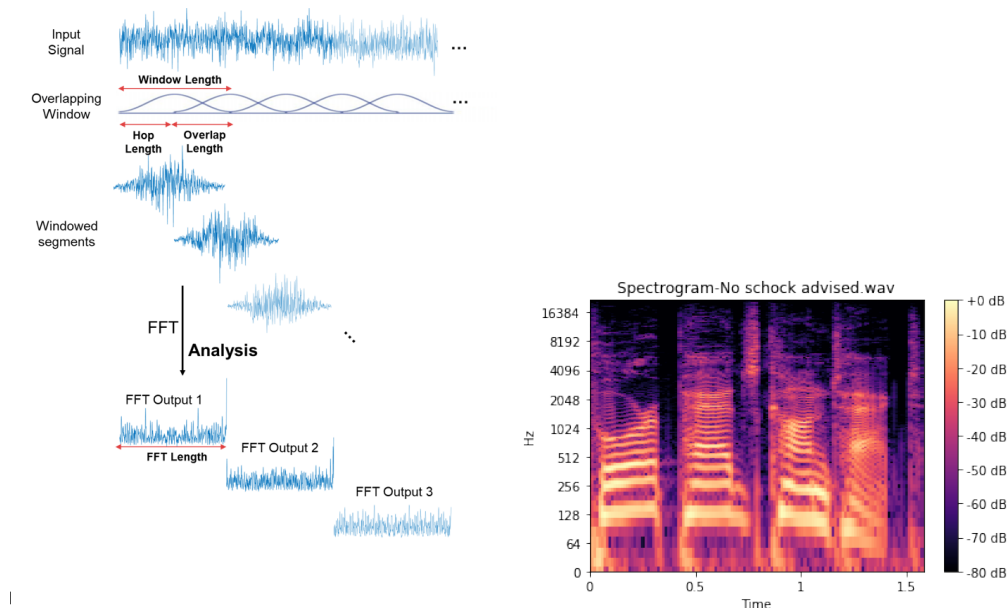


Figure 2.6: Spectrum for 'no shock advised', $sample_rate = 44100$, $number_of_FFT = 69418$

2.2.2.2 Spectrogram

Spectrograms can be computed by calculating the FFT of a signal over overlapping windowed segments of the signal. This method is called the short-term Fourier transform (STFT) [32]. A spectrogram can be seen as a lot of FFTs stacked on top of each other. It is a method to visually represent a signal's loudness or amplitude as it varies over time at different frequencies. Graphically, the y-axis is converted to a log scale and the color dimension is converted to decibels. Figure 2.7 shows an example of a spectrogram.



(a) Illustration to show how a spectrogram is formed. (b) Spectrogram: 'No Shock Advised', Reprinted with permission from [33] `sample_rate = 44100, hop_length = 512`

Figure 2.7: Illustration of how spectrograms are formed and an example of a spectrogram.

2.2.2.3 Mel Scale

Mel scale is a proposed unit of pitch by Stevens, Volkman and Newman [34]. They performed a mathematical operation on frequencies such that equal distances in pitch sounded equally distant to the listener. This stems from the fact that, humans are better at detecting differences in lower frequencies than higher frequencies. To give an example, one can easily detect the difference between 500 and 900 [Hz], but hardly will be able to tell apart 10,000 [Hz] and 10,400 [Hz], even though the distance between the two pairs are the same. This is used to construct the mel spectrogram and subsequently, the log-mel spectrogram, which would be used as a feature for the classification problems in Chapter 5.

2.2.3 Audio Features for Classification

This subsection deals with two main audio features used in audio classification in deep learning [35][36]. They are:

- Log-scaled Mel Spectrogram
- Mel-Frequency Cepstral Coefficients(MFCCs)

These are the two main feature sets generated for the thesis and details of these features would be found in Chapter 3.

2.2.3.1 Log-scaled Mel Spectrogram

Mel spectrogram is a spectrogram where the frequencies are converted to the mel scale. The the raw mel spectrogram is displays power in color. To get a log-scaled mel spectrogram which gives superior results in deep learning experiments[37][38], the decibels are log-scaled to give a brighter color indication in the resultant 'image'. The following diagrams show a comparison of the mel spectrogram and the log-scaled mel spectrogram of the same audio file('no shock advised').

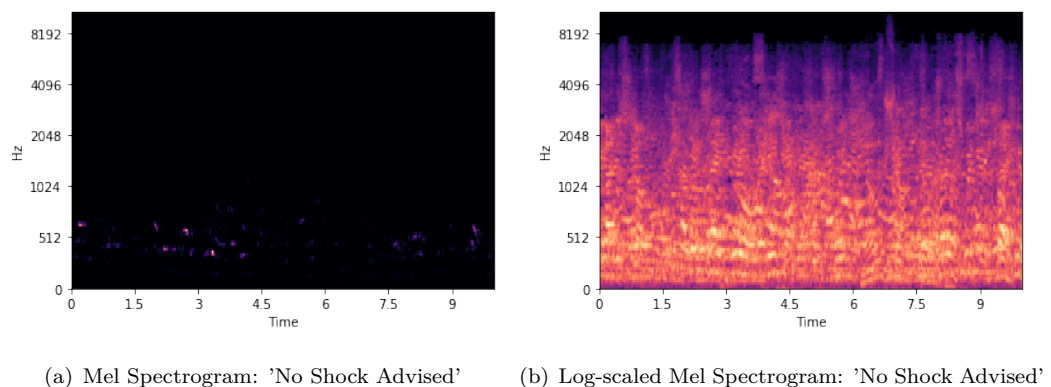


Figure 2.8: Mel versus Log-scaled Mel Spectrograms

2.2.3.2 Mel-Frequency Cepstral Coefficients (MFCC)

MFCC mimics the logarithmic perception of loudness and pitch of human auditory system and tries to eliminate speaker dependent characteristics by excluding the fundamental frequency and their harmonics[39]. A frequency measured in Hertz[Hz], can be converted to the mel scale using the following formula:

$$Mel(f) = 2595 \log \left(1 + \frac{f}{700} \right)$$

MFCC are coefficients that make up the mel-frequency cepstrum and they accurately represent the envelope of time power spectrum of the speech signal which is generated from our vocal tract. To get the MFCCs, an audio signal is first split into short-time frames. Then windowing is applied to counteract the assumption by Fast Fourier Transform(FFT) that, the data is infinite and reduce leakage. Thirdly, Discrete Fourier Transform(DFT) is applied to calculate the number of FFT and then the power spectrum is computed. The next step is to apply the mel filter banks and take the log of these spectrograms values to get the log mel filter bank energies. To decorrelate the mel filter bank coefficients, the Discrete Cosine Transform(DCT) is applied. The result is a list of coefficients that is termed as MFCCs. To get the mathematical details of how MFCC is generated, refer to [40]. Figure 2.9 shows a schematic explanation of how MFCC is generated.

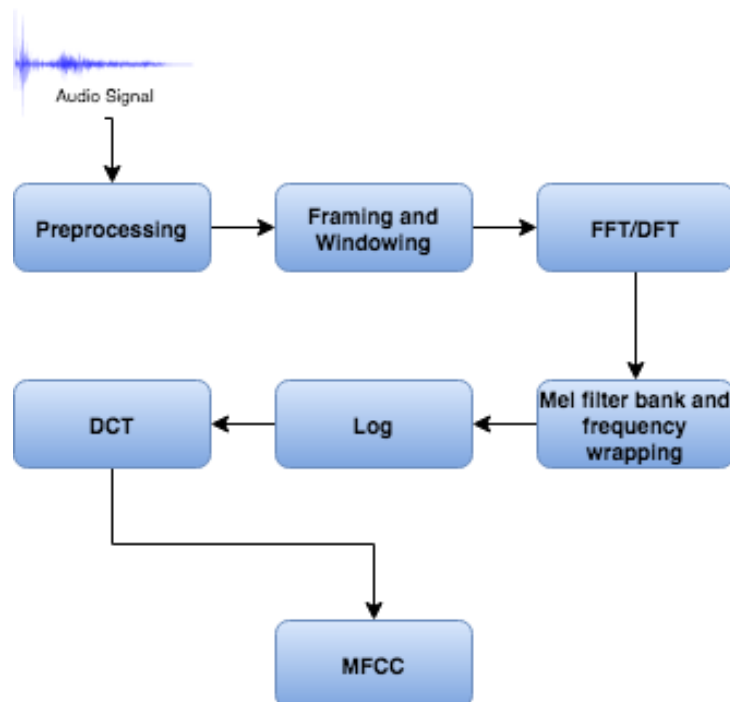


Figure 2.9: Illustration of MFCC generation

Figure 2.10 shows the MFCC representation of the sample audio 'No Shock Advised'.

2.2.4 Sound Event Detection and Related Works

Sound event detection is the identification of separate or individual sound events in an audio which may require the estimation of onset and offset for the identified instances in the sound event. There are several application areas of SED. They range from healthcare

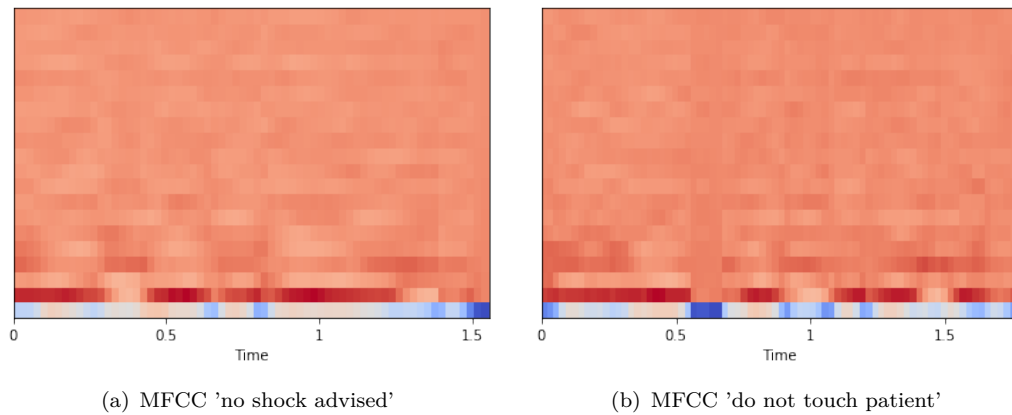


Figure 2.10: Examples of MFCCs from AED audio clips

and wildlife monitoring, surveillance to audio and video content-based indexing and retrieval [41][42][43].

SED can be a supervised learning approach where sound events are labelled as classes. This classification approach could be monophonic or polyphonic sound event detection. The former deals with the detection of the most prominent sound events at each time of an output sequence while the later is the detection of overlapping sounds[44]. DCASE 2013 [44] and TUT 2016 [14] established benchmark datasets for experimentation on polyphonic sound event detection.

Studies on polyphonic sound event detection has gained much attention within deep learning[45] [14] [46]. The ability of CNN to learn features from an image and make predictions based on the input data has attracted recent studies within this area. CNNs are widely used for polyphonic sound event detection [11][47][48] on various public available datasets. The use of CNNs with recurrent neural networks(RNNs) also referred to as convolutional recurrent neural networks(CRNNs), has been proposed and studied on both DCASE 2016 and TUT 2016 dataset[11] [13]. Transfer learning has also been explored in the field of sound event detection [49]. The most recent application is in cough detection in COVID-19 patients [50].

In all these methodologies and applications, there have been varying results in relation to the kind of dataset and data preprocessing steps applied in drawing to a conclusion. The most used feature extraction methods are MFCC and log-scaled mel spectrograms with the later achieving superior results than the former [11] [51]. On the basis of this, these features would be studied in Chapter 5.

2.3 Technical Background and Deep Neural Networks

This section gives a general overview of machine learning, CNNs, RNNs, DNNs and the theory behind them; transfer learning and the statistical metrics used in this thesis.

2.3.1 Machine Learning

Machine learning is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. In the past decade, machine learning has given us self-driving cars, practical speech recognition, facial recognition among others [52]. In machine learning, algorithms are 'trained' to find patterns and features in data in order to make decisions and predictions based on the results obtained. The better the algorithm, the more accurate the decisions and predictions will become as it processes more data. The need for better algorithms has led to the development of neural networks [53], which will be discussed in the next subsections.

2.3.2 Neural Networks (NNs)

Neural networks or artificial neural networks (ANNs) are mathematical structures that when given input can map to a desired output. They were developed to mimic the biological operation of human brain cells called neurons. The building block of neural networks is the perceptron which is similar to biological building blocks of lipids, nucleic acid, carbohydrates and proteins [54]. Figure 2.11 represents an outlook of a single and a multi-layer perceptron.

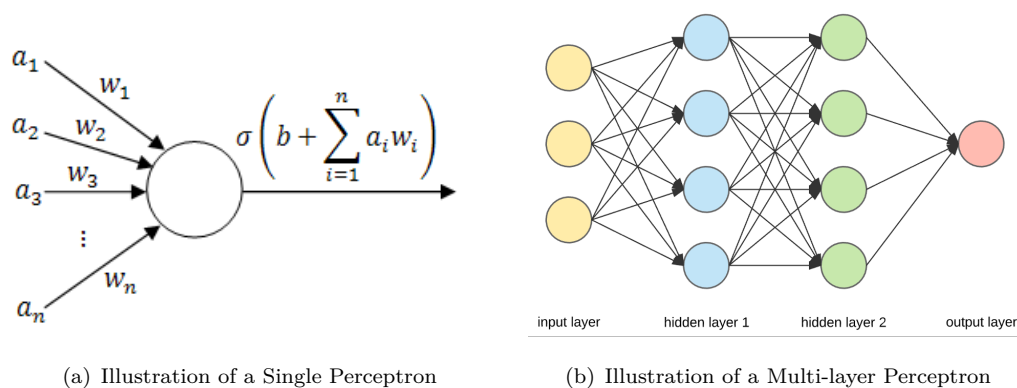


Figure 2.11: Representation of NNs with different layers

The output of a single neuron is given by the formula:

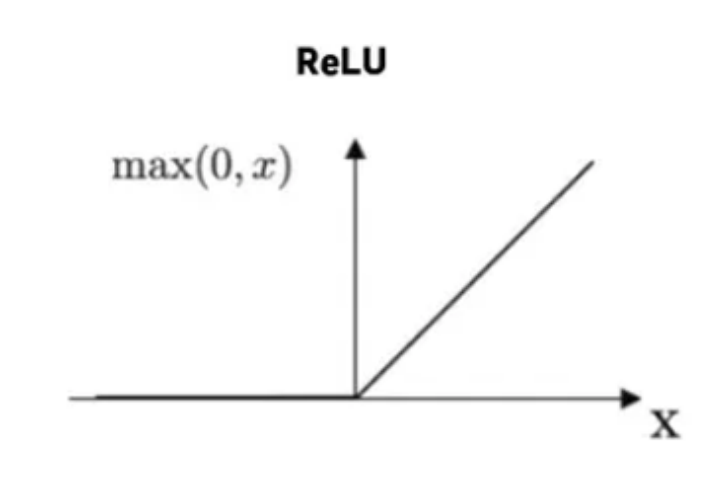
$$\hat{y} = \sigma \left(b + \sum_n^{i=1} a_i w_i \right)$$

where σ is the activation function, b is the bias, w_i is the weight and a_i is the input value. Figure 2.11(a) shows a visual representation of a simple NN. Three different input a_i are feeding the artificial neuron with three different weight values w_i . The artificial neuron has one layer. Figure 2.11(b) shows a Feed Forward Neural Network (FFNN). The FFNN contains two hidden layers and an output layer. The input layer consists of three artificial neurons, and the output layer has one artificial neuron. No calculation is required during the load of the input layer; thus building the FFNN would consist of implementing two computational layers. The inputs in the hidden layer are fully connected to the artificial neurons in the input layer. Moreover, a full connection is presented between the artificial neurons of the hidden layer and the artificial neurons in the output layer.

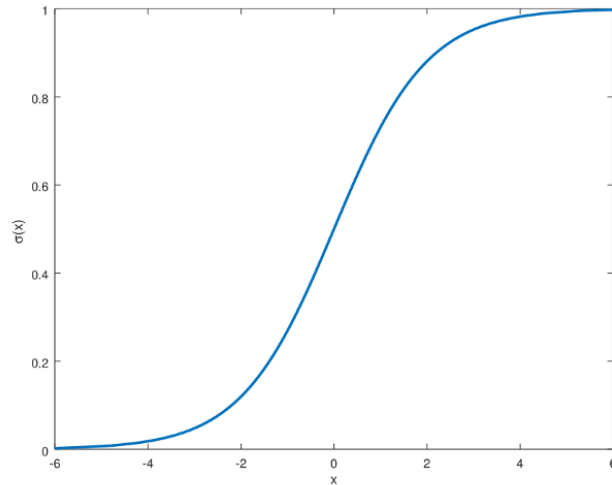
An activation function determines how much of a neuron's value is used in a calculation. There are several possible activation functions. Selecting the activation function has a great effect on the training dynamics and performance of the neural network[55] and different activation functions might be chosen depending on the layer network. Three activation functions that were used in this thesis are:

- Rectified Linear Unit (ReLU)[56]
- Sigmoid function[57]
- Softmax[58]

ReLU provides a straightforward nonlinear transformation. The function is defined as the maximum between 0 and a given element x . This activation function will be used in the CNN architectures in the preceding chapters.



Sigmoid function takes a range real number and returns the output value which falls in the range of 0 to 1. It produces the curve which will be in the Shape S . This function is used for binary classification in this thesis.



Softmax behaves similarly to the sigmoid function. The difference is that it outputs probabilities range. The range for each output node is from 0 to 1, and the sum of all the probabilities will be equal to one.

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}$$

for $i = 1, \dots, K$ and $x = (x_1, \dots, x_K) \in \mathbb{R}^K$

The softmax function computes the exponential power of the given input value x and the sum of exponential values of all the values in the inputs. Then the ratio of the exponential of the input value and the sum of exponential values is the output. It will be used in multiclass classification.

Loss Functions

To measure the error between the predicted and real outputs in a neural network, the loss function is used. Two main loss functions were used in this work. These are:

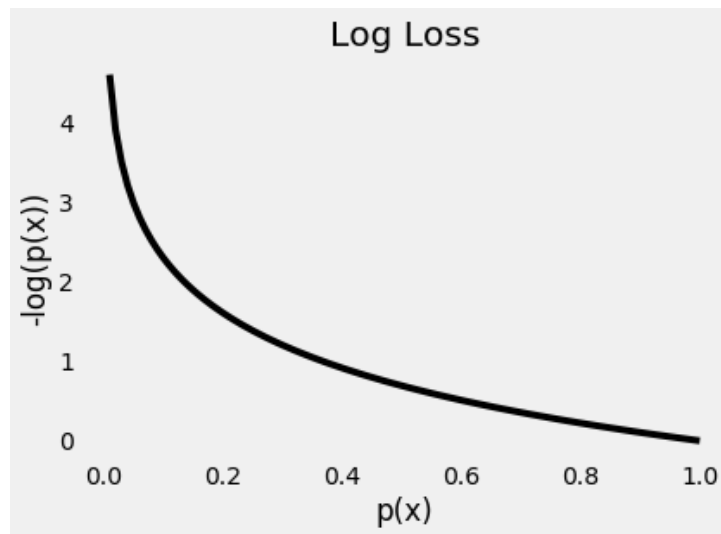
- Binary cross-entropy [59]
- Categorical cross-entropy [60]

Binary cross entropy or log-loss is used for binary classification. The formula is given below:

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) - (1 - y_i) \cdot \log(1 - p(y_i))$$

where, N = number of samples, y = true label, $p(y)$ = predicted probability of label

Reading this formula, it explains that, for each true label point ($y = 1$), it adds $\log(p(y))$ to the loss, that is, the log probability of it being the true label. Conversely, it adds $\log(1 - p(y))$, that is, the log probability of it being the false label, for each point ($y=0$). The picture below shows an example of how the log-loss function behaves. As the predicted probability of the true class gets closer to zero, the loss increases exponentially[59]:



Categorical cross-entropy function is used for multi-class classification to output a probability for a specific number of classes greater than two. It measures the probability error for a classification tasks where the classes are mutually exclusive. It is also called softmax loss because it can be described as a softmax activation plus a cross-entropy loss[60].

$$CCE = - \sum_i^N g_i \sigma(\mathbf{x})_i (\log(y_i))$$

where g_i is the ground truth, y_i is the score for each class i in N and $\sigma(\mathbf{x})_i$ is the softmax function.

Optimization Algorithms

Optimization algorithms help the model to minimize the loss function in a neural network. An optimization function works by finding the weight and bias terms of a given data and the corresponding target values that are associated to each data point. It then approximates a new target value with a minimum error approximation [61]. One type of

optimization algorithm is backpropagation. The optimization algorithms implemented in this thesis work are the stochastic gradient descent (SGD) function and the adaptive moment estimation (Adam).

Backpropagation

Backpropagation is a mechanism to calculate the gradient of the loss function. It is important in the calculation of the weights involved in the network. Backpropagation is used to adjust the weights during the training of the model in order to minimize the error of the output [62].

Stochastic Gradient Descent

SGD is an iterative method for optimizing a differential objective function, a stochastic approximation of gradient descent optimization. SGD is famous for large scale optimization but has slow convergence asymptotically due to the inherent variance [63]. The equation of SGD is used to minimize an objective function is given in the form of a sum:

$$Q(w) = \frac{1}{n} \sum_{i=1}^n Q_i(w)$$

where, the parameter w that minimizes $Q(w)$ is to be estimated. Each function Q_i is associated with the i^{th} observation in the dataset.

Adaptive Moment Estimation Adaptive Moment Estimation

Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [64]. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods [65].

2.3.3 Deep Learning

Deep learning is a type of machine learning where algorithmic representation learning methods are used with multiple levels of representation. Non-linear modules are composed to transform the representation of an image at one level, to a higher and slightly abstract level. The layers in the algorithm manipulate the output from previous layers, the input layers. There are two main learning approaches; supervised learning approach (example, classification and regression) and unsupervised learning approach (example, clustering).

Supervised Learning In supervised learning, labelled datasets are used to train algorithms to predict outcomes or classify data. This work will use a form of supervised learning called classification. Details can be found in Chapters 4 and 5.

Unsupervised Learning In unsupervised learning, unlabelled datasets are used. The algorithm does not know the true labels of the input data. Thus, the model gets to learn by itself by modeling the probability density of input data. This learning approach is not used in this work.

2.3.4 Deep Neural Networks(DNNs)

A deep neural network is a NN with two or more hidden layers between the input and output layer. Many different architectures of DNNs has been used and created in literature [66][67], including transfer learning [68].

2.3.5 Convolutional Neural Network(CNN)

The organization of the animal visual cortex and the ability of humans to recognise different features of an object, inspired the development of convolutional neural networks. The visual cortex contains a vast number of cells responsible for identifying light in overlapping sub-regions of the visual field, the receptive fields. These cells behave as filters over the input; the more complex cells have larger receptive fields[69]. CNNs have the ability to learn hierarchical representation of raw input data without relying on handcrafted features. They are therefore widely used in image recognition, image classification, object detection, facial recognition etc. CNN consists of an input layer plus an output layer linked by a non-fixed number of hidden layers. Figure 2.12 shows an example of a typical CNN architecture.

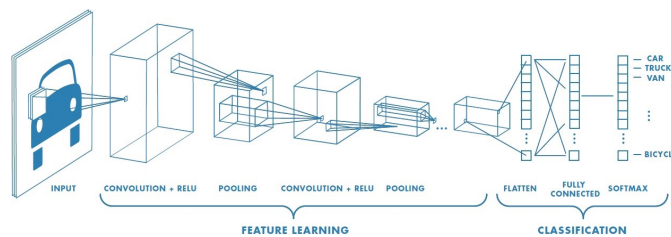


Figure 2.12: Example of a CNN architecture. Reprinted with permission from [3]

Convolutional Layer

Convolutional layer is the first layer to extract features from an input image. This layer is made of a set of small filters which has the ability to learn. The convolutional layer

operation is by convolving each filter over the entire input and computing a dot product between the input at any given position [3].

$$g(x, y) = \omega * f(x, y) = \sum_{s=-m}^m * \sum_{t=-n}^n \omega(s, t) f(x - s, y - t)$$

The above equation defines a convolution operation whereby $g(x, y)$ is the filtered image, $f(x, y)$ represents the original image and ω is the filter kernel. Each element of the filter kernel is examined within the ranges of $-m \leq s \leq m$ and $-n \leq t \leq n$

Pooling Layers

A pooling layer calculates the output for each element in a fixed-shape window of input data. The pooling layer decreases the resolution of the window to prevent misleading noise and distortion of pixels[70]. Figure 2.13 shows an example of how the pooling layer works. The three types of pooling layers are:

- Max pooling
- Average pooling
- Sum pooling

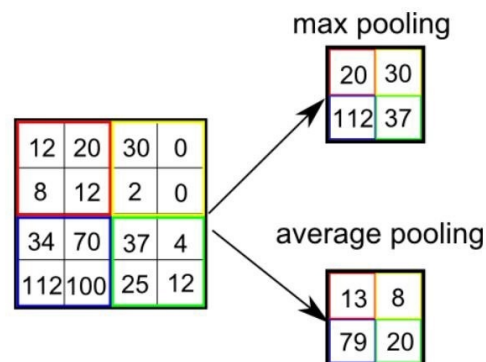


Figure 2.13: Illustration of Max Pooling and Average Pooling. The figure shows an example of max pooling operation and average pooling with a 2x2 pixel filter size from 4x4 pixel input. At max pooling, each filter is taken the maximum value, then arranged into a new output with a size of 2x2 pixels. While the average pooling value taken is the average value of the filter size. The figure is reprinted in unaltered form from: “Using convolutional neural networks for image recognition” [4]

Normalization Layer

The normalization layer calculates the mean and variance from the distribution of the summed input to a neuron, and then normalize this summed data to the next neuron each

time in a neural network. This process significantly reduces the training time of a neural network. This also enables each layer to learn a bit more independently as compared to other layers in the network[71]. The different types of normalization layers are: batch normalization(BN),weight normalization,layer normalization,group normalization and weight standardization. Batch normalization layer is used in Chapters 4 and 5. BN standardizes the inputs to any particular layer. This means that, the inputs to any layer in the neural network should have approximately zero mean and a unit variance[72].

Dropout Layer

A dropout layer periodically removes random nodes from the network, forcing the network to adjust to learn with different internal structures and become better equipped to handle any input. This process reduces overfitting in a NN by preventing complex co-adaptations on input data[73].

Permute Layer Permute layer changes the order or arrangement of the dimensions of the input data according to a given pattern[74].

Flatten Layer A Flatten layer reshapes the dimensions of the input data to have a shape that is equal to the number of elements contained in the input data. This layer removes all the dimensions except for one[75].

2.3.6 Recurrent Neural Network(RNN)

FFNNs propagate data forward and between connection units. To solve sequential data better, recurrent neural networks contain cyclic connections that make them a more powerful tool to model this data. RNNs propagate data forward and backwards from late processing stages to earlier stages. These kind of networks have an internal memory which helps them to process sequential data(Figure 2.14). Some examples of RNNs include; Long-Short Term Memory(LSTM)[76], Gated Recurrent Unit(GRU)[77] and the bidirectional forms of LSTM and GRU. RNNs have been used in handwriting recognition, speech recognition, time series forecasting etc[78][79]. In recent works by [11], convolutional recurrent neural networks(CRNN) have been used in polyphonic sound event detection which showed some promising results. Later in Chapter 4, a CRNN model is proposed for both binary and multiclass classification task in this thesis.

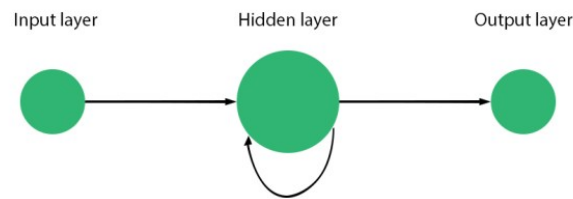


Figure 2.14: Illustration of a RNN structure.

2.3.7 Pretrained Models and Transfer Learning

A pretrained model is a saved neural network that was previously trained on a large dataset, typically on a large-scale image-classification task [68]. The pretrained model can be used as it is or use transfer learning to customize the model to fit a given tasks.

In computer vision, the lower layers of transfer learning models have been used as feature extractors such as detecting edges of a picture, while the final layers work toward task specific features [80]. Some popular models often used for transfer learning include Xception [81], VGG16 and 19 [82], Inception-v4, Inception-ResNet [83], Residual Networks (ResNet) [84] etc. Some of these models would be use in Chapter 5.

2.3.8 Hyperparameter Tuning

A hyperparameter is a criterion whose value is set before the learning process of a neural network begins. It refers to the task of testing different values for the hyperparameters, and finding the best. The process significantly helps in finding a configuration which produces good performance for a model, even though it can be computationally expensive. Some of the hyperparameters tuned in this thesis include: learning rate, batch size, input dimension, optimizers, loss function, filter size etc.

Learning rate is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function. *Batch size* refers to the number of training examples utilized in one iteration. *Input dimension* refers to the dimension of the input data. *Filter size* the dimensions of the filter kernel in the neural network.

2.3.9 Statistical Information and Metrics

Statistical information and analysis is an important part of this thesis. In order to train, validate and test the proposed models and choose a pre-trained model, various statistical information was used. Based on this information, a confusion matrix for both binary and multiclass classification of the present/absent labelling of the AED sound event drawn.

A confusion matrix is a table that indicates the number of false positives (FP), false negatives (FN), true positives (TP) and true negatives (TN) predictions of the classification model. On the basis of these indicators, the actual performance of the models can be evaluated. Figure 2.15 shows an example of a confusion matrix for binary classification.

Confusion Matrix			
		Actual Value	
		Yes (1)	No (0)
Predicted Value	Yes (1)	TP	FP
	No (0)	FN	TN

TP= True Positive
 FP= False Positive
 FN= False Negative
 TN= True Negative

Figure 2.15: Confusion Matrix

The statistical information on which these models were evaluated are:

Accuracy is a description of systematic errors, a measure of statistical bias; it is calculated as:

$$ACC = \frac{TP + TN}{TN + TP + FP + FN}$$

Recall measures the proportion of actual positives that are correctly identified, also called True Positive Rate (TPR). The equation to calculate it is:

$$TPR = \frac{TP}{TP + FN}$$

Precision is the fraction of relevant instances among all instances. It is also called Positive Precision Value (PPV) and it's defined as:

$$PPV = \frac{TP}{TP + FP}$$

F1 Score takes into consideration both the precision and the recall of the test to compute the final score; it is a measure of a test's accuracy. It is calculated as:

$$F1_Score = 2 \frac{PPV * TPR}{PPV + TPR}$$

ROC and AUC

Receiver Operating Characteristic(ROC) curve is graph that shows the performance of a classification model at all classification thresholds. The ROC plots the true positive rate(TPR) against the false positive rate(FPR). When the classification threshold is lower, the classifier classifies more items as positive, thus increasing both FP and TP.

$$FPR = \frac{FP}{TN + FP}$$

Area Under the Curve(AUC) measures the area underneath the whole ROC curve. It gives the an aggregate measure of performance across all possible classification thresholds. AUC provides the probability that a random positive item is classified as a random negative item. It ranges in value from 0 to 1. A model whose predictions are 100 percent wrong has an AUC value of 0 while an AUC value of 1 means that, all predictions are 100 percent correct. Figure 2.16 shows a diagram of ROC curve and AUC.

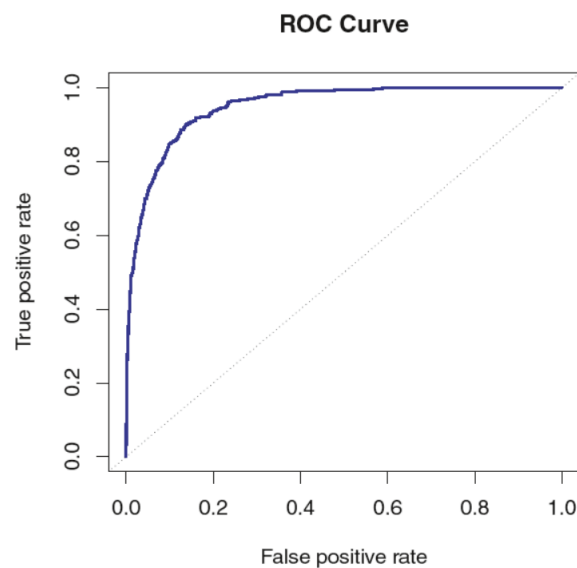


Figure 2.16: Illustration of ROC curve and AUC

3

Dataset

This chapter explains how the datasets used for the thesis work was created and pre-processed. This is an essential part of the thesis and it would further help to understand the feature extraction processes used to solve the proposed problem. Figure(3.1) illustrates the section of the thesis which is approached in the rest of the chapter.

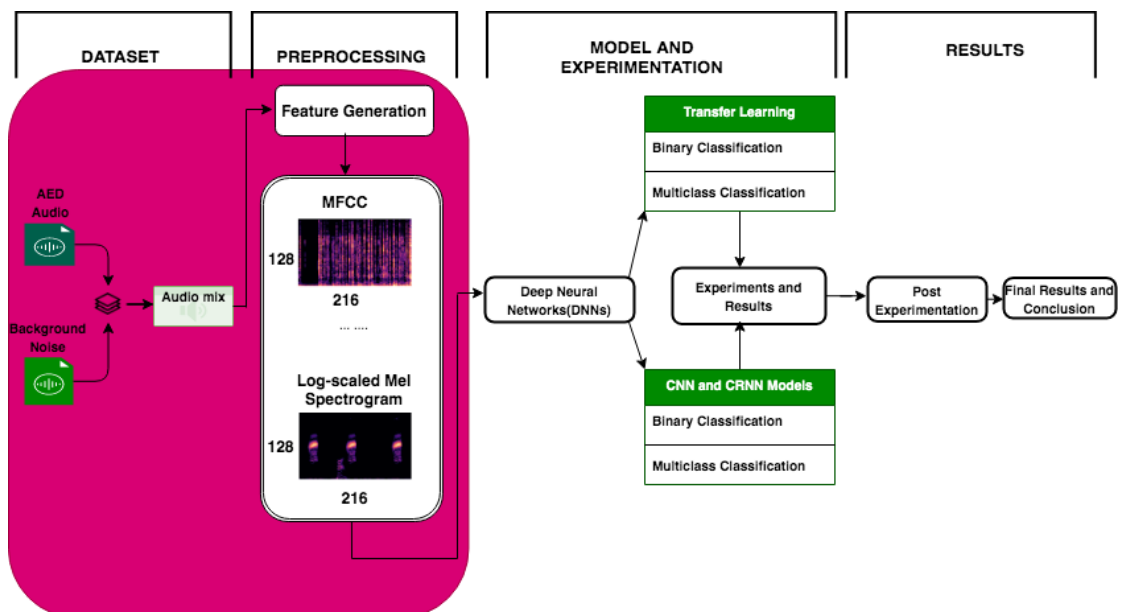


Figure 3.1: An illustration of the approach used for the thesis

3.1 Overview of Dataset

The audio dataset used in this thesis is in two parts: monophonic AED audio data and a mix of monophonic and polyphonic background audio data. The monophonic AED audio data is made of 10 labelled, 1.5 to 2 [s] audio files. Each of the 10 files contain a unique audio event/command from the AED device.

In order to make a final dataset that mimics a real recorded polyphonic audio in a team training environment, audio data from already available datasets [85] and polyphonic audio on the internet [86] is collected to form background audio/noise data. A description of these datasets is presented in the next subsections.

3.1.1 AED Audio Dataset

The table(3.1) below shows properties of the AED audio event samples used for further preprocessing and experimentation. Some of the common properties are:

- WAV audio file
- Monophonic(1 audio channel)
- Sample rate of 44100 [Hz]
- 16 bits per sample

Table 3.1: AED Audio Event Properties

Audio Event	Label	Duration[seconds]	Loudness[dBFS]	Size[KB]
Apply Pads	P03	1.30	-23.1	250
Analysing Heart Rhythm	P08	1.43	-22.6	127
Do Not Touch the Patient	P09	1.75	-24.0	156
Analysing Interrupted	P10	1.58	-22.9	143
No Shock Advised	P13	1.55	-23.3	139
Shock Advised	P17	1.30	-23.3	119
Charging	P18	0.65	-22.6	415
Stay Clear of Patient	P19	1.79	-22.8	160
Shock Delivered	P23	1.16	-23.2	106
No Shock Delivered	P25	1.30	-23.1	250

3.1.2 Background Noise Dataset

The background noise dataset is made of 7685 audio clips, each of length 5[s] and varying [dBFS]. The first 2000 audio clips are sourced from the popular ESC-50 dataset [85], 3000 audio clips from polyphonic noise in a drinking bar setting and 2865 audio clips from intensive care unit(ICU) audio recording. The last two variant are sourced from YouTube [86].

The ESC-50 dataset is a collection of short environmental recordings available in a unified format (5-second-long clips, 44.1 [kHz], single channel, compressed at 192 [kbit per s]). The dataset is organized into 50 semantical classes (with 40 examples per class) loosely arranged into 5 major categories. These categories are: animals, natural soundscapes water sounds, human, non-speech sounds, interior or domestic sounds and exterior or urban noises.

The dataset from the bar and ICU setting are 8 hours and 1 hour long respectively. The original audios are divided into 5-second-long clips using the python library PyDub [87]. These samples are recorded at 44.1 [kHz], single channel and compressed at 164 [kbit per s]. Table (3.2) below shows a a distribution of the samples that make up the background dataset.

Table 3.2: Distribution of background noise dataset

Source of Audio	Number of Audio Clips
ESC-50	2000
ICU	2685
Bar	3000
Total	7685

3.1.3 Selection of AED audio Classes

Five AED audio labels were randomly chosen for further studies. The five AED labels used for the classification tasks are:

- Shock advised
- No shock advised
- Apply pads
- Shock delivered

- Do not touch the patient

Five other AED audio labels are used to create a 'backgrounds noise folder'. This backgrounds noise folder is categorised as a label on it's own. The aim is to provide a 'difficult' label to compare robustness of the models in the binary and multiclass classification tasks. These labels are:

- Analysing heart rhythm
- Stay clear of patient
- No shock delivered
- Charging
- Analysing interrupted

3.1.4 Polyphonic Audio Mixing

Mixtures were created by randomly overlaying AED audio event instances on a background audio segment of length 1-4 seconds. In this way, the whole length of the AED audio clip is present on the 5 second background audio clip. Both AED audio clips and the background noise are sampled at the same sample rate of 44.1 [kHz]. This is done with PyDub[87] and the code for creating these mixtures can be found in Appendix A.3. The following are three different methods and the reasons to why these datasets were created:

1. Firstly, the AED audio samples are overlaid on the background noise clips at their original volume in dBFS(Appendix A.3), assuming that, the volume recorded is the highest for any recording device in a team training setting. This creates a polyphonic audio mixture of 7685 audio clips for each class of AED audio event. This dataset is referred to as dataset 1.
2. Secondly, the volume in dBFS of the AED audio samples is reduced by 10 [dBFS](Appendix A.2) and then overlaid on the background noise clips. This is to create a scenario where by the background noise is louder than the AED audio. In team training scenario, the movement of the recording device from the AED device would reduce the volume at which the AED command will be recorded. Also, the varying volume of the background environments makes this close to a realistic team training case scenario. This dataset is referred to as dataset 2.

3. Finally, to determine how good the model would work in detecting lower AED audio in varying noisy background, the volume of the AED audio clips is reduced by 15 [dBFS](Appendix A.2) and then overlaid on the background noise clips. In some cases when the background noise is very high, it is almost impossible for a human to detect the AED audio event in the samples created. This dataset is referred to as dataset 3.

The above three stages, forms the basis for creating the final datasets that are used in the following Chapters(4 and 5).

3.1.4.1 Creation of Final Dataset

Two datasets are created for work in this thesis. They are referred to as **Reduced10db** and **Reduced15db**.

Reduced10db is a combination of datasets 1 and 2 (Table 3.3). It has 15370 samples per AED audio class of polyphonic audio mixture. In essence, the audio clips in this dataset has AED audio with their original volume and a reduced volume of 10 [dBFS].

Background10db 'class' under Reduced10db dataset. It contains 15370 audio clips. 7685 audio clips are the original background noise and another 7685 audio clips are from the 'five other' AED audio classes(1537 samples per class). The 1537 samples are randomly selected from each of the five classes using the shutil module in Python [88].

Table 3.3: Reduced10db outlook

Label	Number of Sample Images
P03	15370
P09	15370
P13	15370
P17	15370
P23	15370
Background10db	15370(7685 background noise + 7685 of P08,P10,P18,P19 and P25)

Reduced15db is a combination of datasets 1,2 and 3 (Table 3.4). This dataset has 23050 audio clips per AED audio class of polyphonic audio mixture. In simple terms, the audio clips in this dataset has AED audio with their original volume, a reduced volume of 10 [dBFS] and even a lower reduced volume of 15 [dBFS].

Background15db 'class' under Reduced15db dataset. It contains 23050 audio clips. 7685 audio clips are the original background noise while 15365 audio clips are from the

'five other' AED audio classes(3073 audio clips per class). The 3073 audio clips are randomly selected from each of the 'five other' classes.

Table 3.4: Reduced15db outlook

Label	Number of Sample Images
P03	23050
P09	23050
P13	23050
P17	23050
P23	23050
Background15db	23050(7685 background noise + 15365 of P08,P10,P18,P19 and P25)

3.2 Feature Extraction

Numerous feature extraction methods have been used in polyphonic sound event detection studies. The most used features are the log-scaled mel spectrogram and MFCC[11][89][90]. To determine the best features for this dataset, both MFCC and log-scaled mel spectrograms have been generated and tested(Code is in Appendix A.4). The python library used in this section is called Librosa [91].

3.2.1 MFCC

The MFCC features are mostly used in speech recognition tasks. MFCCs generated are as a results of the following process:

1. Sample the audio datasets at a fixed sample rate of 22050 [kHz]. A small sample rate for a 16 bitrate audio gives better results [92].
2. Take the STFT of the audio signal and divide it according to the mel-scale. The mel-scale has a fixed size of 266 [Hz] frequency bins.
3. Perform DCT on the frequency bins and obtain a resultant coefficient of 40.

Thus, 40 MFCCs are calculated for each time window with 50 percent overlap. The same parameters are used to generate a 2-dimensional spectrogram of size 40*218. Samples of some of the images generated for both Reduced10db and Reduced15db datasets are shown in Appendix C. Figure (3.2) shows three examples of MFCC features generated

from Reduced10db. The python file `create_mfcc.py` in Appendix A.4 generates these spectrograms.

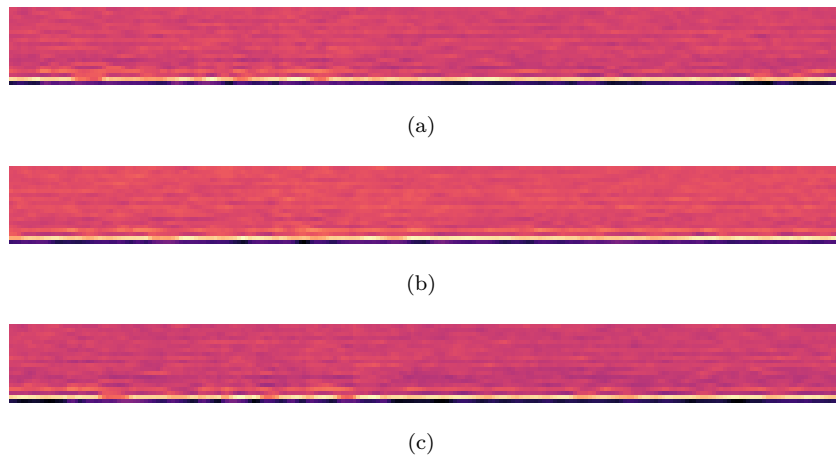


Figure 3.2: MFCC examples

3.2.2 Log-scaled Mel Spectrogram

The log-scaled features are generated with using the following process [91]:

1. Sample the audio datasets at a fixed sample rate of 22050 [kHz]. A small sample rate for a 16 bitrate audio gives better results [92].
2. Perform windowing with a hop length of 512.
3. Take the STFT of the audio signal and divide it according to the Mel-scale. The Mel-scale has a fixed size of 266 [Hz] frequency bins.
4. Calculate the number of FFT for the audio samples. The number of FFT is fixed at 2048.
5. Generate number of mel bands. In this case 128 mel bands.
6. Take the log of the generated spectrogram.

The resultant images are 2-dimensional of sizes 128*218. The python file `create_logscaledmelspec.py` in Appendix A.4 generates these spectrograms. Figure (3.3) shows some examples of the images generated. More examples are attached in Appendix C.

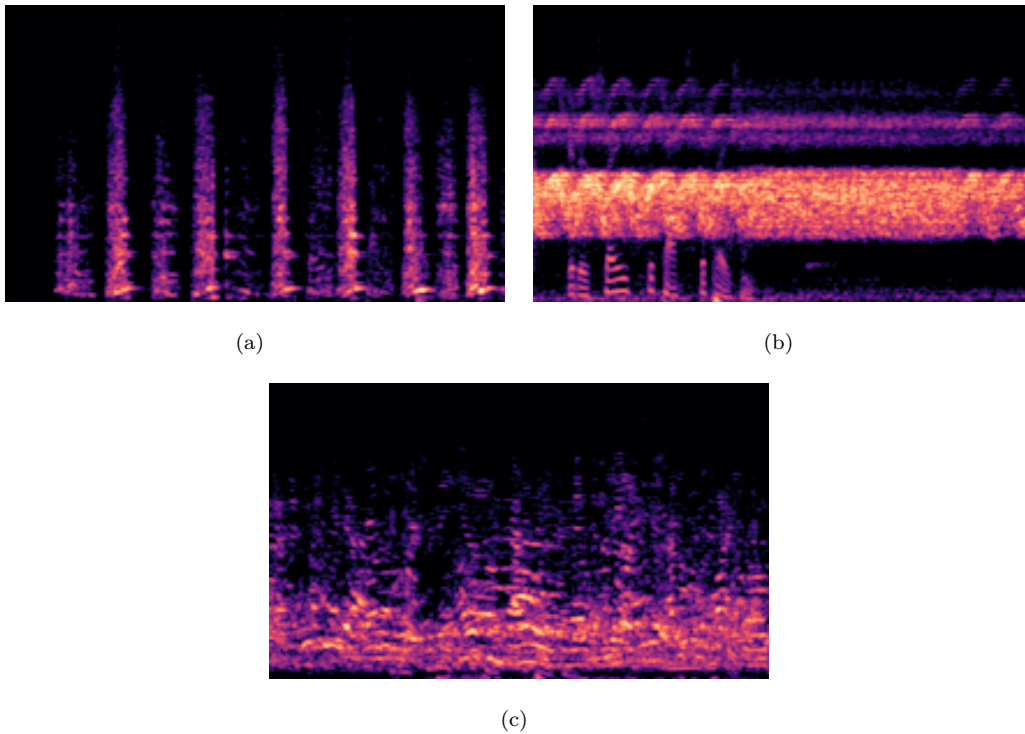


Figure 3.3: Log-scaled Mel Spectrogram examples

3.3 Train, Validation and Testing Set

The final datasets are randomly selected and separated into train, test and validation sets. This is to further make sure that, the models developed are robust enough. 70 percent of the data is used for training the model, 20 percent is used to validate the model and 10 percent is used to test the model. Below(Figure 3.4) is an illustration of how the folders are arranged and stored for further work.

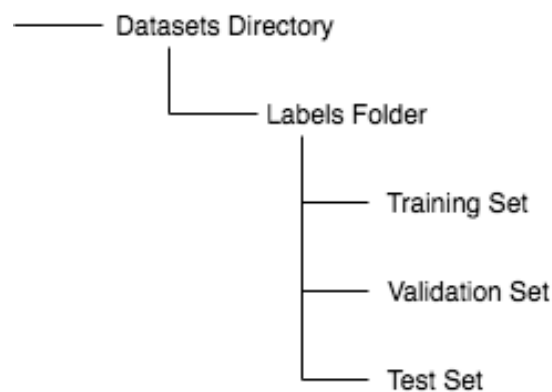


Figure 3.4: Illustration of how the folders are arranged. For each labels folder, the training, validation and tests sets are stored in separate sub-folders.

3.4 Contribution to Dataset and Preprocessing

The main contribution to the dataset is the collection of background noise and creation of polyphonic audio mixtures that mimic team training environments. In essence, the final datasets generated for further studies in this thesis can be regenerated using the codes in Appendix A and if one has access to the AED audio data samples. Furthermore, the preprocessing steps and feature extraction for this work can be done using the python codes attached in Appendix A.

4

Methodology

This chapter elaborates on a proposed system, methods and architectures used for the binary and multiclass classification task in this thesis. Furthermore, the different methodologies used in the classification of the AED audio clips explained. Figure(4.1) illustrates the section of the thesis which is approached in this chapter.

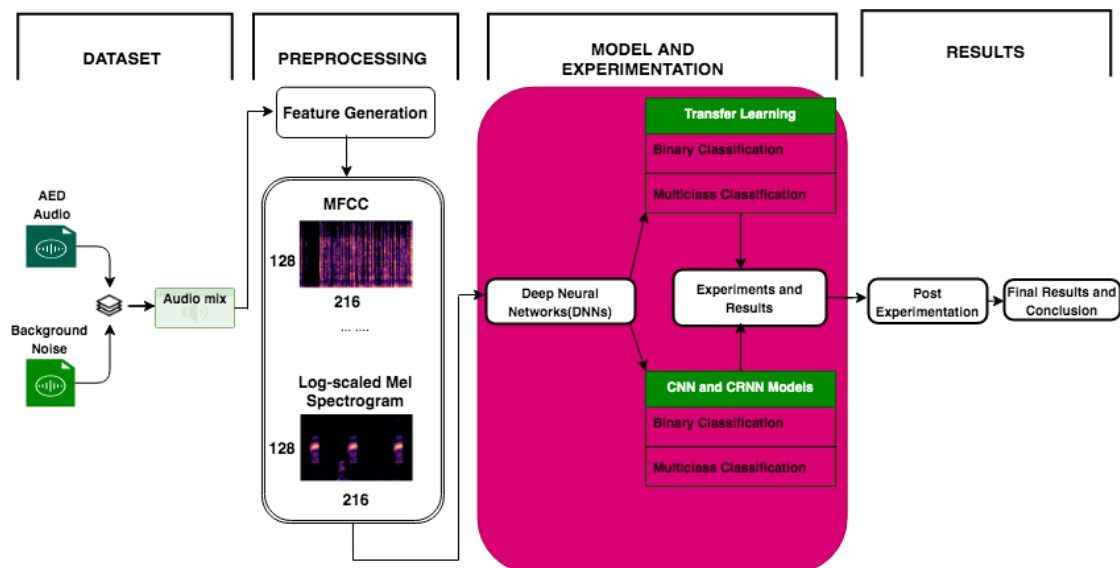


Figure 4.1: An illustration of the approach used for the thesis

4.1 Introduction

This section explores an initial proposal for a final system to detect sound events in team training scenarios, and the DNN architectures used for classifying the presence or absence of a AED audio clip in a background noise. Two CNN models and an CRNN model are presented as the proposed models for both binary and multiclass classification. The main difference in both architecture is detailed in the next sections.

Transfer learning model architectures are also used for both classification scenarios. The models with the best statistical information considered in this thesis are also presented. This helps to have an overview of methods that could be exploited to solve the polyphonic sound event detection problem. Additionally, visual structures of these architectures are presented to show the differences in their outlook and to help the reader understand the various models used in Chapter 5 for experimentation.

4.1.1 Initial Proposed System

A proposal for a system to detect sound events in team training events is shown below (Figure 4.2). The idea behind this system is that, sound is recorded with a device in a team training initially. This recorded sound is divided into chunks as a function of time and the audio chunks are preprocessed into spectrograms. The spectrograms could be in the form of MFCC or log-scaled mel spectrograms using the procedures described in Chapter 3. These spectrograms will form the input data into the DNN model for either binary or multiclass classification.

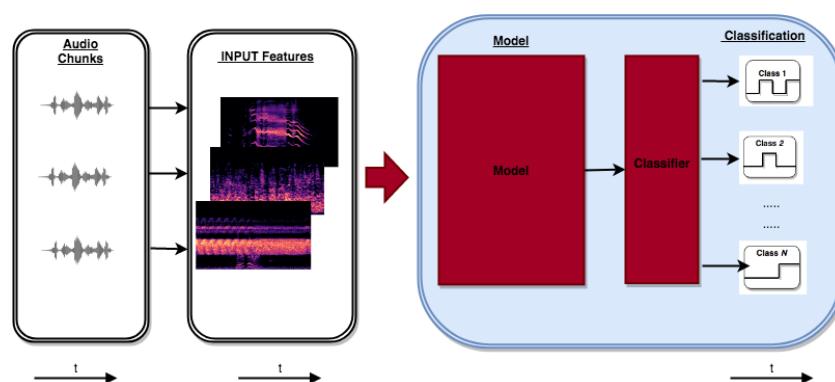


Figure 4.2: Initial system proposed for AED sound event detection in team training scenario

4.1.2 Model Baseline Approach

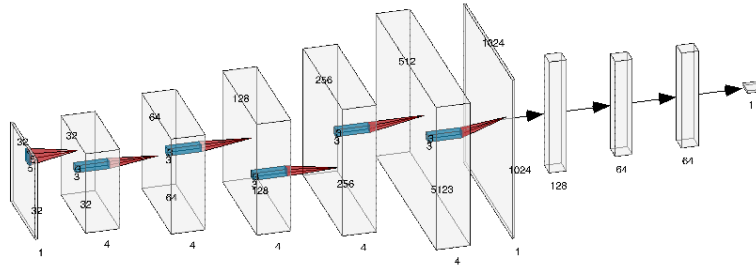
There have been studies to detect the presence or absence of sound in our everyday life events. These authors explored various methods for solving problems of this kind [93–97]. However, the approaches in this work is similar to [98], where they successful used a transfer learning for COVID-19 detection in the form of binary classification. All previous studies used either MFCC or log-scaled mel spectrograms as input images. The proposed architectures in this work are built from scratch.

4.2 Proposed Architectures

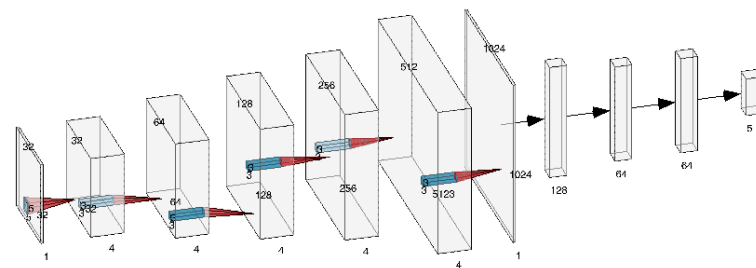
The proposed architectures explore the idea of using CNN and CRNN to perform both binary and multiclass classification tasks. For the binary classification, the loss function implemented is the binary cross-entropy while the categorical cross-entropy was used for the multiclass classification. Under each classification task, the models are each run on the same number of epochs. The optimizers used include: Adam,SGD and Nesterov Momentum into Adam(Nadam).

4.2.1 Architecture A

Architecture A is a CNN model made up of 21 convolutional layers(Figure 4.3). The first convolutional layer is the input layer and the last convolutional layer is the output layer. The input layer has 32 neurons and a filter kernel size of 7*7. This layer takes in a 2-dimensional(2D) image of size 224*224 and gets convoluted with a ReLU activation function, batch normalization and a 2D maximum pooling operation. Architecture A is constructed in a way that, there is a batch normalization and maximum pooling operation after every two convolutional layers in the the hidden layer section of the model. This draws similarity with ResNet18 architecture [99].



(a) Binary classification model



(b) Multiclass classification model

Figure 4.3: AlexNet style of CNN model architecture A

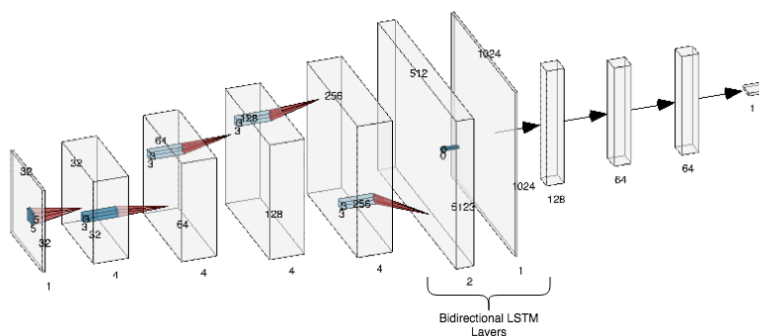
The output of the hidden layers is operated with a 2D global average pooling before it is flattened and fully connected on the first dense layer. Two dropout layers are used to avoid overfitting the model before the prediction layer. For the binary prediction(4.3a), a sigmoid activation function is used to while a softmax activation is used for the multiclass classification(4.3b). In all, the total parameters in this model is 13,460,545, of which the trainable parameters are 13,455,553 while the non-trainable parameters are 4,992. Details of architecture A can be found in Table 4.1. The code for architecture A can be found in Appendix A.6.

Table 4.1: Summary of layers of architecture A

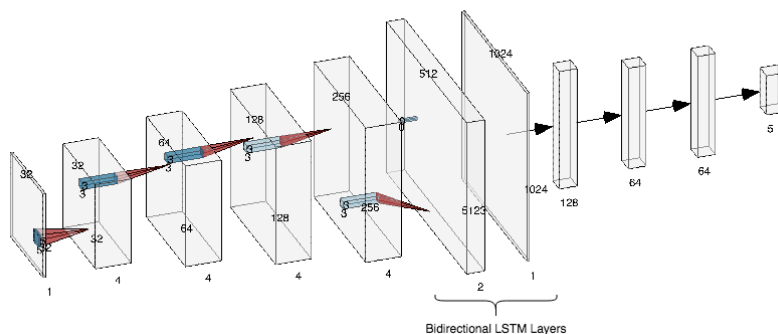
Layer (type)	Output Shape	Param
conv2d (Conv2D)	(None, 119, 119, 32)	4736
max_pooling2d (MaxPooling2D)	(None, 59, 59, 32)	0
conv2d_1 (Conv2D)	(None, 30, 30, 32)	9248
conv2d_2 (Conv2D)	(None, 14, 14, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 32)	0
batch_normalization (BatchNorm)	(None, 5, 5, 32)	128
conv2d_3 (Conv2D)	(None, 2, 2, 32)	9248
conv2d_4 (Conv2D)	(None, 1, 1, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 32)	0
batch_normalization_1 (BatchNorm)	(None, 1, 1, 32)	128
conv2d_5 (Conv2D)	(None, 1, 1, 64)	18496
conv2d_6 (Conv2D)	(None, 1, 1, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 64)	0
batch_normalization_2 (BatchNorm)	(None, 1, 1, 64)	256
conv2d_7 (Conv2D)	(None, 1, 1, 64)	36928
conv2d_8 (Conv2D)	(None, 1, 1, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 64)	0
batch_normalization_3 (BatchNorm)	(None, 1, 1, 64)	256
conv2d_9 (Conv2D)	(None, 1, 1, 128)	73856
conv2d_10 (Conv2D)	(None, 1, 1, 128)	147584
max_pooling2d_5 (MaxPooling2D)	(None, 1, 1, 128)	0
batch_normalization_4 (BatchNorm)	(None, 1, 1, 128)	512
conv2d_11 (Conv2D)	(None, 1, 1, 128)	147584
conv2d_12 (Conv2D)	(None, 1, 1, 128)	147584
max_pooling2d_6 (MaxPooling2D)	(None, 1, 1, 128)	0
batch_normalization_5 (BatchNorm)	(None, 1, 1, 128)	512
conv2d_13 (Conv2D)	(None, 1, 1, 256)	295168
conv2d_14 (Conv2D)	(None, 1, 1, 256)	590080
max_pooling2d_7 (MaxPooling2D)	(None, 1, 1, 256)	0
batch_normalization_6 (BatchNorm)	(None, 1, 1, 256)	1024
conv2d_15 (Conv2D)	(None, 1, 1, 256)	590080
conv2d_16 (Conv2D)	(None, 1, 1, 256)	590080
max_pooling2d_8 (MaxPooling2D)	(None, 1, 1, 256)	0
batch_normalization_7 (BatchNorm)	(None, 1, 1, 256)	1024
conv2d_17 (Conv2D)	(None, 1, 1, 512)	1180160
conv2d_18 (Conv2D)	(None, 1, 1, 512)	2359808
max_pooling2d_9 (MaxPooling2D)	(None, 1, 1, 512)	0
batch_normalization_8 (BatchNorm)	(None, 1, 1, 512)	2048
conv2d_19 (Conv2D)	(None, 1, 1, 512)	2359808
conv2d_20 (Conv2D)	(None, 1, 1, 512)	2359808
max_pooling2d_10 (MaxPooling2D)	(None, 1, 1, 512)	0
batch_normalization_9 (BatchNorm)	(None, 1, 1, 512)	2048
conv2d_21 (Conv2D)	(None, 1, 1, 512)	2359808
max_pooling2d_11 (MaxPooling2D)	(None, 1, 1, 512)	
batch_normalization_10 (BatchNorm)	(None, 1, 1, 512)	2048
global_average_pooling2d	(None, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 128)	65664
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 1)	65

4.2.2 Architecture B

Architecture B is a CRNN model made up of 16 convolutional layers (Figure 4.4). The main difference between architecture A and B is the introduction of 3 bidirectional LSTM layers. After the first 16 layers, the model parameters are reshaped from 2D to 1D through the reshape layer. The parameters are then permuted and fed into the bidirectional LSTM layers.



(a) Binary classification model



(b) Multiclass classification model

Figure 4.4: AlexNet style of CRNN model architecture B

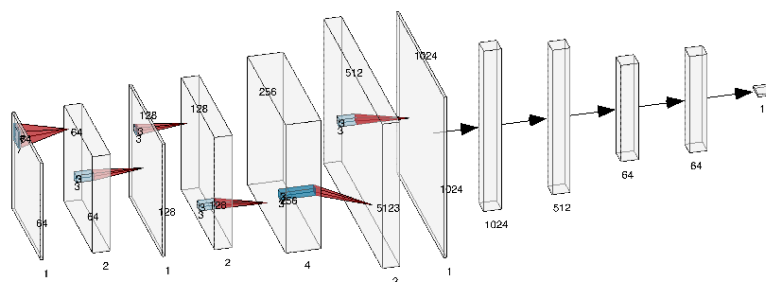
The output of the CRNN hidden layers is normalized before it is flattened and fully connected to the first dense layer. Two dropout layers are used to avoid overfitting the model before the prediction layer. For the binary prediction (Figure 4.4a), a sigmoid activation function is used while a softmax activation is used for the multi-class classification (Figure 4.4b). In all, architecture B has a total parameters of 9,232,961 of which the trainable parameters are 9,230,529, while the non-trainable parameters are 2,432. Details of architecture A can be found in Table 4.2. The code for architecture B can be found in Appendix A.6.

Table 4.2: Summary of layers of architecture B

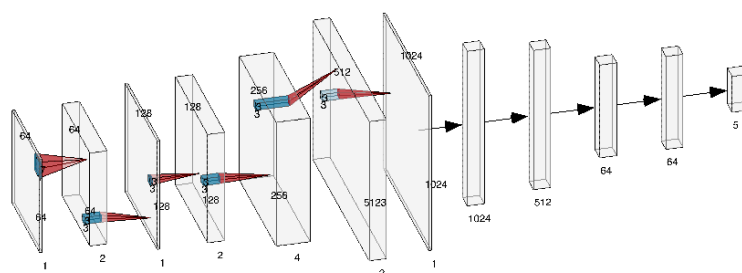
Layer (type)	Output Shape	Param
conv2d (Conv2D)	(None, 119, 119, 32)	4736
max_pooling2d (MaxPooling2D)	(None, 59, 59, 32)	0
conv2d_1 (Conv2D)	(None, 30, 30, 32)	9248
conv2d_2 (Conv2D)	(None, 14, 14, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 32)	0
batch_normalization (BatchNorm)	(None, 5, 5, 32)	128
conv2d_3 (Conv2D)	(None, 2, 2, 32)	9248
conv2d_4 (Conv2D)	(None, 1, 1, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 32)	0
batch_normalization_1 (BatchNorm)	(None, 1, 1, 32)	128
conv2d_5 (Conv2D)	(None, 1, 1, 64)	18496
conv2d_6 (Conv2D)	(None, 1, 1, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 64)	0
batch_normalization_2 (BatchNorm)	(None, 1, 1, 64)	256
conv2d_7 (Conv2D)	(None, 1, 1, 64)	36928
conv2d_8 (Conv2D)	(None, 1, 1, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 64)	0
batch_normalization_3 (BatchNorm)	(None, 1, 1, 64)	256
conv2d_9 (Conv2D)	(None, 1, 1, 128)	73856
conv2d_10 (Conv2D)	(None, 1, 1, 128)	147584
max_pooling2d_5 (MaxPooling2D)	(None, 1, 1, 128)	0
batch_normalization_4 (BatchNorm)	(None, 1, 1, 128)	512
conv2d_11 (Conv2D)	(None, 1, 1, 128)	147584
conv2d_12 (Conv2D)	(None, 1, 1, 128)	147584
max_pooling2d_6 (MaxPooling2D)	(None, 1, 1, 128)	0
batch_normalization_5 (BatchNorm)	(None, 1, 1, 128)	512
conv2d_13 (Conv2D)	(None, 1, 1, 256)	295168
conv2d_14 (Conv2D)	(None, 1, 1, 256)	590080
max_pooling2d_7 (MaxPooling2D)	(None, 1, 1, 256)	0
batch_normalization_6 (BatchNorm)	(None, 1, 1, 256)	1024
conv2d_15 (Conv2D)	(None, 1, 1, 256)	590080
conv2d_16 (Conv2D)	(None, 1, 1, 256)	590080
max_pooling2d_8 (MaxPooling2D)	(None, 1, 1, 256)	0
batch_normalization_7 (BatchNorm)	(None, 1, 1, 256)	1024
reshape (Reshape)	(None, 256, None)	0
permute (Permute)	(None, None, 256)	0
bidirectional (Bidirectional)	(None, None, 1024)	3149824
bidirectional_1 (Bidirectional)	(None, None, 512)	2623488
bidirectional_2 (Bidirectional)	(None, 256)	656384
batch_normalization_8 (BatchNorm)	(None, 256)	1024
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 128)	32896
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 1)	65

4.2.3 Architecture C

Architecture C is a CNN model of 6 convolutional layers (Figure 4.5). The input layer has 32 neurons and a filter kernel size of 7×7 . This layer takes in a 2-dimensional (2D) image of size 224×224 and gets convoluted with a ReLU activation function, batch normalization and a 2D maximum pooling operation. The first two hidden layers have 64 and 128 neurons respectively and a filter kernel size of 3×3 . Both layers are normalized with a batch normalization layer before they are inputted into the third, fourth and fifth convolutional layers. The output of these layers are further normalized and inputted into the output convolutional layer.



(a) Binary classification model



(b) Multiclass classification model

Figure 4.5: AlexNet style of CNN model architecture C

The fully connected part of this model is similar to architecture A. The main difference is that, it has one more dense and dropout layer. For the binary prediction (4.5a), a sigmoid activation function is used while a softmax activation is used for the multi-class

classification(4.5b). In all, the total parameters in this model is 8,524,358 of which the trainable parameters are 8,521,030 while the non-trainable parameters are 3,328. Details of architecture C can be found in Table 4.3. The code for architecture C can be found in Appendix A.6.

Table 4.3: Summary of layers of architecture C

Layer (type)	Output Shape	Param
conv2d (Conv2D)	(None, 119, 119, 32)	4736
max_pooling2d (MaxPooling2D)	(None, 59, 59, 32)	0
conv2d_1 (Conv2D)	(None, 30, 30, 64)	18496
conv2d_2 (Conv2D)	(None, 14, 14, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 128)	0
batch_normalization (BatchNorm)	(None, 5, 5, 128)	512
conv2d_3 (Conv2D)	(None, 2, 2, 128)	147584
conv2d_4 (Conv2D)	(None, 1, 1, 512)	590336
conv2d_5 (Conv2D)	(None, 1, 1, 512)	2359808
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 512)	0
batch_normalization_1 (BatchNorm)	(None, 1, 1, 512)	2048
conv2d_6 (Conv2D)	(None, 1, 1, 1024)	4719616
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 1024)	0
batch_normalization_2 (BatchNorm)	(None, 1, 1, 1024)	4096
global_average_pooling2d (GloAvgPool2D)	(None, 1024)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 512)	524800
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 128)	65664
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	4160
dropout_2 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 1)	65

4.3 Transfer Learning Models

Pretrained models are trained on large and more general datasets. Therefore, they serve as generic and robust models that have the potential to classify our data effectively. Since our datasets are different from the original datasets (ImageNet [100]) on which these pretrained models were used, the pretrained models are fine-tuned by freezing the top layers and using their last layers in addition to a newly-added classifier. This allows effective and faster training while minimizing overfitting. It also helps in training these models to learn the most relevant features for the specific classification tasks we have. The models studied during this work include:

- Xception [81]
- VGG16 and VGG19 [101]

- ResNet50 and ResNet18 [99]
- NASNetMobile [102]

Based on the training and validation results of the above models, ResNet18 showed the best results for our datasets and the architecture is presented in the following subsections.

4.3.1 ResNet18

ResNet18 is a smaller(18 layer) residual network. It is computationally less expensive and trains on less amount of parameters as compared to ResNet50. ImageNet weights are used for the transfer learning process while using this model. It is fine-tuned to fit data. Figure 4.6 shows a summary of the model architecture.

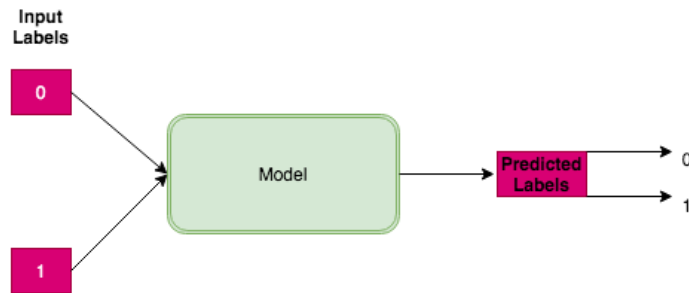
Layer Name	ResNet18
conv1	(7x7,64), stride 2
Pooling Operation	(3x3), maximum pooling 2D
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
Pooling Operation	global average pooling 2D

Figure 4.6: Summary of ResNet18 architecture

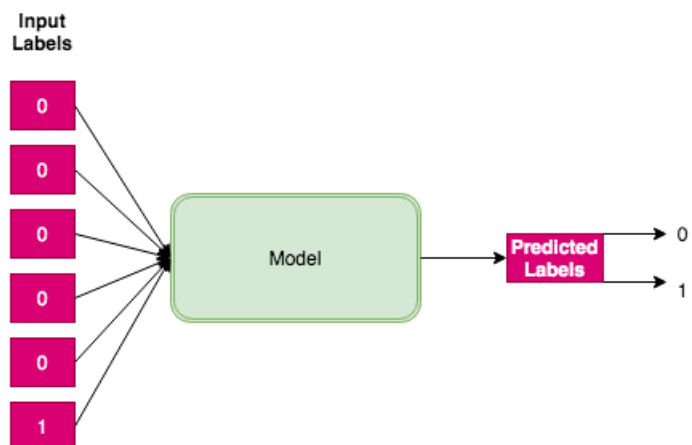
4.4 Binary Classification Methods

Classifying the presence or absence of AED clips which have similar commands could be difficult for humans at a lower volume. Bearing this in mind, we performed two binary

classification methods. The first method involves one-versus-one(1-versus-1) binary classification in which two classes serve as the input data. A comparison is made to determine if the model will be able to correctly classify two similar AED audio clips in a noisy environment. In this method, the AED command of interest is labelled '1', while the other is labelled '0'. For example, the commands 'no shock advised' and 'shock advised' are very similar. The same can be said of 'shock delivered' and 'no shock delivered'. Below (Figure 4.7a) is a diagram to illustrate the 1-versus-1 binary classification.



(a) Illustration of 1-versus-1 binary classification method



(b) Illustration of 1-versus-others binary classification method

Figure 4.7: Illustration of binary classification methods

The second binary classification method in this work is referred to as one-versus-others(1-versus-others) method. In this method, the AED command of interest '1' is labelled '1' while the rest of the AED commands are labelled '0'. The data is then inputted into the models mentioned above and then classified appropriately. The diagram below(Figure 4.7b) shows how the 1-versus-others binary classification method is implemented. This is the main method used for the major part of binary classification tasks in this work.

4.5 Multiclass Classification Methods

The multiclass classification method uses the classic 1-hot-encoding whereby the input datasets are labelled from 0,1 to N . N is the highest number of classes that are to be determined. In experimenting with the models in Chapter 5, $N = 4$ for multiclass classification of only the AED audio classes(5 classes) and $N = 5$ for multiclass classification of AED audio classes including the background noise class(6 classes). Figure 4.8 illustrates the multiclass classification method used for $N = 5$ labels.

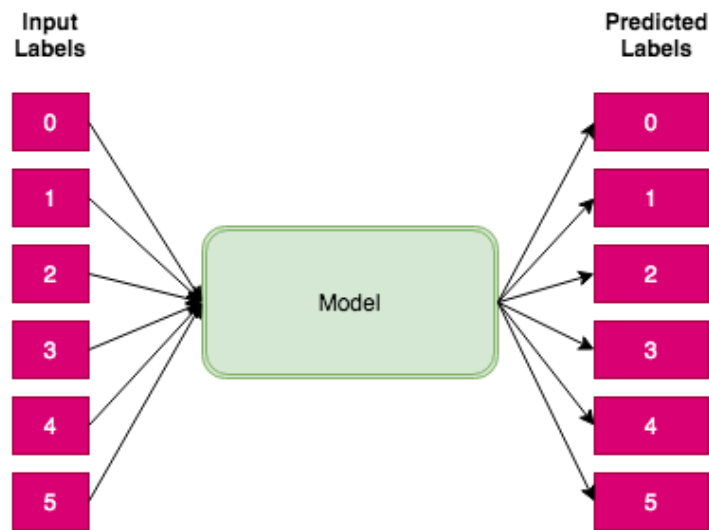


Figure 4.8: Illustration of multiclass classification method using 6 classes

5

Experiments and Results

This chapter presents explanation to the experiments carried out in this thesis. The results of the experiments are presented to demonstrate the performance of the models. Figure(5.1) illustrates the section of the thesis which is approached in the rest of the chapter.

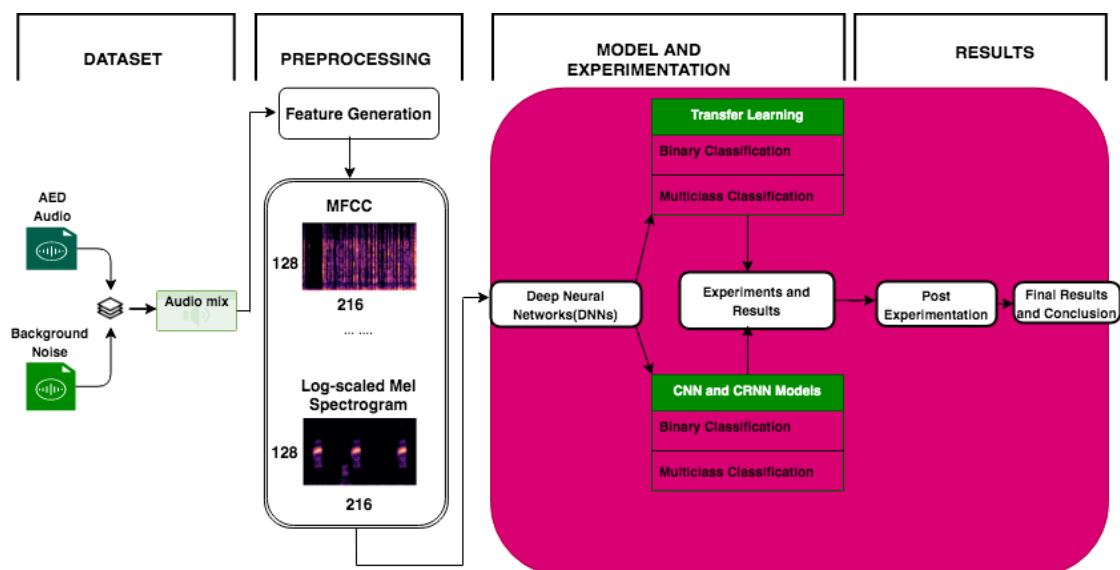


Figure 5.1: An illustration of the approach used for the thesis

5.1 Introduction

Various experiments have been carried out to determine the best performing models in this work. This chapter gives a detailed explanation of the procedures explored while using the models and methods elaborated in Chapter 4 to come to the final results and conclusion. The later sections of this chapter also presents some experimental results, statistical analysis for both binary and multiclass classification. The experimentation process follows the same procedure for all the architectures and transfer learning models described in the previous chapter. Figure 5.2 shows an illustration of the steps involved.

5.1.1 Experimental Setup

The flow of the experiments starts with the initialization of the image shapes. Architectures A to C are experimented with the original 'image' sizes generated through the feature extraction. Also varying image heights and width are used as input shapes for experimentation purposes to find the optimal input image shape. The best image size was 224*224 for both MFCC and log-scaled mel spectrogram images. Then preceded with loading the datasets and normalizing them. Since the images are in 'RGB' form, each image is divided by 255. The normalized images are then resized and labelled(from 0 to N) according to which class they belong.

An optimizer is then chosen for each experiment followed by hyperparameter tuning. The model is then trained on a fixed number of epochs. The resultant model is saved and loaded to evaluate it with the validation dataset and the results of the validation loss, validation accuracy, training loss and training accuracy are compared to verify model performance(overfitting and underfitting). In case the model is overfitting, underfitting or of low accuracy, new hyperparameters are used until the best results are achieved for each experiment. A final evaluation of the best model with the test dataset and the calculation of a confusion matrix and performance matrix is performed. Figure 5.2 shows a flow diagram of the experimental setup.

The setups for the experiments are divided into three. Experiment 1 will stand for binary classification using the 1-versus-others method. Experiment 2 is performed to compare the the how two similar classes perform in the models in a binary classification setting. The 1-versus-1 method is used here. The final experiment, experiment 3, is an experiment for multiclass classification.

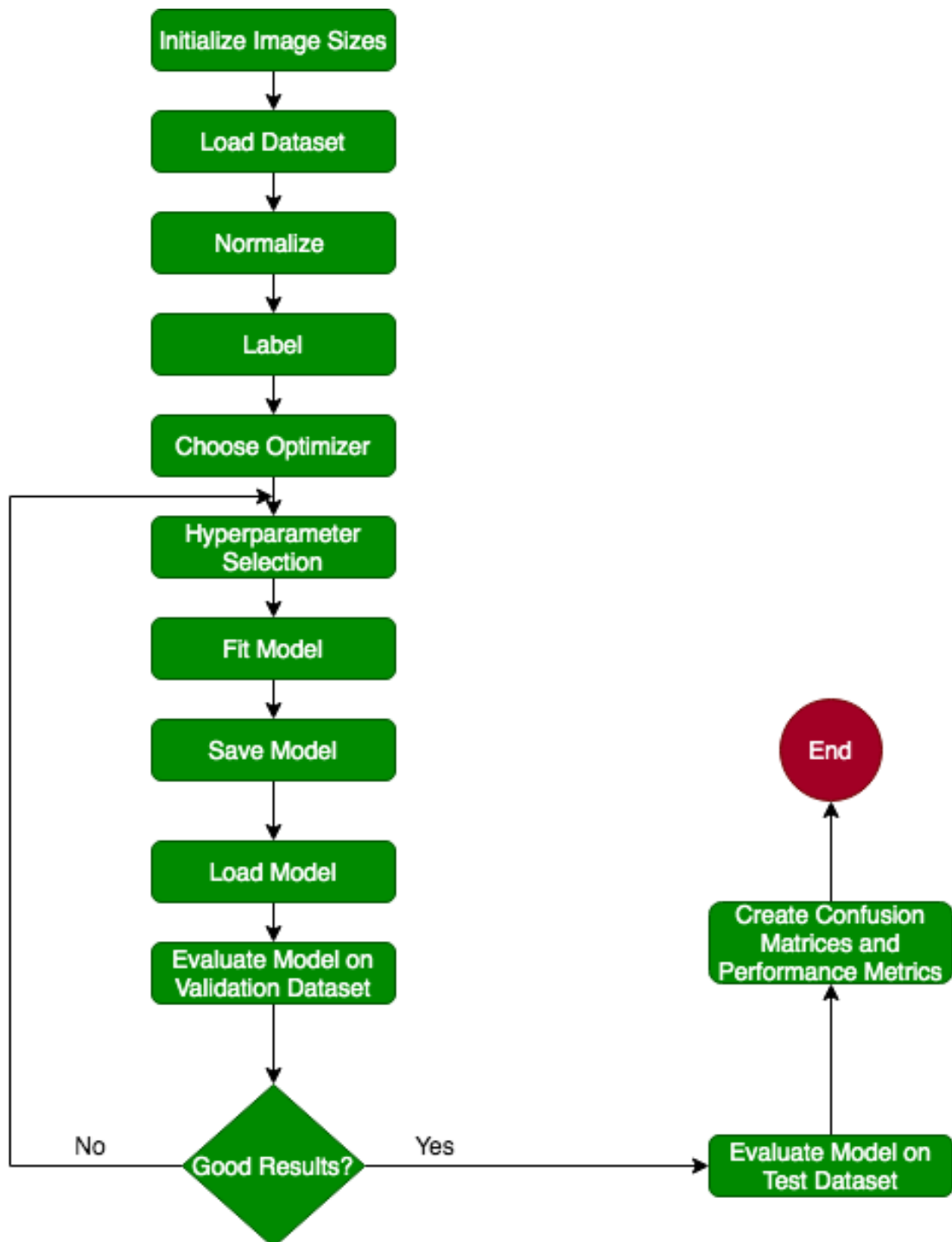


Figure 5.2: Flow diagram of experimental setup

5.1.2 Choice of Hyperparameters

Tuning of the hyperparameters has been achieved by training multiple models with different combinations of optimizers, learning rates, batch sizes, loss functions and dropout. The ultimate combination of hyperparameters was evaluated with the validation datasets.

Grid search is the strategy used for hyperparameter optimization in the experiment for both binary and multiclass classification.

5.1.2.1 Learning Rate

The momentum was set to 0.9 with the use of SGD. The learning rates were set in the initial grid search, and further adjustments of the learning rates were performed based on the observed results. The best learning rate with SGD is 0.0001 while that for Adam is 0.001.

5.1.2.2 Dropout

Dropout was implemented to solve overfitting problems. The dropout rate for architecture A was 0.25, for architecture B and C was 0.3, and for the transfer learning models was 0.1.

5.2 Trial Experiment for Superior Features

The aim of this experiment was to determine if MFCC or log-scaled mel spectrograms were the best set of features with high model performance for our particular case using Reduced10db dataset. Binary classification of P17 using 1-versus-others method described above was used for this experimentation. By using architecture A, it was concluded that based on the epoch versus accuracy graphs (Figure 5.3), the log-scaled mel spectrogram features are better for model training. This feature is henceforth used for the rest of the experimentation process.

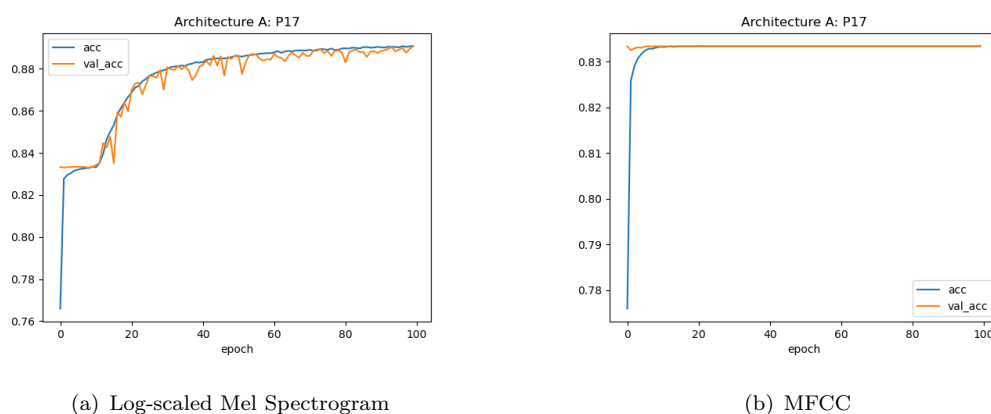


Figure 5.3: Epochs versus Accuracy Graphs. The graphs show two features used for training and validation of P17 class

5.3 Presentation of Results

Following the experimentation of the models presented above, this section shows the results based on using Reduced10db and Reduced15db datasets. For each model, a comparison of the epochs versus accuracy, validation accuracy, loss and validation is plotted to make conclusions on model performance and stability. The results and some plots will be shown in the next subsections. The plots for classes P09,P13 and P23 can be found in Appendix B.

5.3.1 Experiment 1: Binary Classification (1-versus-others)

Experiment 1 involves the use of architectures A, B and C, as well as ResNet18 to perform binary classification using the second binary classification method(Figure 4.7b). The results of these experiments are shown in the following subsections.

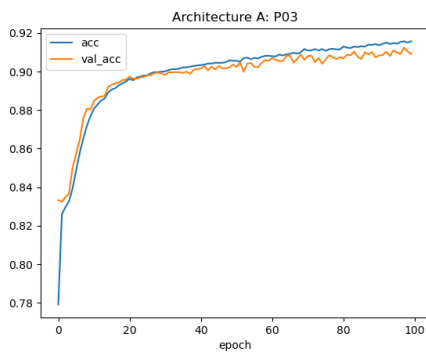
5.3.1.1 Architecture A: Binary (1-versus-others)

This experiment relates to the use of architecture A in binary classification of the datasets. Architecture A is run for 100 epochs and with SGD optimiser of learning rate 0.0001. The results for the validation and test datasets are shown in table(5.1).

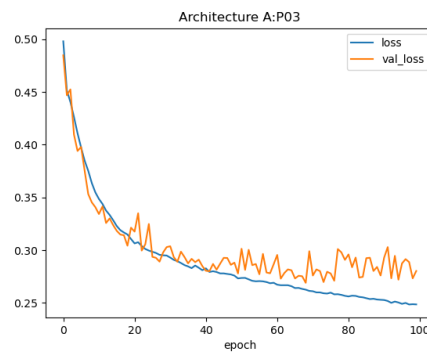
Table 5.1: Binary classification results for architecture A

Class	Metrics									
	Validation Set					Test Set				
	Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
Reduced10db										
P03	0.9090	0.8471	0.9100	0.9100	0.89	0.9080	0.8369	0.9029	0.9010	0.89
P09	0.9034	0.7932	0.9168	0.9057	0.89	0.9022	0.7868	0.9116	0.9005	0.89
P13	0.8664	0.7017	0.8892	0.8649	0.86	0.8642	0.6964	0.8792	0.8629	0.86
P17	0.8907	0.7629	0.9000	0.8988	0.87	0.8836	0.7437	0.8900	0.8800	0.86
P23	0.8333	0.6447	0.6687	0.8235	0.78	0.8333	0.6367	0.6589	0.8211	0.78
Background10db	0.8385	0.6883	0.6998	0.8385	0.79	0.8340	0.6813	0.6997	0.8350	0.79
Reduced15db										
P03	0.8849	0.7377	0.9270	0.8865	0.88	0.8851	0.7298	0.9242	0.8800	0.88
P09	0.8968	0.7456	0.9188	0.9076	0.89	0.8941	0.7377	0.9168	0.9043	0.89
P13	0.8599	0.6622	0.8819	0.8658	0.81	0.8589	0.6556	0.8763	0.8622	0.81
P17	0.8710	0.6804	0.8900	0.8700	0.84	0.8714	0.6814	0.8900	0.8700	0.87
P23	0.8333	0.6356	0.6942	0.8350	0.79	0.8333	0.6339	0.6882	0.8330	0.79
Background15db	0.8405	0.6558	0.8577	0.8100	0.84	0.8403	0.6501	0.8554	0.8049	0.84

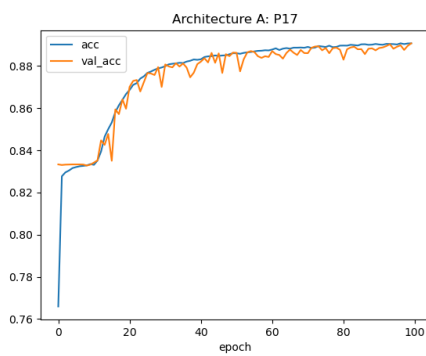
Figures(5.4) show the model performance(accuracy and loss) with increasing number of epochs. The classes shown are P03,P17 and background10db using Reduced10db dataset. The plots for the rest of the classes can be found in Appendix B.



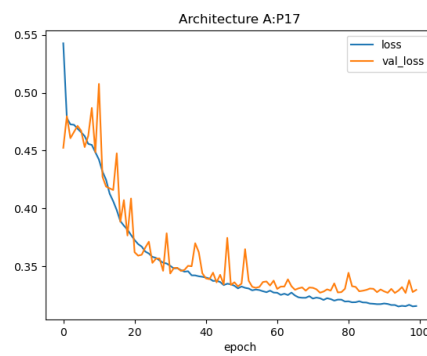
(a) P03



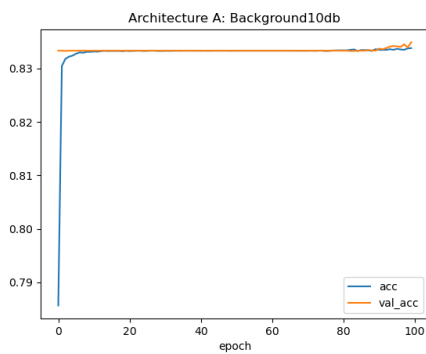
(b) P03



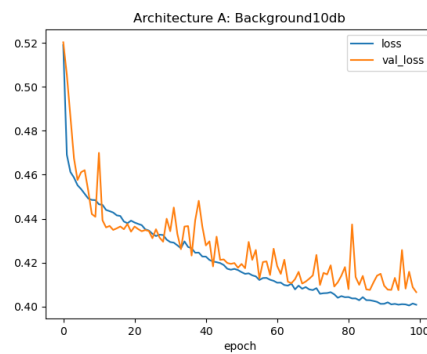
(c) P17



(d) P17



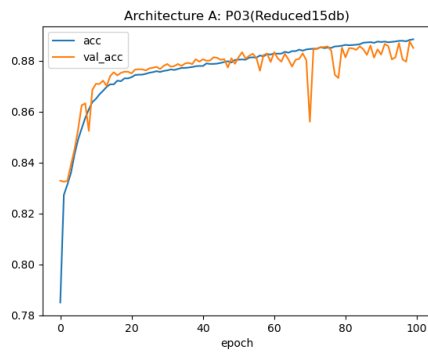
(e) Background10db



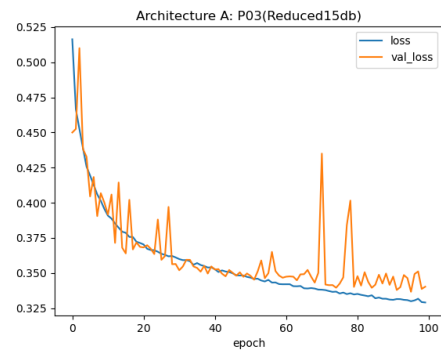
(f) Background10db

Figure 5.4: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background10db. Early stopping can be applied after 40 epochs to avoid overfitting for all the classes.

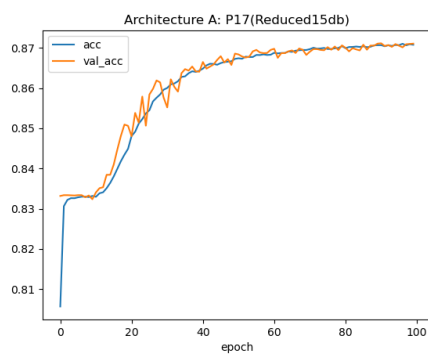
Figure 5.5 shows the model performance(accuracy and loss) with increasing number of epochs. The classes shown are P03,P17 and background10db using Reduced15db dataset. The plots for the rest of the classes can be found in Appendix B.



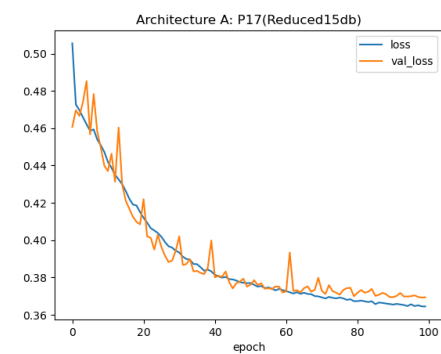
(a) P03



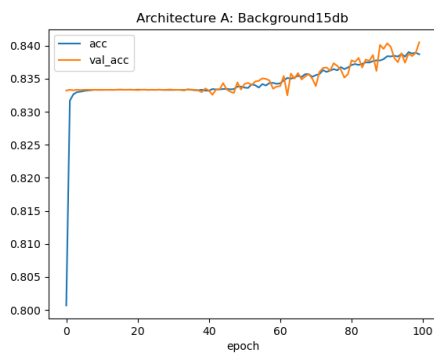
(b) P03



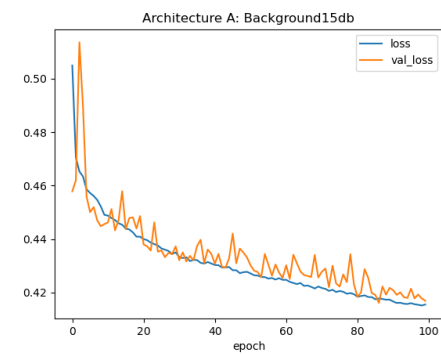
(c) P17



(d) P17



(e) Background15db



(f) Background15db

Figure 5.5: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background15db. Using Reduced15db, the model runs for longer epochs before overfitting starts to show. Early stopping after 65 epochs can be applied.

5.3.1.2 Architecture B: Binary (1-versus-others)

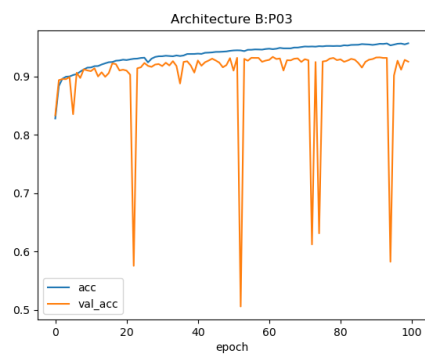
This architecture is experimented with Adam and at a learning rate of 0.001. This optimizer gave the best results for the above stated architecture. The results obtained

Table 5.2: Binary classification results for architecture B

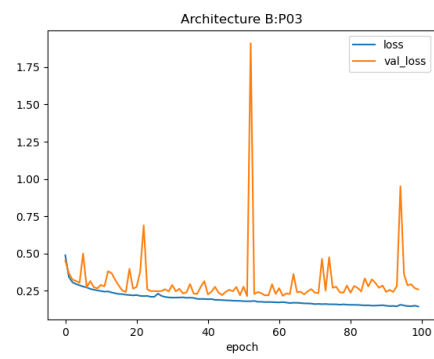
Class	Metrics									
	Validation Set					Test Set				
	Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
Reduced10db										
P03	0.9254	0.8618	0.9345	0.9140	0.93	0.9246	0.8449	0.9450	0.9117	0.93
P09	0.9127	0.8094	0.9192	0.9174	0.91	0.9111	0.7978	0.9172	0.9170	0.91
P13	0.8831	0.7508	0.9093	0.8866	0.87	0.8801	0.7475	0.8995	0.8865	0.87
P17	0.8948	0.7695	0.9533	0.8993	0.89	0.8900	0.7557	0.9489	0.8948	0.89
P23	0.8333	0.7186	0.8263	0.6068	0.65	0.8333	0.7119	0.8260	0.6063	0.65
Background10db	0.8461	0.7139	0.7708	0.7879	0.78	0.8447	0.7128	0.7717	0.7869	0.78
Reduced15db										
P03	0.9010	0.7770	0.9325	0.8964	0.89	0.9001	0.7736	0.9259	0.8944	0.88
P09	0.8878	0.7294	0.8889	0.8867	0.86	0.8859	0.7358	0.8887	0.8864	0.86
P13	0.8668	0.6857	0.7856	0.6262	0.67	0.8658	0.6837	0.7756	0.6260	0.67
P17	0.8736	0.7005	0.9489	0.8762	0.87	0.8740	0.7054	0.9487	0.8760	0.87
P23	0.8333	0.7016	0.7958	0.6564	0.65	0.8333	0.6980	0.7946	0.6563	0.65
Background15db	0.8333	0.6217	0.7556	0.8054	0.77	0.8323	0.6238	0.7557	0.8051	0.77

are shown below in table 5.2 for both datasets.

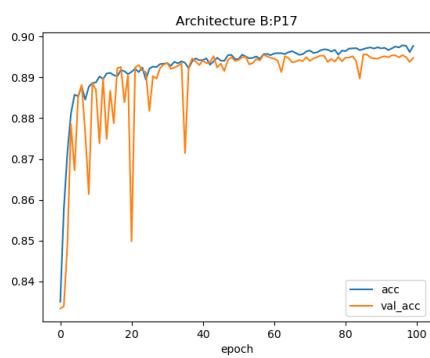
Figure 5.6 and 5.7 shows a comparison of the loss and accuracy measurements for Reduced10db and Reduced15db respectively. For both datasets, the validation accuracy and loss curves do not smooth out as compared to the plots in architecture A.



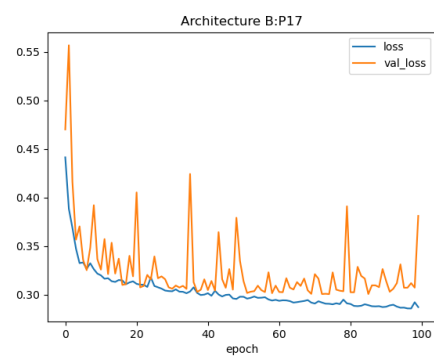
(a) P03



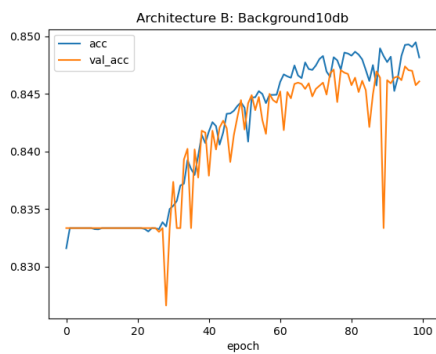
(b) P03



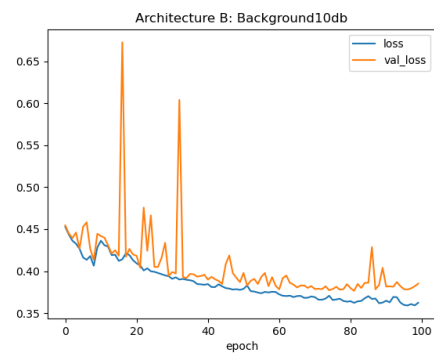
(c) P17



(d) P17

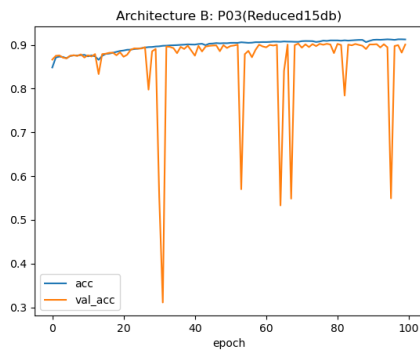


(e) Background10db

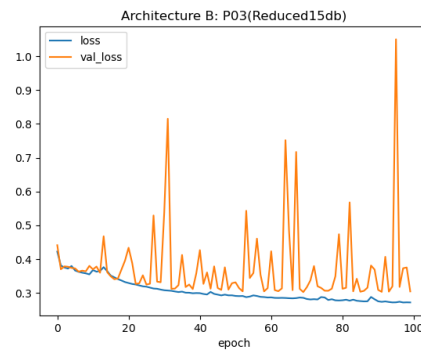


(f) Background10db

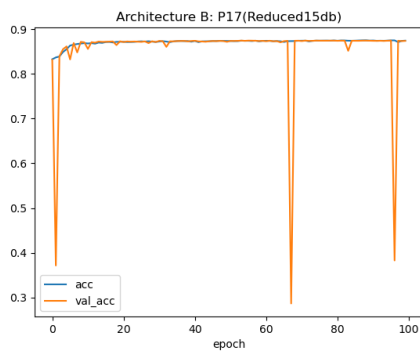
Figure 5.6: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background10db



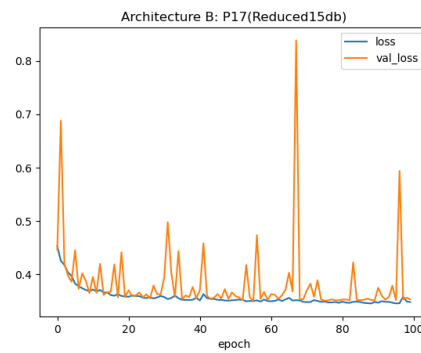
(a) P03



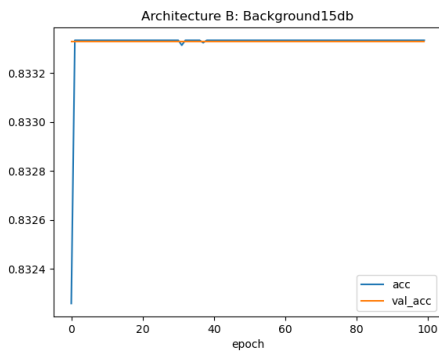
(b) P03



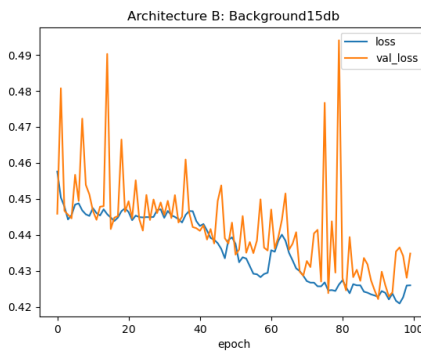
(c) P17



(d) P17



(e) Background15db



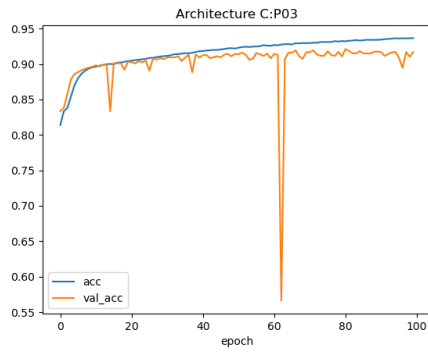
(f) Background15db

Figure 5.7: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(right hand side) and epochs versus loss(left hand side) of the training and validation of P03, P17 and background15db

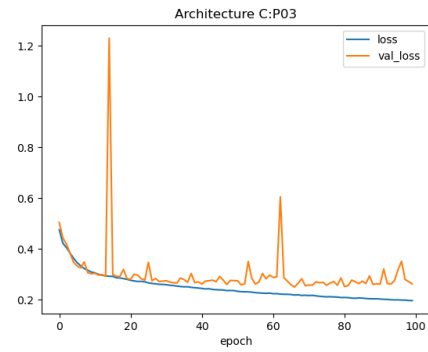
5.3.1.3 Architecture C: Binary (1-versus-others)

Architecture C is optimized with SGD with a learning rate of 0.0001 just like in architecture A. Due to the smaller number of convolutional layers in this architecture, the experimentation is faster as compared to both architectures A and B. The results are

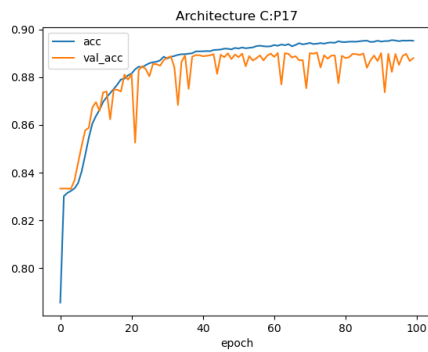
shown in table (5.3). Figures(5.8) and (5.9) show the model performance(accuracy and loss) with increasing number of epochs. The classes shown are P03,P17 and background noise class for both datasets. The plots for the rest of the classes can be found in Appendix B. In these plots, it can be deduced that, architecture C begins to over fit after 30 epochs for Reduced10db dataset and after 50 epochs for Reduced15db dataset.



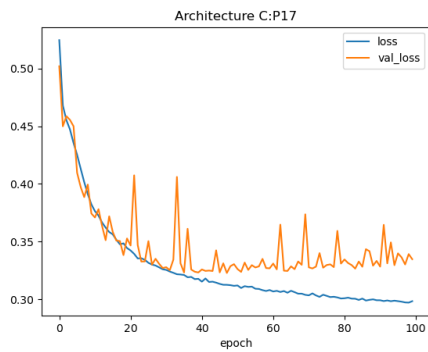
(a) P03



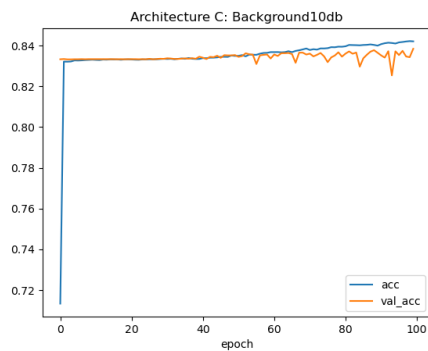
(b) P03



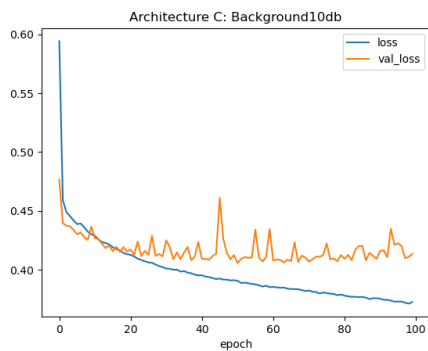
(c) P17



(d) P17



(e) Background10db

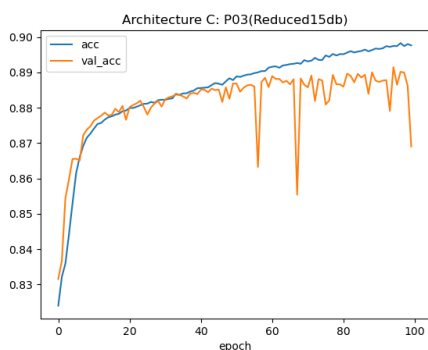


(f) Background10db

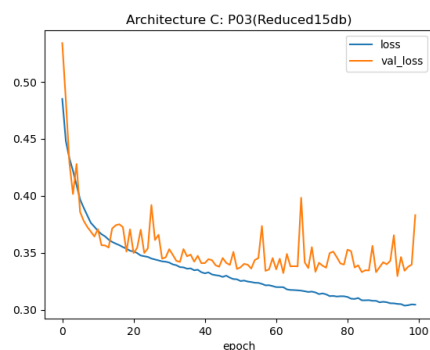
Figure 5.8: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background10db

Table 5.3: Binary classification results for architecture C

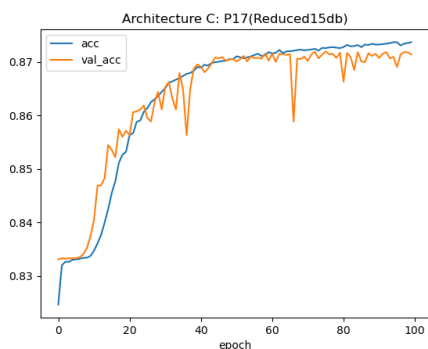
Class	Metrics									
	Validation Set					Test Set				
	Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
Reduced10db										
P03	0.9170	0.8720	0.9273	0.9226	0.92	0.9153	0.8562	0.9251	0.9225	0.92
P09	0.9041	0.7922	0.9194	0.9173	0.90	0.9023	0.7922	0.9173	0.9147	0.90
P13	0.8758	0.7155	0.8992	0.8762	0.85	0.8715	0.6997	0.8891	0.8756	0.85
P17	0.8879	0.7546	0.9080	0.8945	0.89	0.8844	0.7366	0.9213	0.8951	0.89
P23	0.8333	0.6804	0.6921	0.8344	0.76	0.8333	0.6697	0.6858	0.8336	0.76
Background10db	0.8384	0.6865	0.8675	0.8454	0.84	0.8363	0.6825	0.8539	0.8342	0.83
Reduced15db										
P03	0.8690	0.7417	0.8761	0.8844	0.87	0.8673	0.7433	0.8860	0.8845	0.87
P09	0.8861	0.7243	0.8697	0.8678	0.86	0.8842	0.7010	0.8619	0.8642	0.86
P13	0.8576	0.6604	0.8685	0.8680	0.84	0.8559	0.6576	0.8648	0.8660	0.84
P17	0.8714	0.6799	0.9497	0.8724	0.87	0.8728	0.6836	0.9459	0.8728	0.87
P23	0.8333	0.6348	0.6942	0.8350	0.76	0.8333	0.6278	0.6856	0.8311	0.76
Background15db	0.8365	0.6698	0.8622	0.8319	0.83	0.8392	0.6819	0.8650	0.8350	0.83



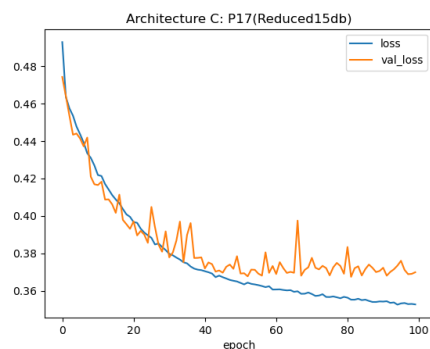
(a) P03



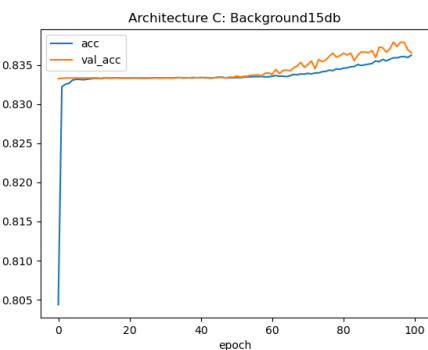
(b) P03



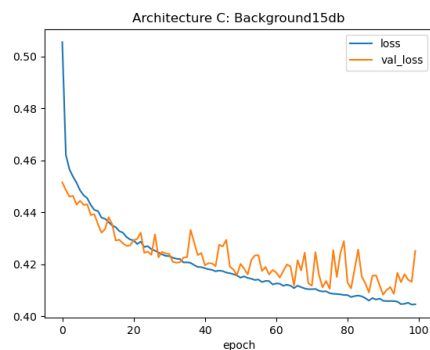
(c) P17



(d) P17



(e) Background15db



(f) Background15db

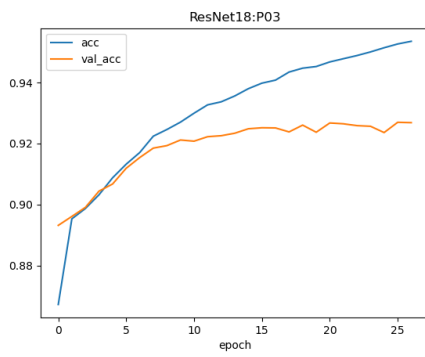
Figure 5.9: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background15db

5.3.1.4 ResNet18: Binary (1-versus-others)

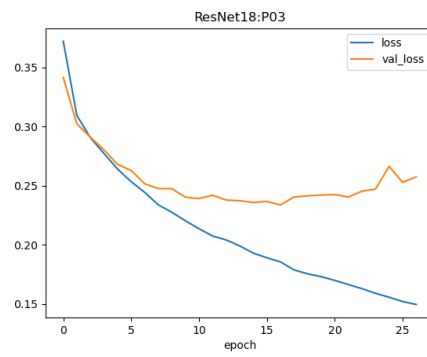
The ResNet18 model is optimized with SGD optimizer and a learning rate of 0.0001. This model is run for a fewer epochs of 20 as compared to the rest of the models. The results for binary classification using pre-trained and fine-tuned ResNet18 are shown in table(5.4). Figures (5.10 and 5.11) show the epochs versus accuracy and loss plots for both datasets.

Table 5.4: Binary classification results for ResNet18

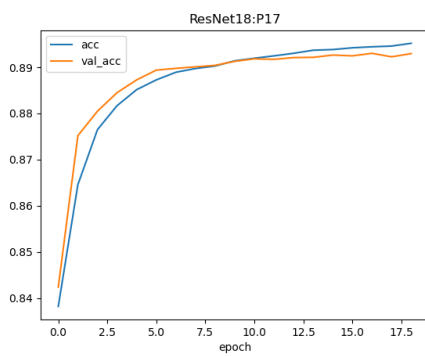
Class	Metrics									
	Validation Set					Test Set				
	Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
Reduced10db										
P03	0.9251	0.8887	0.9446	0.9276	0.92	0.9253	0.8751	0.9441	0.9266	0.92
P09	0.9104	0.8151	0.9295	0.9172	0.91	0.9099	0.8132	0.9153	0.9147	0.91
P13	0.8816	0.7352	0.8992	0.8864	0.86	0.8803	0.7457	0.8988	0.8854	0.86
P17	0.8904	0.7639	0.8856	0.9070	0.88	0.8832	0.7495	0.8793	0.8966	0.87
P23	0.8333	0.7192	0.7000	0.8400	0.74	0.8333	0.7027	0.6967	0.8388	0.74
Background10db	0.8405	0.7023	0.7195	0.8454	0.84	0.8390	0.6872	0.7097	0.8350	0.83
Reduced15db										
P03	0.8984	0.7683	0.9319	0.8968	0.89	0.8984	0.7633	0.9019	0.8917	0.89
P09	0.8848	0.7097	0.9094	0.8865	0.86	0.8835	0.7097	0.9068	0.8842	0.86
P13	0.8667	0.6666	0.8891	0.8659	0.83	0.8635	0.6620	0.8847	0.8621	0.83
P17	0.8738	0.6812	0.8687	0.8879	0.83	0.8723	0.6818	0.8687	0.8776	0.85
P23	0.8333	0.6187	0.6947	0.8331	0.74	0.8333	0.6148	0.6921	0.8299	0.74
Background15db	0.8515	0.7376	0.7076	0.8556	0.85	0.8488	0.7305	0.8676	0.8546	0.85



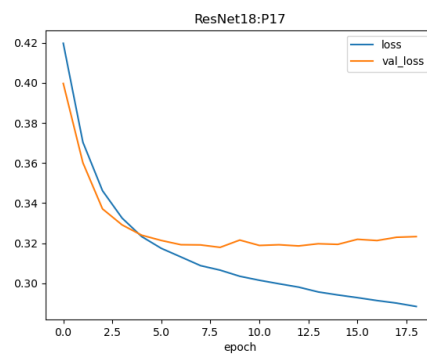
(a) P03



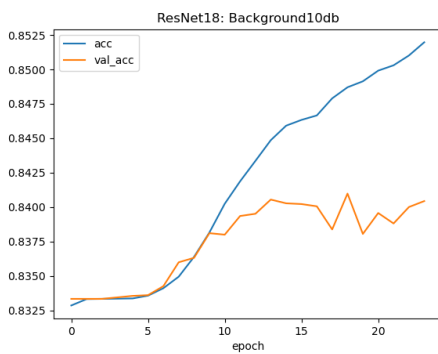
(b) P03



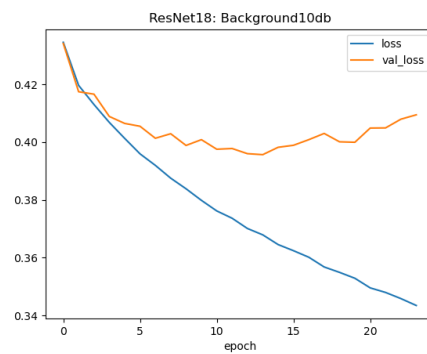
(c) P17



(d) P17

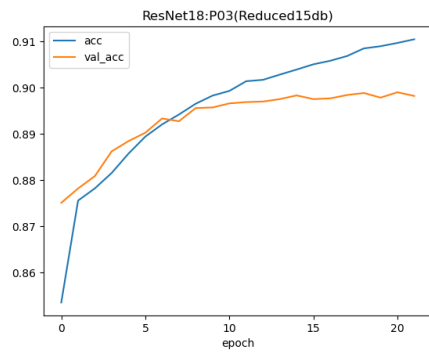


(e) Background10db

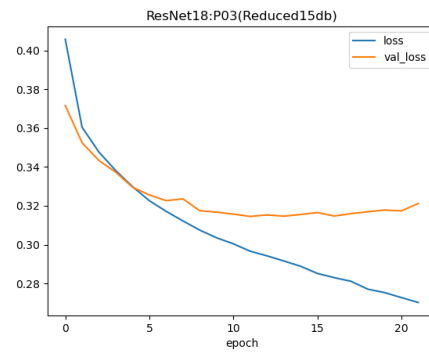


(f) Background10db

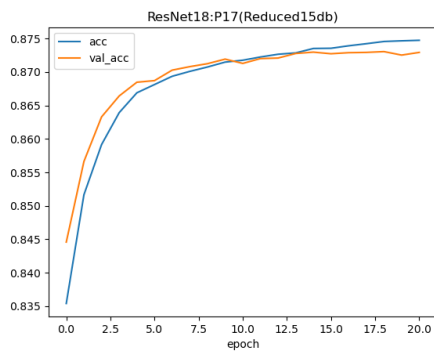
Figure 5.10: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background10db



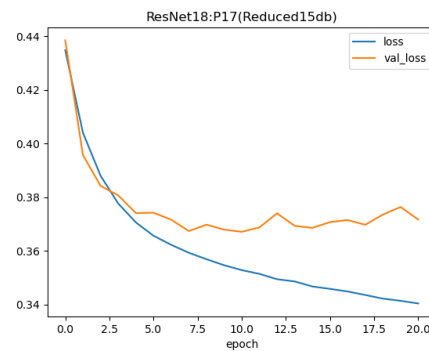
(a) P03



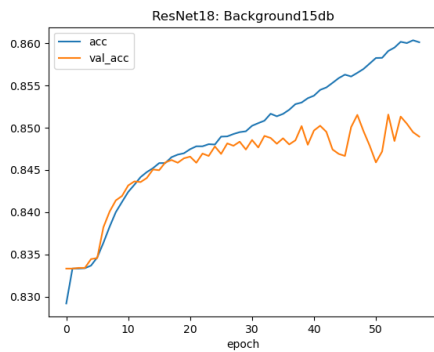
(b) P03



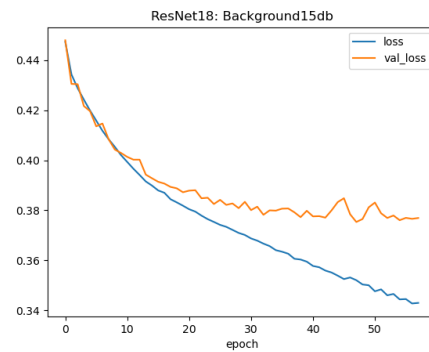
(c) P17



(d) P17



(e) Background15db



(f) Background15db

Figure 5.11: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P03, P17 and background15db

5.3.2 Experiment 2: Binary Classification (1-versus-1)

Experiment 2 is a binary classification experiment which is conducted to test how similar classes can be classified by model architectures A, B and C, as well as ResNet18, as explained in chapter 4. In this experiment, the '1-versus-1' binary model(Figure 4.7a)

is used to test the models on two similar classes. The classes are P13 and P17 which correspond to the AED commands 'no shock advised' and 'shock advised' respectively. This is because the AED command that makes the difference in these two classes is 'no'. P17 is labelled '1' and P13 is labelled '0' and a 0.5 threshold is applied for evaluation and prediction. The results for the experiments for Reduced15db are therefore shown in the following subsections.

5.3.2.1 Architecture A: Binary Classification (1-versus-1)

Since fewer data is used for this experiment, there is a reduction in model accuracy for both datasets in comparison to experiment 1. Table 5.5 shows the results obtained for Reduced15db.

Figure 5.12 shows the training and validation accuracy and loss of this experiment.

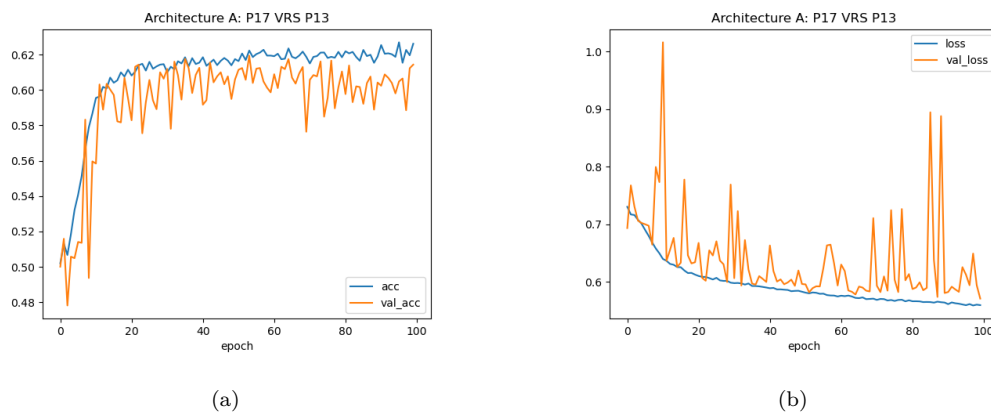


Figure 5.12: Comparison of binary classification of P13 and P17 using architecture A. The graphs display epochs versus accuracy and loss graphs for Reduced15db

5.3.2.2 Architecture B: Binary Classification (1-versus-1)

The plots in figure 5.13 show the accuracy and loss function plots for 100 epochs of training and validation. Subsequently, table 5.5 shows the model performance results for Reduced15db.

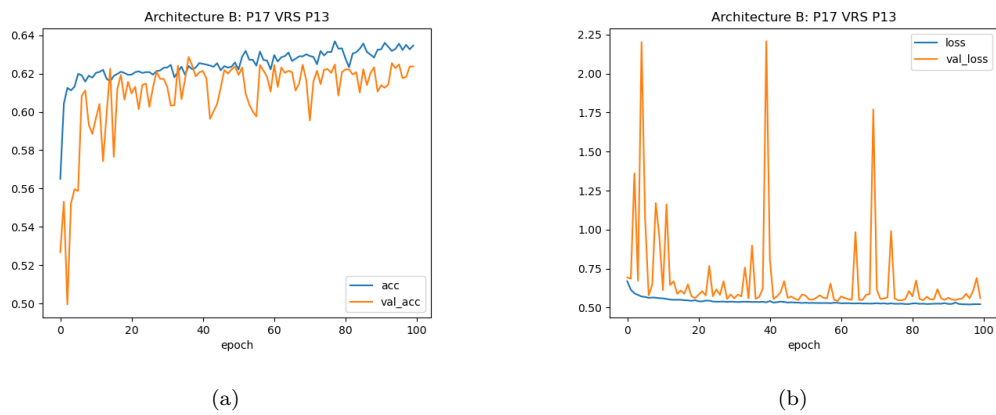


Figure 5.13: Comparison of binary classification of P13 and P17 using architecture B. The graphs display epochs versus accuracy and loss graphs for Reduced15db

5.3.2.3 Architecture C: Binary Classification (1-versus-1)

The plots in figure 5.14 show the accuracy and loss plots for 100 epochs of training and validation. Subsequently, table 5.5 shows the model performance results for Reduced15db.

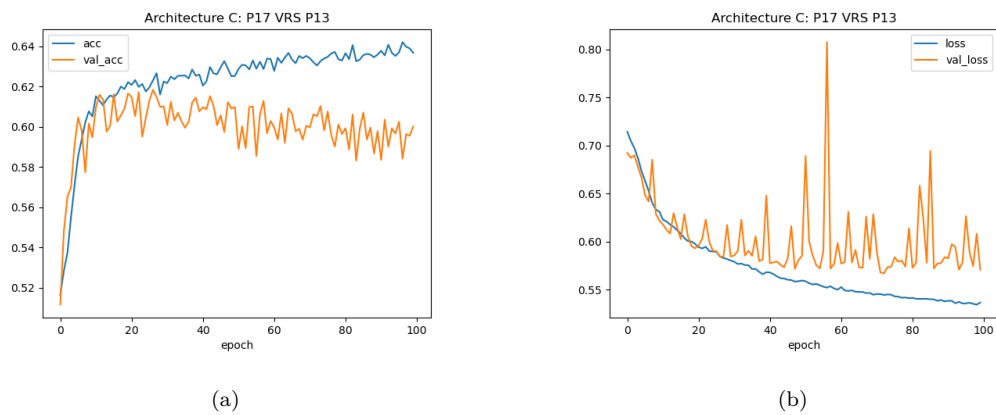


Figure 5.14: Comparison of binary classification of P13 and P17 using architecture C. The graphs display epochs versus accuracy and loss graphs

5.3.2.4 ResNet18: Binary Classification (1-versus-1)

The ResNet18 model was run for 30 epochs. This is because of the overfitting due to small dataset that was used. Table 5.5 shows the results obtained for this experiment. The plots in figure 5.15 show the training and validation accuracy and loss of this experiment.

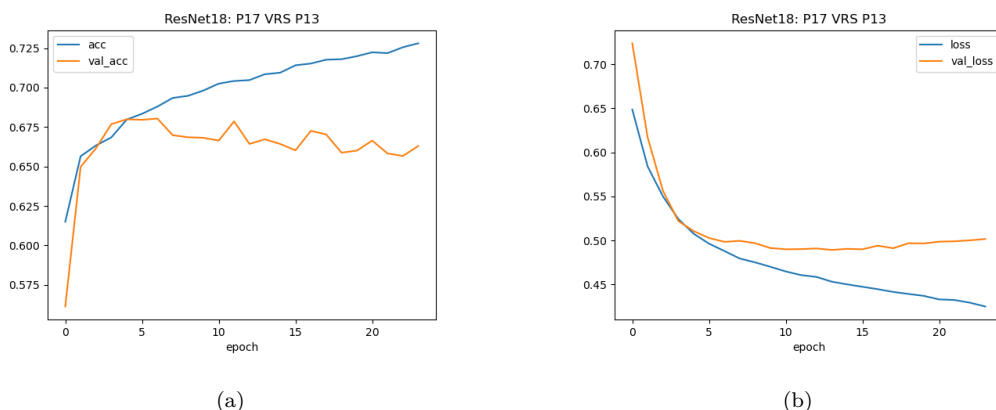


Figure 5.15: Comparison of binary classification of P13 and P17 using ResNet18. The graphs display epochs versus accuracy and loss graphs

Table 5.5: Model performance results for architectures A, B, C and ResNet18. The table shows the results obtained by binary classification of two similar classes using Reduced15db dataset

Architecture	Metrics									
	Validation Set					Test Set				
	Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
A	0.6260	0.7220	0.5860	0.8738	0.63	0.6252	0.7037	0.5837	0.8575	0.63
B	0.6286	0.7186	0.8420	0.3167	0.63	0.6237	0.7037	0.8425	0.3042	0.62
C	0.6007	0.6834	0.7086	0.3419	0.60	0.6002	0.6788	0.7044	0.3452	0.60
ResNet18	0.6065	0.6953	0.5726	0.8409	0.61	0.6029	0.6915	0.5626	0.8399	0.60

5.3.3 Experiment 3: Multiclass Classification

In this experiment, multiclass classification is performed on the datasets. The classes are categorically labelled from zero to five and then experiments are conducted with architectures A, B,C and ResNet18, as described in Chapter 4, and as used for the binary classification tasks. The prediction layer uses a softmax activation function and the loss function used is 'categorical-crossentropy'. Furthermore, multiclass classification is performed in the 5 AED audio classes without the background15db class in the first part of this experiment, and then all the 5 classes plus the background15db class are used for the experimentation in the second part of this experiment. The results for each architecture is shown using Reduced15db dataset in the following subsections.

5.3.3.1 Architecture A: Multiclass Classification

The model is trained for 30 epochs for both 5 and 6 class multiclass classification tasks. On the top two figures of Figure 5.16, the 5-class multi classification plots a shown while

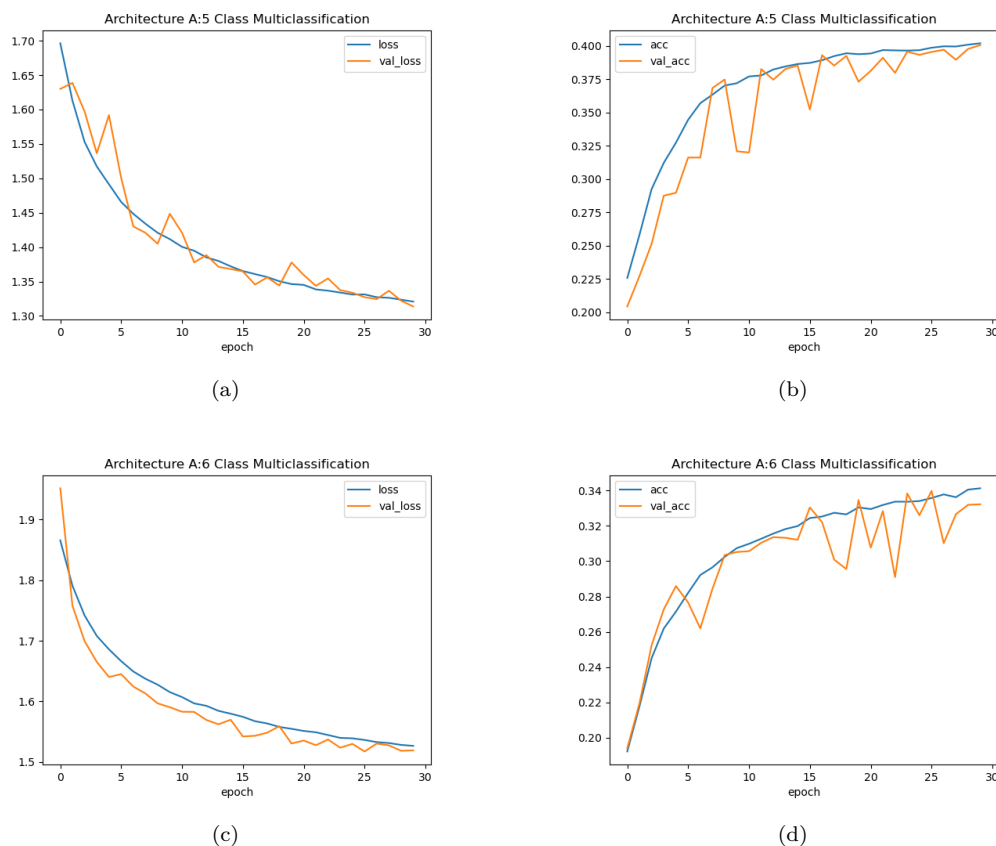


Figure 5.16: Architecture A training and validation plots for multiclass classification. The graphs display epochs versus accuracy and loss graphs

the bottom plots represent the 6-class multi classification plots. Table 5.6 shows the results for this model architecture.

Table 5.6: Multi-class classification metrics for architecture A

Dataset	Number of Classes	Metrics									
		Validation Set					Test Set				
		Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
Reduced15db	5	0.4050	0.7171	0.9410	0.2000	0.41	0.4001	0.7118	0.9437	0.1897	0.41
	6	0.3350	0.6908	0.9486	0.1522	0.33	0.3321	0.6861	0.9424	0.1492	0.33

Furthermore, the classification report for all six classes is shown in Table 5.7. The report shows the precision, recall and F1 score for each individual class. For this architecture, class P23 is the most difficult to train. It has the lowest precision of 18 percent and a recall of 2 percent. The effect of this low scores for P23, affects the overall results seen in table 5.6 above. Also, the effect of the background15db class is evident has it has a very low precision score of 0.21.

Table 5.7: Multi-class Classification Report: Architecture A

Class	Precision	Recall	F1 Score	Number of Test Samples
P03	0.68	0.36	0.37	2305
P09	0.86	0.30	0.44	2305
P13	0.61	0.19	0.28	2305
P17	0.59	0.27	0.37	2305
P23	0.18	0.02	0.04	2305
Background15db	0.21	0.88	0.33	2305
Macro Average	0.52	0.33	0.32	13830
Weighted Average	0.52	0.33	0.32	13830

5.3.3.2 Architecture B: Multiclass Classification

The model is also trained for 30 epochs for both 5- and 6-class multi classification tasks. On the top two figures of Figure 5.17, the 5-class multi classification plots a shown while the bottom plots represent the 6-class multi classification plots. Table 5.8 shows the results for this model architecture.

Table 5.8: Multiclass classification metrics for architecture B

Dataset	Number of Classes	Metrics									
		Validation Set					Test Set				
		Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
Reduced15db	5	0.3738	0.6981	0.8525	0.2363	0.37	0.3689	0.6923	0.8361	0.2293	0.37
	6	0.3783	0.7271	0.9847	0.2001	0.38	0.3847	0.7237	0.9802	0.1945	0.37

Architecture B is better at classifying P23 than architecture A. In general, the classification report(Table 5.9) for individual classes using architecture B is better than the previous architecture. It can also be observed that, this architecture gives almost the same results even when the background15db class is removed.

Table 5.9: Multiclass Classification Report: Architecture B

Class	Precision	Recall	F1 Score	Number of Test Samples
P03	0.95	0.40	0.56	2305
P09	0.87	0.33	0.48	2305
P13	0.86	0.23	0.37	2305
P17	0.82	0.26	0.40	2305
P23	0.21	0.96	0.35	2305
Background15db	0.83	0.09	0.16	2305
Macro Average	0.76	0.38	0.38	13830
Weighted Average	0.76	0.38	0.38	13830

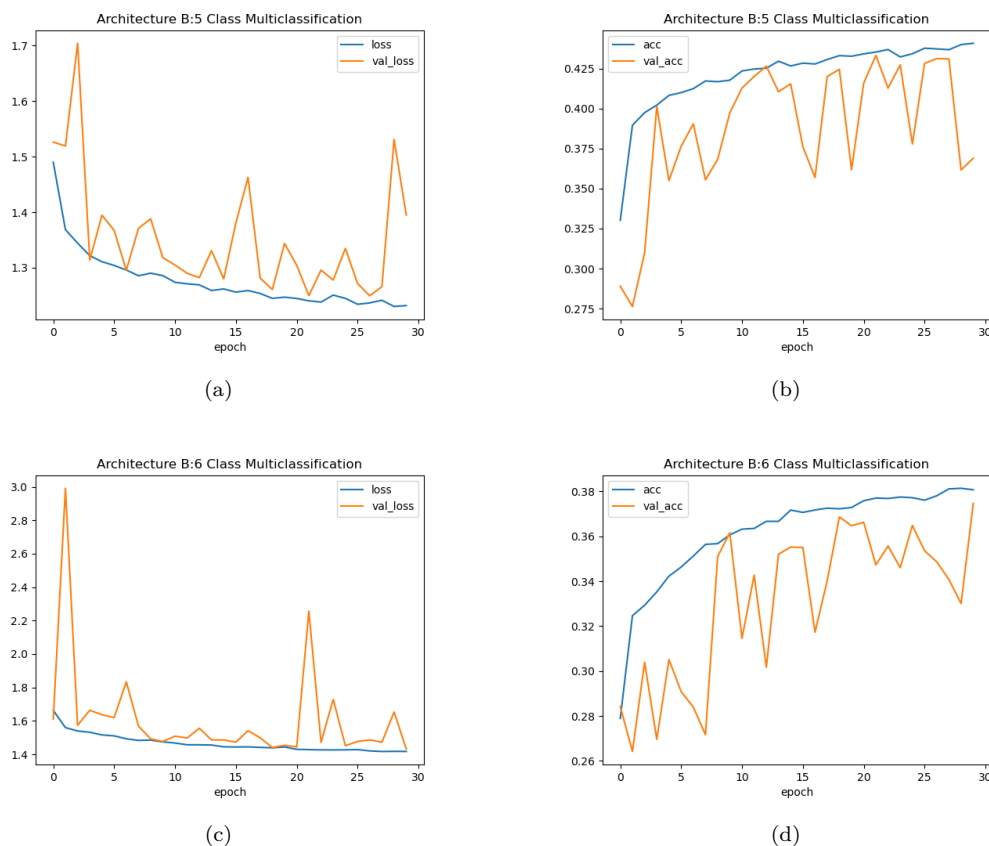


Figure 5.17: Architecture B training and validation plots for multiclass classification. The graphs display epochs versus accuracy and loss graphs

5.3.3.3 Architecture C: Multiclass Classification

This model is the shallowest of all the models and it trains faster than the rest of the models. It is trained for 30 epochs just like the previous models on multiclass classification. On the top two figures on figure 5.18, the 5-class multi classification plots a shown while the bottom plots represent the 6-class multi classification plots. Table 5.10 shows the results for this model architecture.

Table 5.10: Multiclass classification metrics for architecture C

Dataset	Number of Classes	Metrics									
		Validation Set					Test Set				
		Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
Reduced15db	5	0.5266	0.8156	0.9179	0.3215	0.53	0.5226	0.7994	0.8959	0.3198	0.53
	6	0.4576	0.7966	0.9158	0.2841	0.46	0.4556	0.7921	0.8987	0.2811	0.46

This architecture outperforms the rest of the proposed models and the transfer learning models in multiclass classification task. The individual classification report (Table 5.11) shows that, the model is able to classify P23 and background15db classes with better

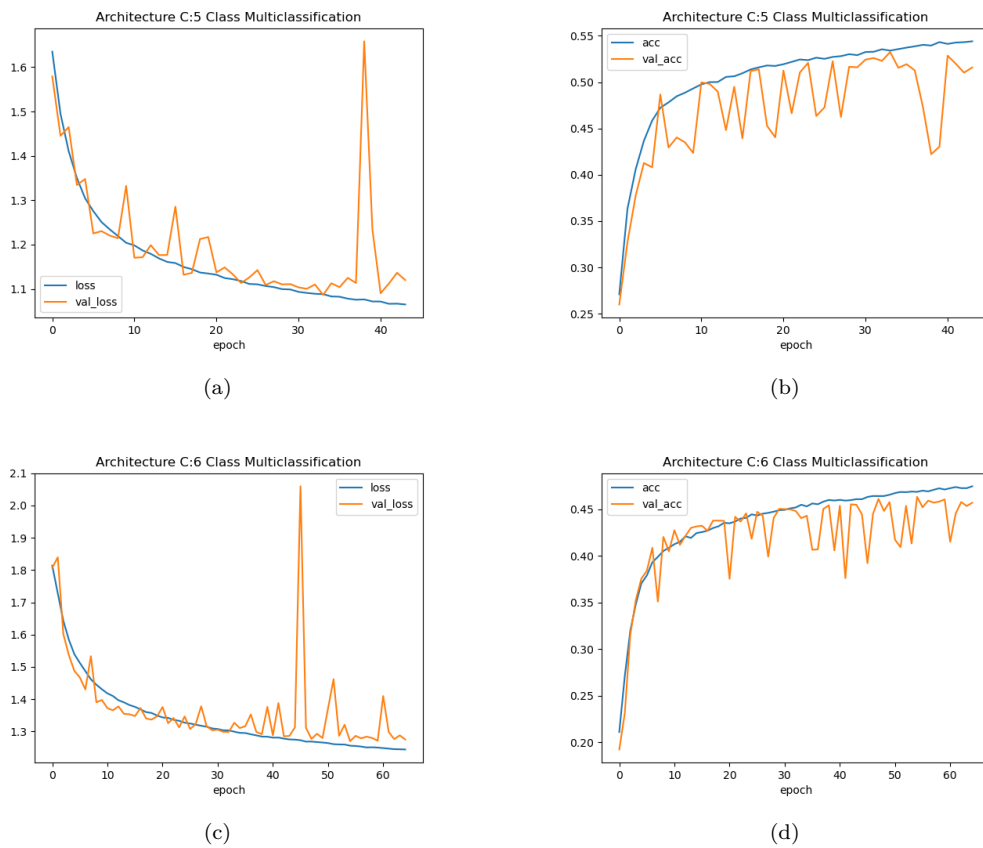


Figure 5.18: Architecture C training and validation plots for multiclass classification. The graphs display epochs versus accuracy and loss graphs

precision and recall. This results in the overall higher macro and weighted average scores for the statistical measurements used in this work.

Table 5.11: Multiclass Classification Report: Architecture C

Class	Precision	Recall	F1 Score	Number of Test Samples
P03	0.76	0.60	0.67	2305
P09	0.73	0.49	0.59	2305
P13	0.69	0.36	0.47	2305
P17	0.82	0.34	0.49	2305
P23	0.25	0.78	0.38	2305
Background15db	0.35	0.17	0.23	2305
Macro Average	0.60	0.46	0.47	13830
Weighted Average	0.60	0.46	0.47	13830

5.3.3.4 ResNet18: Multiclass Classification

The results for multiclass classification tasks using ResNet18 is shown below. The top two figures on figure 5.19, the 5-class multi classification plots a shown while the bottom plots represent the 6-class multi classification plots. Table 5.12 shows the results for this model architecture.

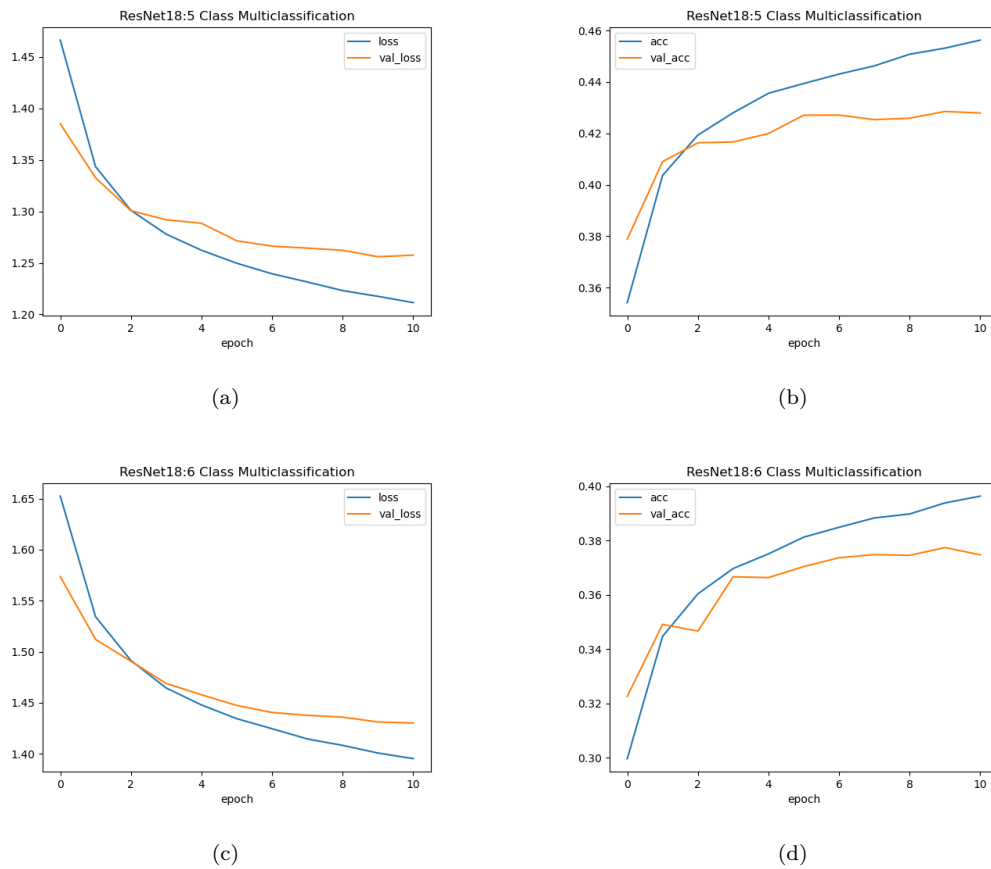


Figure 5.19: ResNet18 training and validation plots for multiclass classification. The graphs display epochs versus accuracy and loss graphs

Table 5.12: Multiclass classification metrics for ResNet18

Dataset	Number of Classes	Metrics									
		Validation Set					Test Set				
		Accuracy	AUC	Precision	Recall	F1 Score	Accuracy	AUC	Precision	Recall	F1 Score
Reduced15db	5	0.4296	0.7370	0.9490	0.2439	0.43	0.4221	0.7298	0.9331	0.2393	0.43
	6	0.3885	0.7301	0.9551	0.2200	0.39	0.3818	0.7257	0.9506	0.2150	0.38

Background15db class affects the overall macro and weighted scores for the 6-class multi-class classification results using ResNet18. This is evident from Table 5.13. The overall performance report is very similar to that of architecture A and it goes to confirm

that, P23 and background15db are the hardest classes to classify. That also explains the low scores obtained in experiment 3.

Table 5.13: Multi-class Classification Report: ResNet18

Class	Precision	Recall	F1 Score	Number of Test Samples
P03	0.79	0.47	0.59	2305
P09	0.68	0.36	0.47	2305
P13	0.64	0.26	0.37	2305
P17	0.68	0.27	0.39	2305
P23	0.22	0.73	0.34	2305
Background15db	0.34	0.23	0.28	2305
Macro Average	0.56	0.39	0.40	13830
Weighted Average	0.56	0.39	0.40	13830

5.4 Analysis of Results

This section presents some analysis of the results obtained so far in the experiments.

5.4.1 Experiment 1: Binary Classification (1-versus-others)

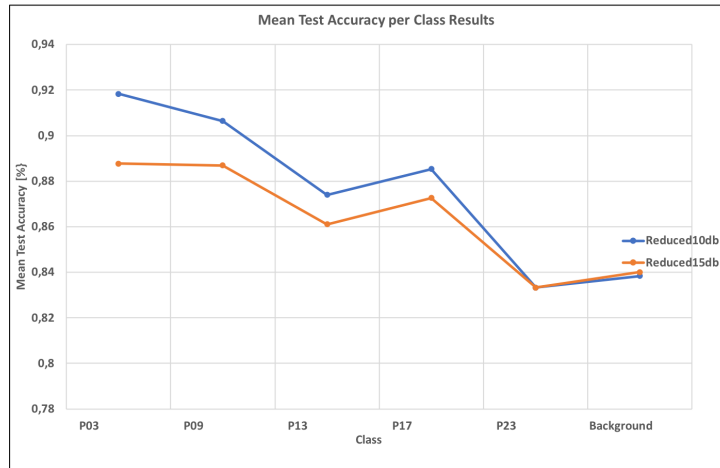
To analyse how the models performed in this experiment, the mean test accuracy and F1 score are calculated for each class (Table 5.14).

Table 5.14: Mean Test Accuracy and F1 Score Results. The results compare the mean test accuracy and f1 score between the classes used in experiment 1

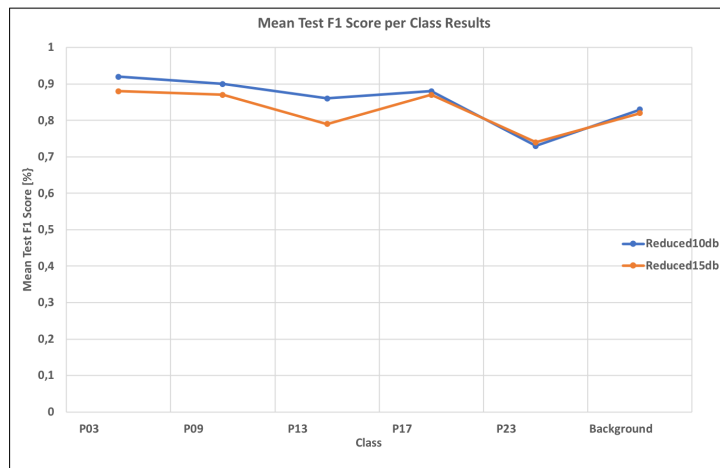
Class	Mean Test Accuracy		Mean F1 Score	
	Reduced10db	Reduced15db	Reduced10db	Reduced15db
P03	0.9183	0.8877	0.92	0.88
P09	0.9064	0.8869	0.90	0.87
P13	0.8740	0.8610	0.86	0.79
P17	0.8853	0.8726	0.88	0.87
P23	0.8333	0.8333	0.73	0.74
Background	0.8384	0.8401	0.83	0.82

The test accuracy for all classes in the Reduced15db dataset is 4.65 percent lower for class P03. While for P09, it is 2.2 percent less. Generally, the percentage loss in the mean test accuracy for Reduced15db dataset is 0 to 4.65 percent lower across all the classes except for the background noise class which rather had 0.01 percent better accuracy

for the Reduced15db dataset as compared to the Reduced10db dataset(Figure 5.20). A similar behaviour is observed in the mean f1 score across all the classes. For each class, the Reduced15db dataset has 0 to 4.5 percent lower mean F1 score.



(a) Class versus Mean Test Accuracy



(b) Class versus Mean Test F1 Score

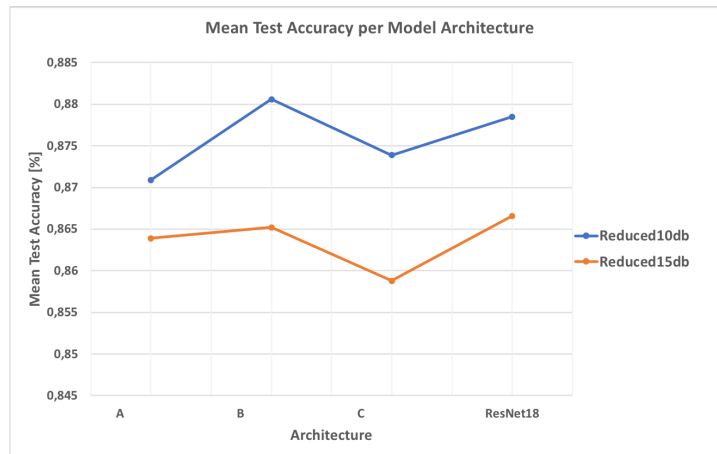
Figure 5.20: Analysis of mean test and F1 Scores for experiment 1

Furthermore, the mean test accuracy and F1 score for each model architecture is analysed to observe how each of the models performed in experiment 1. The mean of the test accuracy and F1 score for all classes in each model is calculated and presented in Table 5.15).

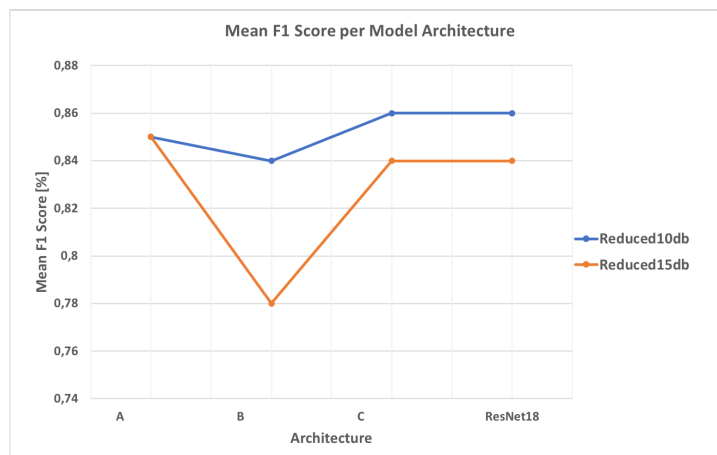
Table 5.15: Mean Test Accuracy and F1 Score Results. The results compare the mean test accuracy and F1 score between the architectures used in experiment 1

Architecture	Mean Test Accuracy		Mean F1 Score	
	Reduced10db	Reduced15db	Reduced10db	Reduced15db
A	0.8709	0.8639	0.85	0.85
B	0.8806	0.8652	0.84	0.78
C	0.8739	0.8588	0.86	0.84
ResNet18	0.8785	0.8666	0.86	0.84

The models obtained less test accuracy and F1 scores for Reduced15db dataset but within a smaller percentage difference of 0 to 0.15. Model architecture A has the same F1 score for both dataset while architecture has the lowest F1 score for Reduced15db dataset. The plots for the mean test accuracy and F1 scores are shown in Figure 5.21.



(a) Class versus Mean Test Accuracy



(b) Class versus Mean Test F1 Score

Figure 5.21: Analysis of Mean Test and F1 Scores for all architectures used in experiment 1

5.4.2 Experiment 2: Binary Classification (1-versus-1)

The average test accuracy and F1 score for this experiment is 0.6110 and 0.6 respectively and for all the models. The experiment was conducted for only 2 similar classes, using Reduced15db dataset.

5.4.3 Experiment 3: Multiclass Classification

In architecture A and ResNet18, there is a 20 percent increase in test accuracy when the experiment is conducted without the background noise class. The F1 score is improved slightly by 9 percent.

In architecture B, there is a slight decrease in model test accuracy when all 6 classes are used. This is different as compared to architecture A and C which had an increase in test accuracy when 5 classes are used. The F1 score is the same for both 5- and 6-class multi experiments.

In architecture C, the 5-class multi classification resulted in a 14.7 percent increase in test accuracy. This is a lower increase as compared to architecture A and ResNet18. The F1 score also increases by 15.3 percent in as compared to the 6-class multi classification.

6

Discussion and Future Works

This chapter discusses the experimental results achieved using the stated models in Chapter 5. Achieved results, encountered limitations, and possible improvements will be discussed. Furthermore, a final system is proposed and suggestions for future work will also be presented.

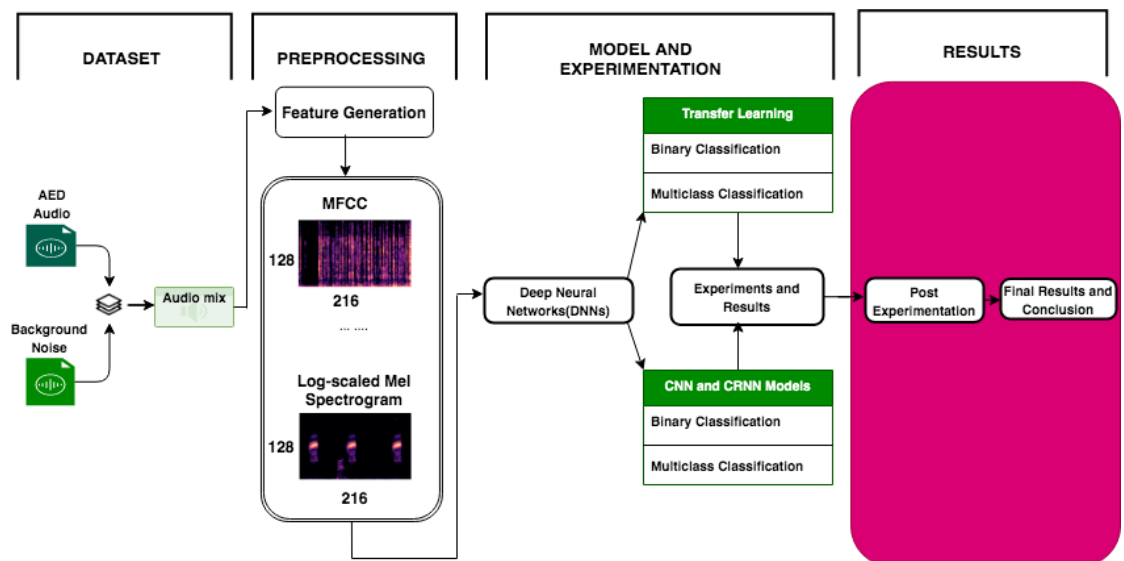


Figure 6.1: An illustration of the approach used for this chapter of the thesis

6.1 Model Performance

The datasets are balanced in this work, therefore, the main performance matrix considered for these models is the accuracy. Also, models' robustness is evaluated by using the F1 score. Furthermore, the cost of miss-classification in case of the implementation of these models is discussed by considering the precision and recall scores from each experiment. The following subsections discuss the results and observations under each experiment. Figure 6.2 shows a diagrammatic overview of the average training time per epoch of each model stated in Chapter 4. Each architecture was trained on GPU Tesla V100-PCIE (32GB).

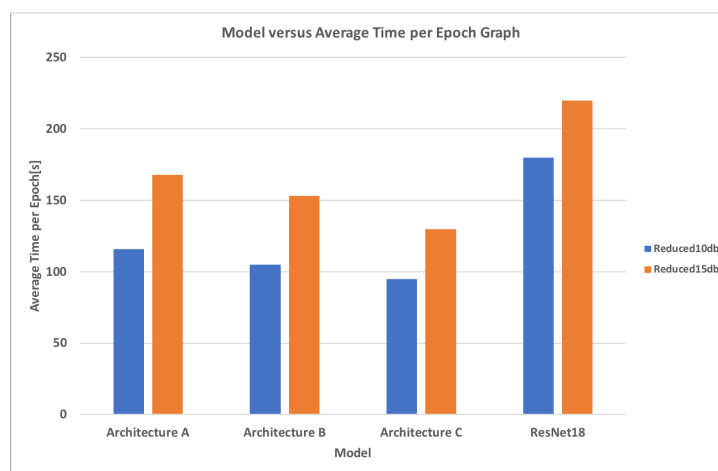


Figure 6.2: Model versus average time per epoch graph. The average training time for Reduced10db is higher than in Reduced15db. Architecture C trains the fastest while ResNet18 is the slowest training model.

6.1.1 Experiment 1: Binary Classification (1-versus-others)

The results from this experiment indicate that, for both datasets, ResNet18 is slightly better by 0.01 points in accuracy when compared with architecture C and 0.02 points higher when compared with architecture A. ResNet18 however gives almost the same accuracy result as architecture B. Also, in trying to maximize the precision and recall of all the predictions, ResNet18 is more superior for all the classes including the most difficult class, the background noise class. Architecture B is the least robust when it comes to calculating the harmonic mean of the precision and recall of the background noise class.

From the mean test accuracy results for all classes considered, architecture B is the most accurate model for Reduced10db dataset. It however, has a moderate F1 score as compared to the rest of the architectures for the same dataset. For Reduced15db dataset, ResNet18 is the most accurate model and it has very good mean F1 score. The

less accurate model for Reduced10db is architecture C and that of Reduced15db, is architecture B.

6.1.2 Experiment 2: Binary Classification (1-versus-1)

The objective of this experiment is to determine if models can easily differentiate between two similar classes. It can be observed from the test set results that, architecture C and ResNet18 have lower F1 scores of approximately 0.60 as compared to 0.63 and 0.62 for architectures A and B respectively. These are much lower than the values obtained in experiment 1.

Even-though the F1 score for architectures A and B is the same for the test set, architecture A is better if the precision of the binary classification is prioritized. Similarly, architecture B is better for higher recall. Both models could therefore be said to be of similar relevance when used to compare similar classes or audio commands from the AED device.

6.1.3 Experiment 3: Multiclass Classification

The multiclass classification had the least test accuracy scores in all the experiments. When the background folder is discarded, there is 10 percent increase in validation and test accuracy for ResNet18 and architectures A and C. Architecture B does not gain any significant increase in accuracy in this scenario.

The best performing model for both five and six labelled multiclass classification is architecture C. The best test accuracy is 52 percent for the five labelled AED audio clips. It is 0.7 points higher than the 6-class multi classification experiment. This model architecture also has the best F1 score when compared with the rest of the models.

6.2 Proposed System

Following the experiments conducted in this work, a final proposed system for the detection of sound events during team training is shown below(Figure 6.3).

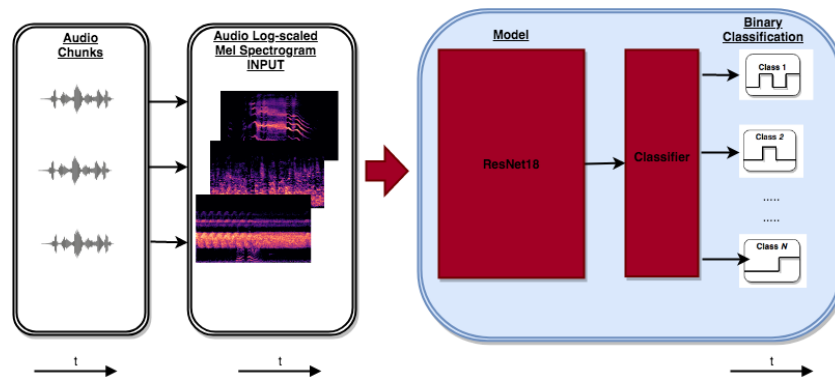


Figure 6.3: Diagram of the proposed system for sound event detection in team training. The system takes in real time audio from the team training event and divide the audio into chunks as a function of time(t). The audio chunks pass through a preprocessing stage where they are converted into log-scaled mel spectrograms. The spectrograms are then serve as an input into the model. The final predictions are interpreted as a function of time.

The proposed system takes in real time audio and the audio is split into equal chunks as a function of time(t). The audio chunks go through a preprocessing stage. This process converts the audio chunks into log-scaled mel spectrograms as described in Chapter 3. These spectrograms serve as input data to the model.

Based on the results and analysis in this thesis work, the proposed system uses binary classification using the 1-versus-others method. ResNet18 is the best performing architecture and it is more robust in classifying the AED audio events used for experimentation work in Chapter 5. The proposed system uses ResNet18 for feature extraction and a classifier for binary classification(1-versus-others) of the AED audio events. The AED audio events are predicted one by one and interpreted as a function of time(t) as shown in Figure 6.3.

6.3 Limitations

This work starts with the generation of audio mixtures with background noise. The background noise gathered are mix of recorded audio in a controlled environment [85] and polyphonic sound from the hospital and restaurant settings. This provides a wide range of background noise for model training. However, the main limitation is that, realistic audio from team training environments is needed to test the efficiency and robustness of the proposed system.

6.3.1 Volume Reduction

The experiments conducted indicate a reduction in test accuracy for all classifications and architectures when there is a reduction in loudness, measured in dBFS. Therefore, recording AED audio sound below -40 dBFS (approximately), will result in a further reduction in test accuracy.

6.4 Future Work

This section will provide some suggestions for further work and development in relation to this work.

6.4.1 Network Architectures

The use and testing of more pre-trained models could help increase the results. Apart from the pre-trained models used here, there are some more models that could be tested on this work. Also, fine-tuning the proposed models with a variety of feature learning layers could help improve the results achieved so far.

6.4.2 Realistic Data Material

The use of recorded polyphonic sound from team training events in training, validating and testing these models would be the surest way. This could help in easier expansion and implementation of this work to real world devices.

6.4.3 More Dataset per Class

In experiment 2, the accuracy for all architectures used could be improved by using more datasets per class of AED audio event. Also, data augmentation methods could help improve the results of this experiment.

7

Conclusion

The objective of this thesis was to explore classification methods for automatic sound event detection from AED devices during medical team training. Two CNN based, one CRNN and pretrained ResNet18 architectures are implemented to classify these the AED sound events. A final system is also proposed based on the the best model and method studied in this thesis.

The data was created by overlaying monophonic AED audio on monophonic and polyphonic background noise(audio) to create polyphonic audio clips. The polyphonic audio clips are then converted to log-scaled mel spectrograms and the resultant spectrograms('images'), are used as input data for the aforementioned DNN models. Two datasets are created to study the effect of reduced volume in classifying AED audio events. Each dataset is divided into training,validation and testing sets. Reduced10db dataset has 15370 spectrograms while Reduced15db consist of 23050 spectrograms.

The best classification methodology was the binary classification, using the 1-versus-other method. ResNet18 obtained the highest overall accuracy of 86.7 percent and F1 score of 0.84, for Reduced15db dataset and across all the 6 classes. The final proposed system,records audio chunks as a function of time and converts them into log-scaled mel spectrograms. The spectrograms are then used as input data into the ResNet18 model and the classified AED sound events are obtained(as output), as a function of

time. In the evaluation of the obtained results in this thesis, it is considered that the use of log-scaled mel spectrograms and DNNs for binary classification of AED sound event detection is an optimistic method worth exploring further.



Appended Codes

This section gives a short description of the codes used for the thesis. The required libraries are in each python file. The codes are accessible through GitHub [103].

A.1 Generating Audio Chunks

This code generates audio into 5 second chunks.

create_audioChunks.py is a function that generates automatically audio into 5 second chunks or clips. The file part for the audio needs to be inserted in the input path

A.2 Reduce Volume of AED Audio

This code generates AED audio of different volume in dBFS. The volume has to be set at the desired level.

reduceVolume.py generates lower volume audio samples using PyDub library from Python.

A.3 Create Polyphonic Audio Mix

Polyphonic audio is generated by overlaying the AED audio clips on the background noise/audio clips. This is done using the PyDub library on Python.

create_Audiomix.py This code is creates the polyphonic audio mix. The audio directories have to be indicated.

A.4 Create Features

Spectrograms are generated as features for the model. First MFCC and log-scaled mel spectrograms.

create_mfcc.py This file generates MFCC spectrograms using Librosa from Python.

create_logscaledmelspec.py This file generates log-scaled mel spectrograms using Librosa from Python.

A.5 Import Dataset

import_dataset.py imports the training, validation and testing data from their respective folders.

A.6 Architectures

These are the codes for the proposed models.

architectureA.py this is a code for architecture A. The code includes evaluation and prediction sections for both binary and multiclass classification.

architectureB.py this is a code for architecture B. The code includes evaluation and prediction sections for both binary and multiclass classification.

architectureC.py this is a code for architecture C. The code includes evaluation and prediction sections for both binary and multiclass classification.

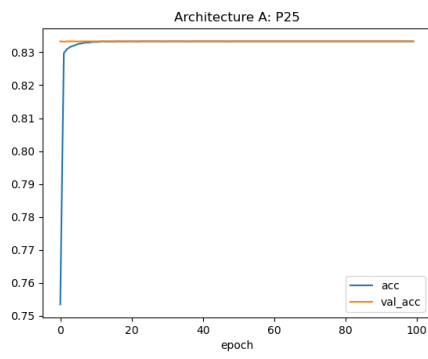
ResNet18 the source code for this model can be found on [\[104\]](#).

B

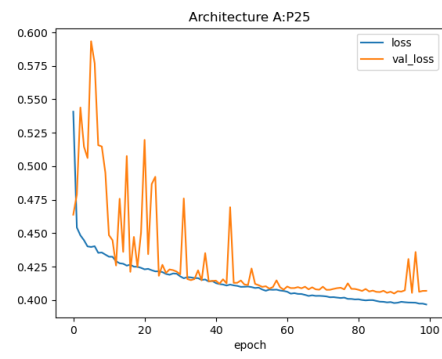
Plots for Binary Classification(1-versus-others)

B.1 Plots for Architecture A

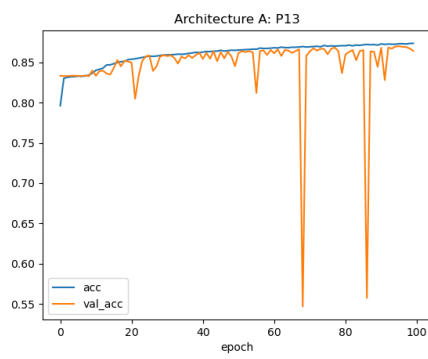
The plots show the epochs versus accuracy and loss graphs for classes P09,P13 and P25. Figure B.1 shows the plots using Reduced10db while Figure B.2 shows the results for Reduce15db.



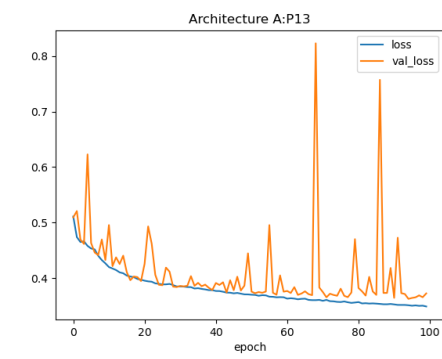
(a)



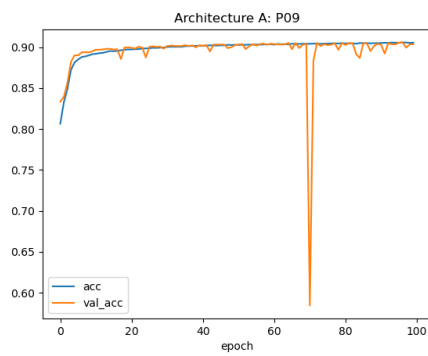
(b)



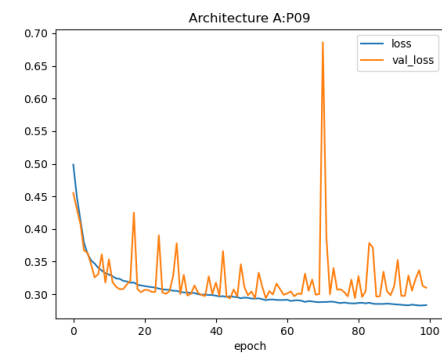
(c)



(d)



(e)



(f)

Figure B.1: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25

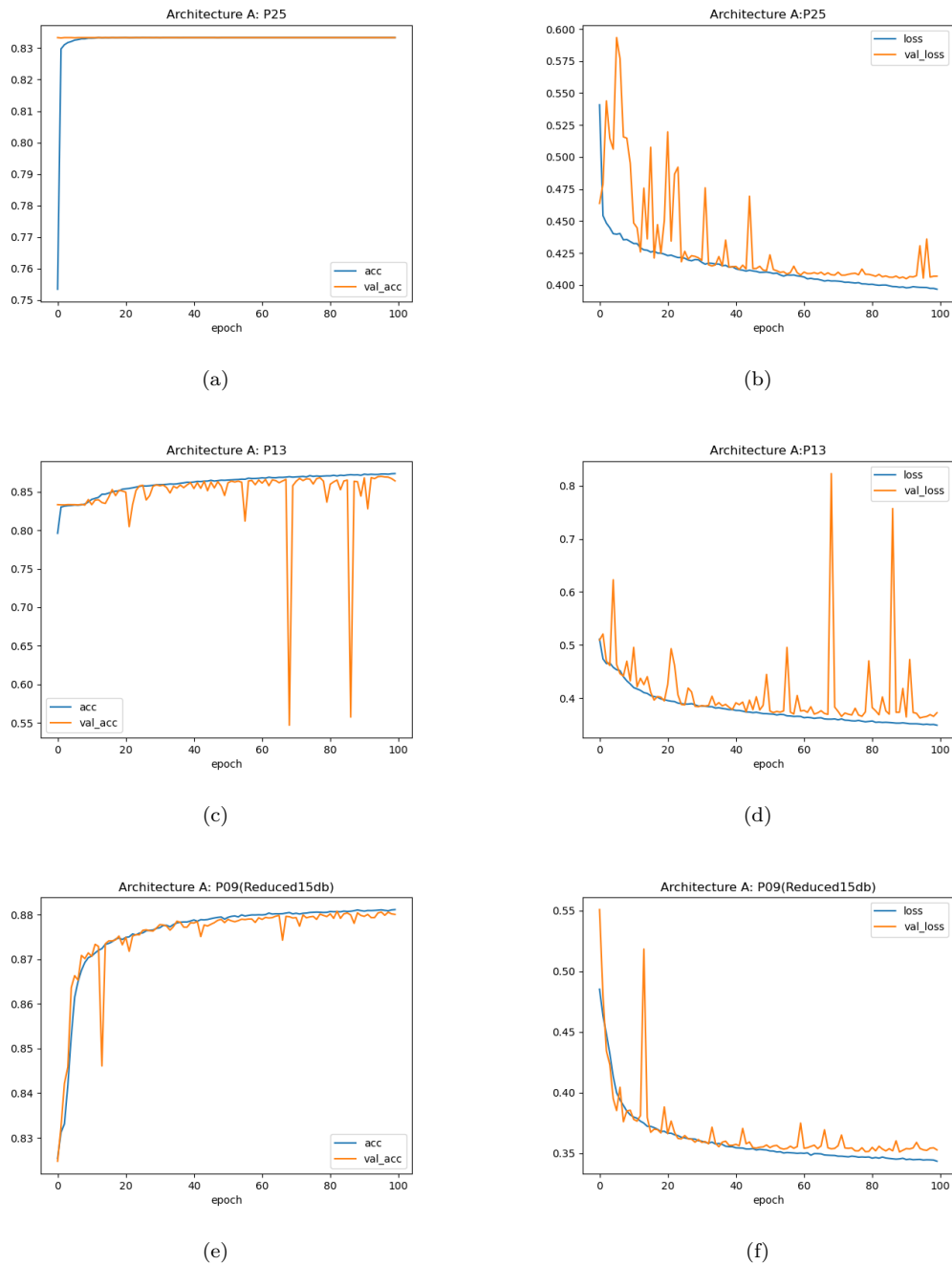
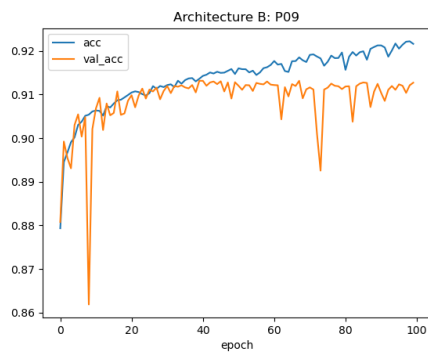


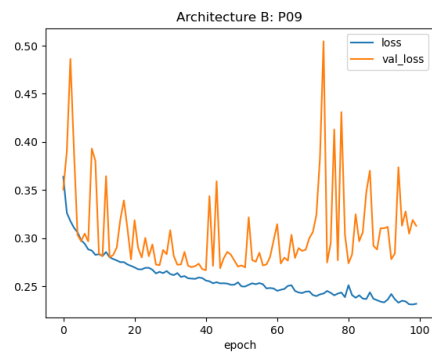
Figure B.2: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25

B.2 Plots for Architecture B

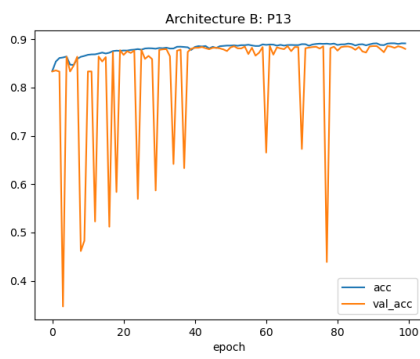
The plots show the epochs versus accuracy and loss graphs for classes P09,P13 and P25. Figure B.3 shows the plots using Reduced10db while Figure B.4 shows the results for Reduce15db.



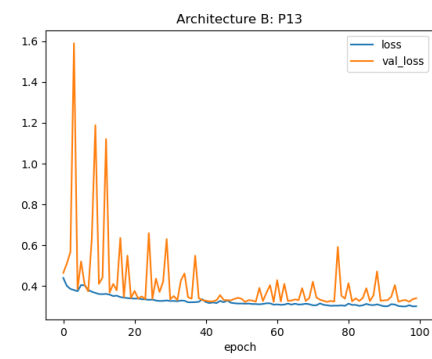
(a)



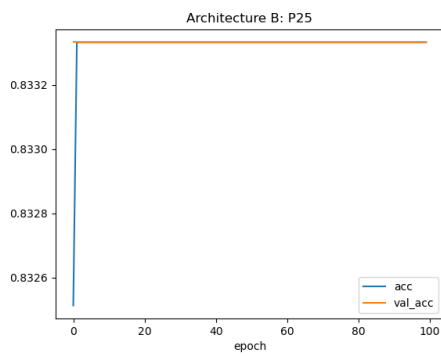
(b)



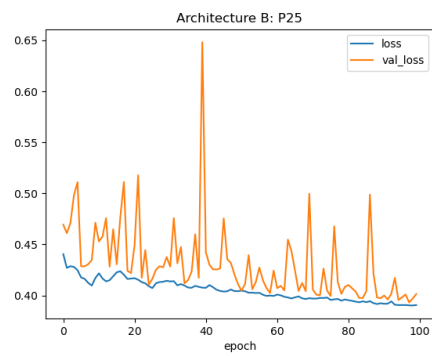
(c)



(d)

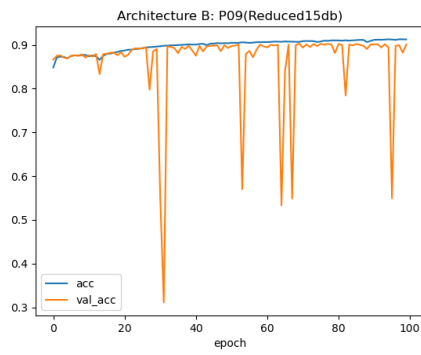


(e)

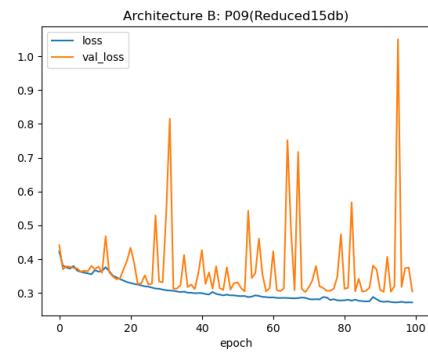


(f)

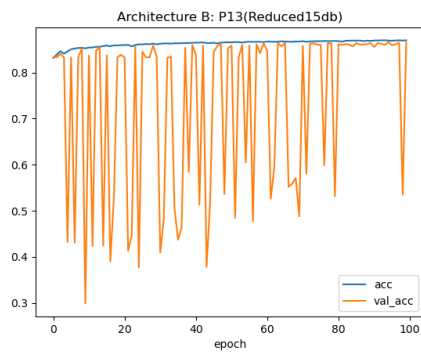
Figure B.3: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25



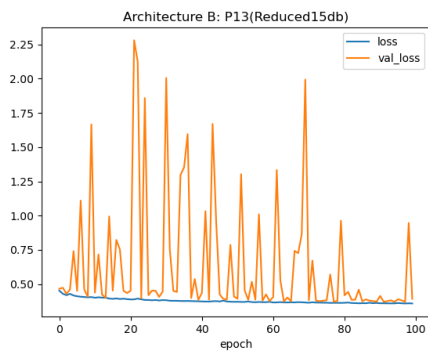
(a)



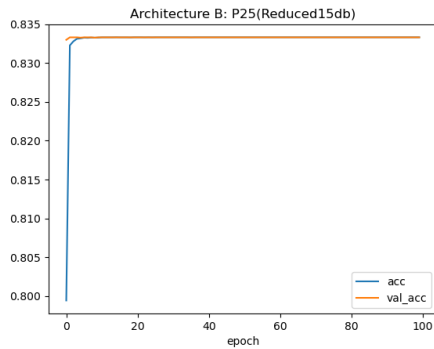
(b)



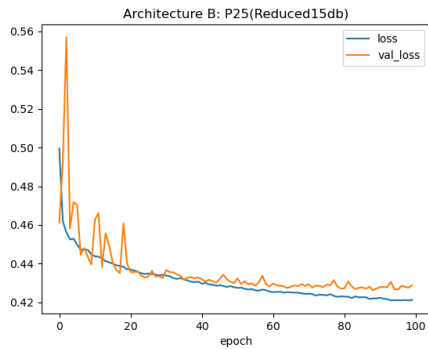
(c)



(d)



(e)

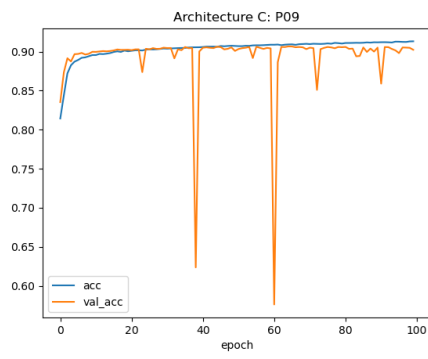


(f)

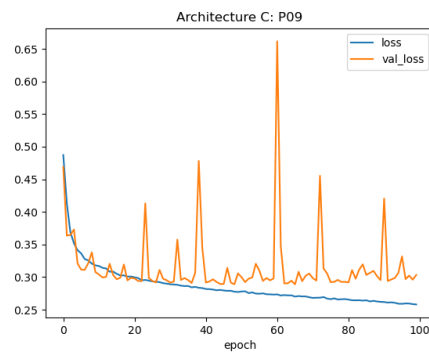
Figure B.4: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25

B.3 Plots for Architecture C

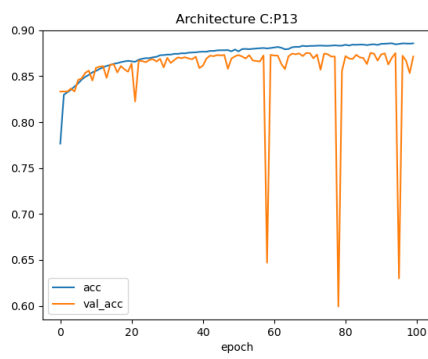
The plots show the epochs versus accuracy and loss graphs for classes P09,P13 and P25. Figure B.5 shows the plots using Reduced10db while Figure B.6 shows the results for Reduce15db.



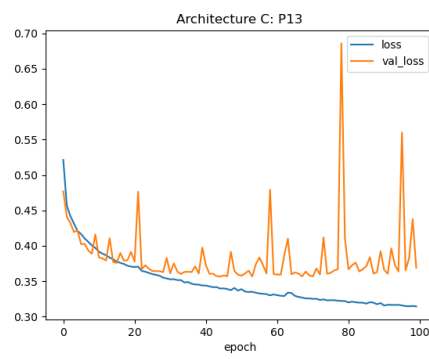
(a)



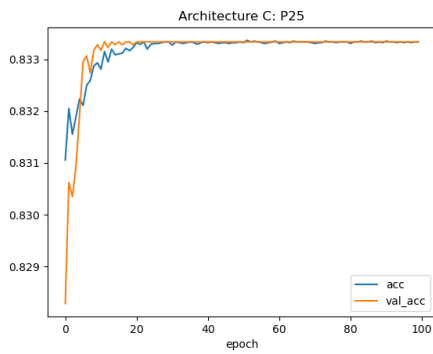
(b)



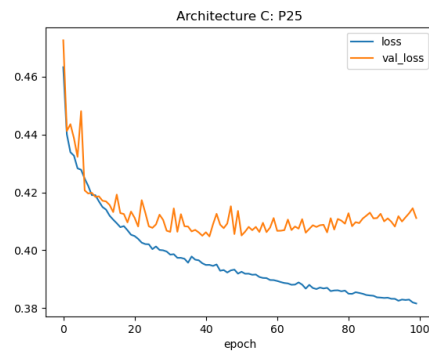
(c)



(d)



(e)



(f)

Figure B.5: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25

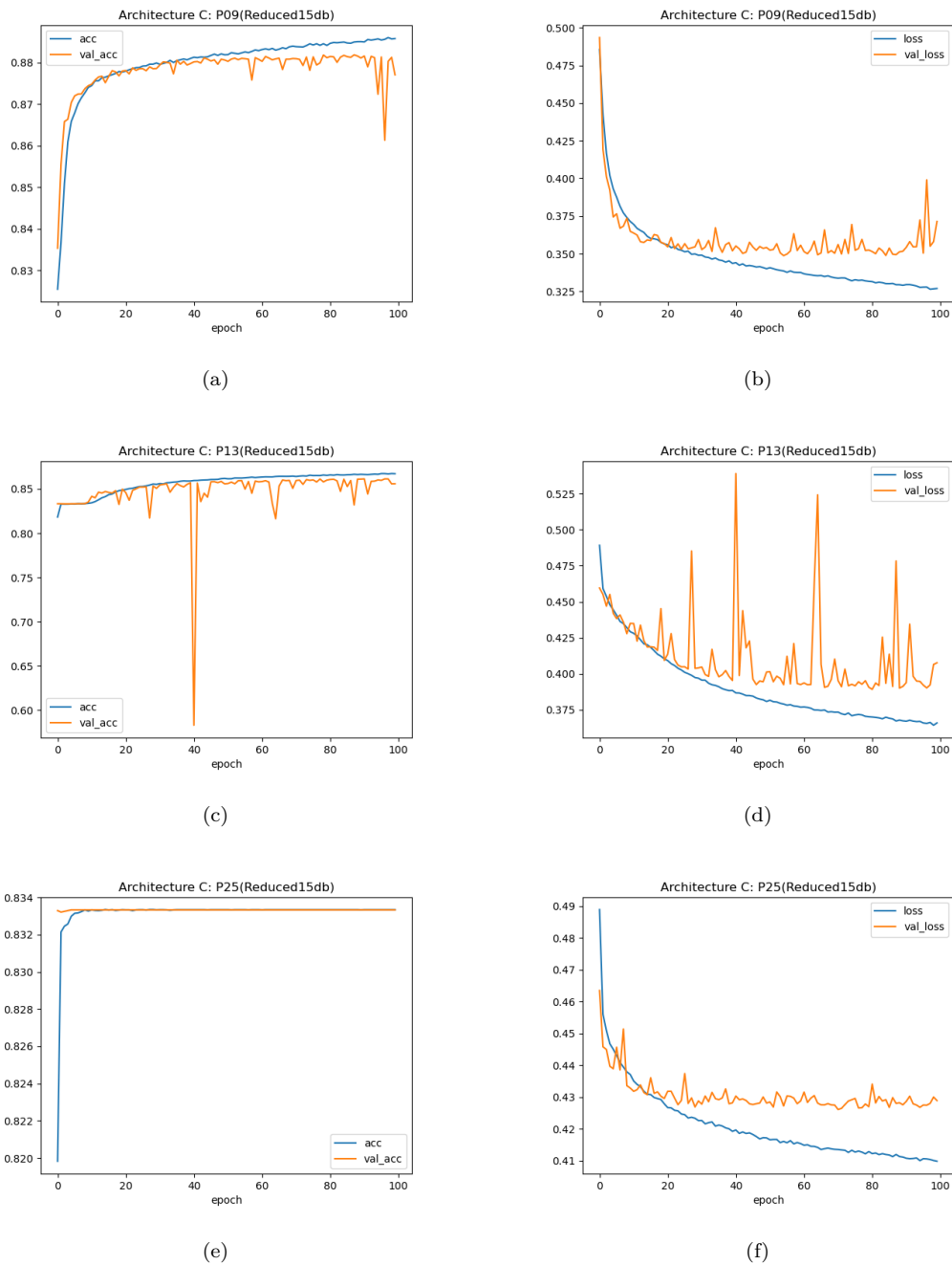
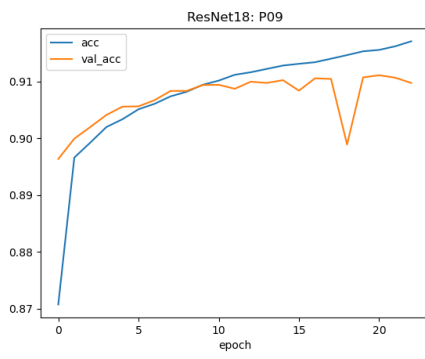


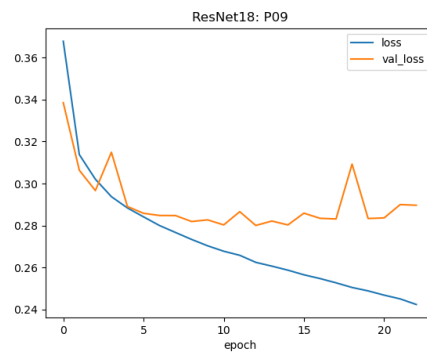
Figure B.6: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25

B.4 Plots for ResNet18

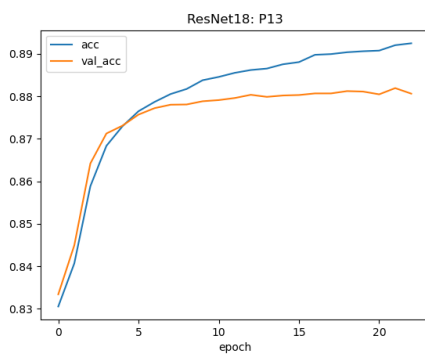
The plots show the epochs versus accuracy and loss graphs for classes P09,P13 and P25. Figure B.7 shows the plots using Reduced10db while Figure B.8 shows the results for Reduce15db.



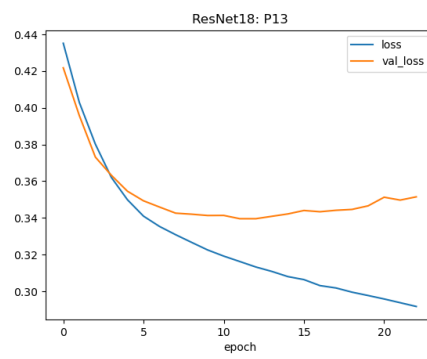
(a)



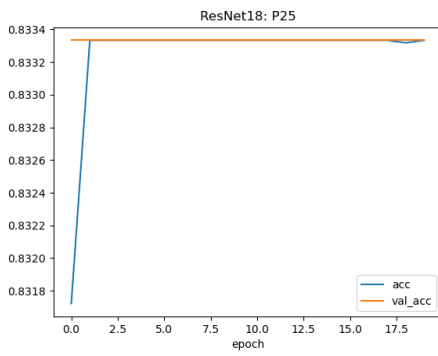
(b)



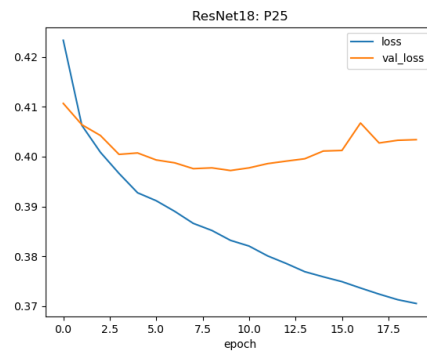
(c)



(d)

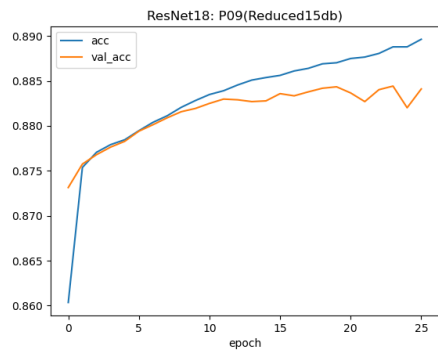


(e)

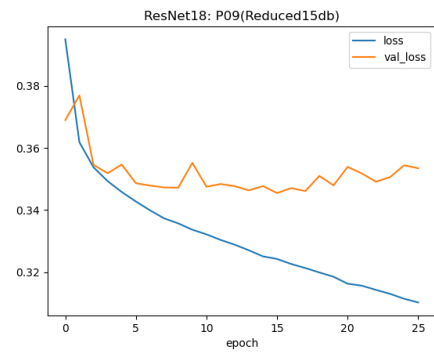


(f)

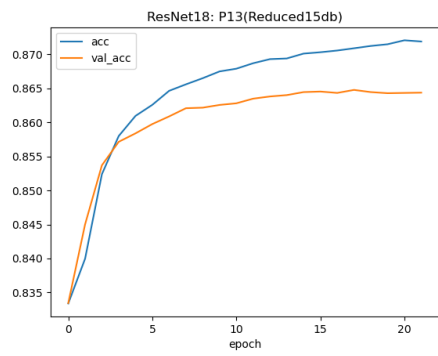
Figure B.7: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25



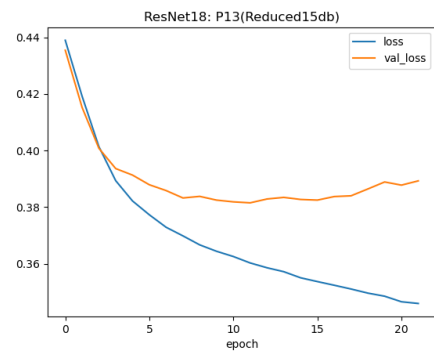
(a)



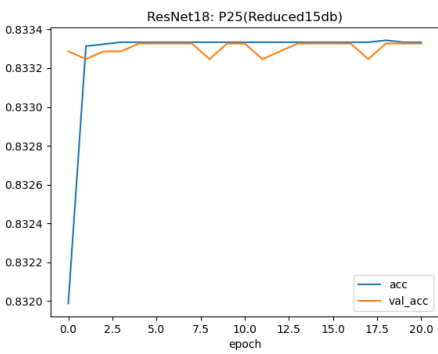
(b)



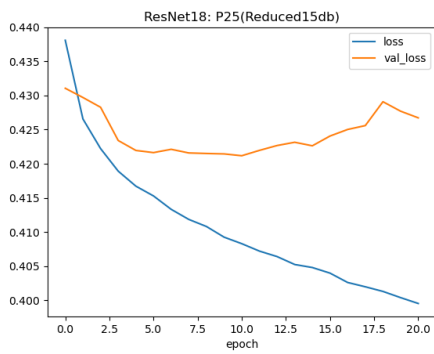
(c)



(d)



(e)



(f)

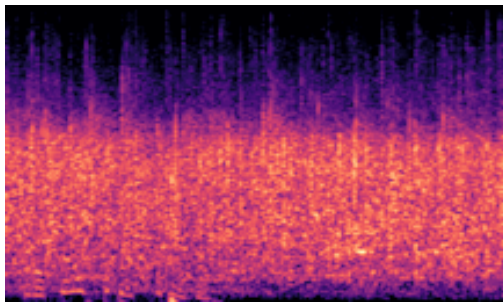
Figure B.8: Epochs versus Accuracy and Loss Graphs. The graphs show the epochs versus accuracy(left hand side) and epochs versus loss(right hand side) of the training and validation of P09, P13 and P25

C

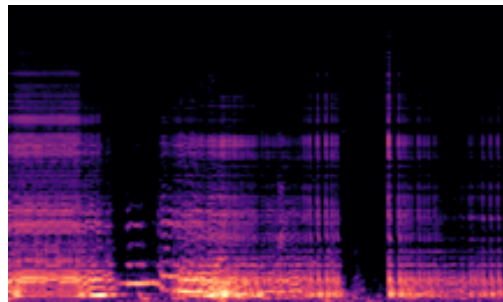
Sample Spectrograms

C.1 Log-scaled mel spectrogram samples

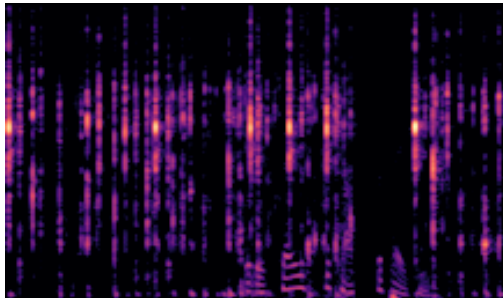
These are random samples of log-scaled mel spectrograms used for in this thesis.



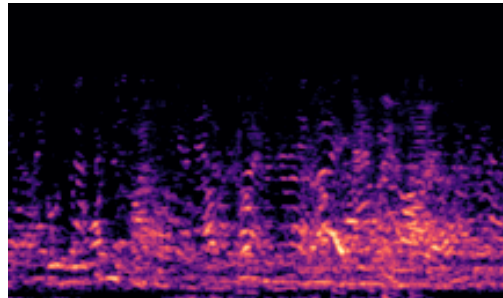
(a)



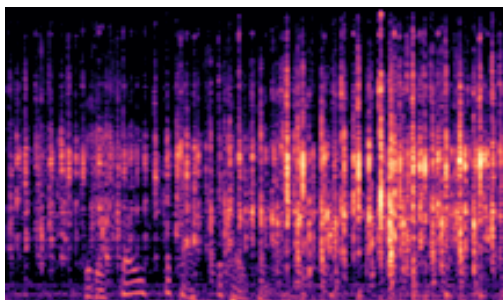
(b)



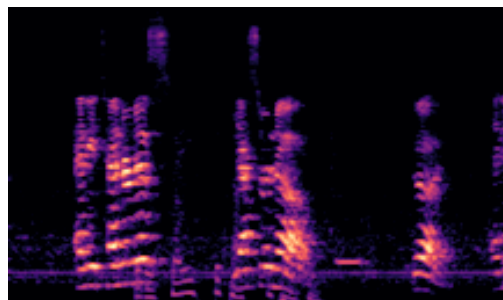
(c)



(d)



(e)



(f)

Figure C.1: Log-scaled mel spectrograms. These are examples of log-scaled mel spectrograms used for the experimentation

C.2 MFCC spectrogram samples

These are random samples of MFCC spectrograms used for in this thesis.

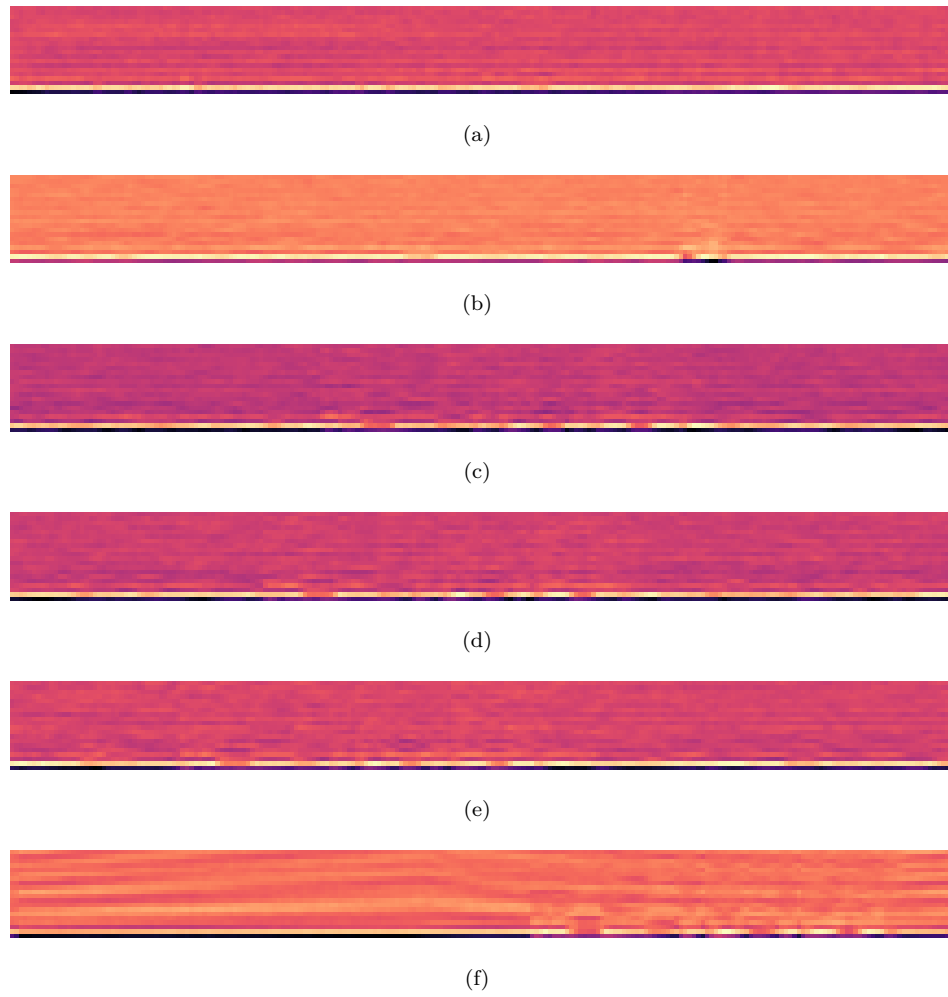


Figure C.2: MFCC spectrograms. These are examples of MFCC spectrograms used for the experimentation

Bibliography

- [1] Mandy Kubik. What is sound?, 2010. URL <https://blogs.jwpepper.com/what-is-sound/>. [Online; accessed 26-April-2021].
- [2] Aavos International. Fourier transform, 2017. URL <https://aavos.eu/glossary/fourier-transform/>. [Online; accessed 26-April-2021].
- [3] Prabhu. Understanding of convolutional neural network (cnn) — deep learning, 2018. URL <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>. [Online; accessed 28-April-2021].
- [4] Kolmakov Maksim. Using convolutional neural networks for image recognition. 2017.
- [5] Benjamin S Abella, Jason P Alvarado, Helge Myklebust, Dana P Edelson, Anne Barry, Nicholas O’Hearn, Terry L Vanden Hoek, and Lance B Becker. Quality of cardiopulmonary resuscitation during in-hospital cardiac arrest. *Jama*, 293(3): 305–310, 2005.
- [6] Mickey S Eisenberg, Bruce T Horwood, Richard O Cummins, Robin Reynolds-Haertle, and Thomas R Hearne. Cardiac arrest and resuscitation: a tale of 29 cities. *Annals of emergency medicine*, 19(2):179–186, 1990.
- [7] Ingvild Beathe Myrhaugen Tjelmeland, Kristin Alm-Kruse, Lars-Jøran Andersson, Ståle Bratland, Arne-Ketil Hafstad, Bjørn Haug, Jørund Langørgen, Alf Inge Larsen, Thomas Werner Lindner, Jan Erik Nilsen, et al. Cardiac arrest as a reportable condition: a cohort study of the first 6 years of the norwegian out-of-hospital cardiac arrest registry. *BMJ open*, 10(7):e038133, 2020.
- [8] Norway: number of deaths, by cause of death 2019 the leading cause of death in norway in 2019 was cancer, which led to around 11 thousand deaths that year. <https://www.statista.com/statistics/942953/number-of-deaths-in-norway-by-cause-of-death/>. Accessed: 2021-04-20 17:35:26.

- [9] Randi Ballangrud, Marie Louise Hall-Lord, Mona Persenius, and Birgitta Hedelin. Intensive care nurses' perceptions of simulation-based team training for building patient safety in intensive care: a descriptive qualitative study. *Intensive and Critical Care Nursing*, 30(4):179–187, 2014.
- [10] Benjamin S Abella, Dana P Edelson, Salem Kim, Elizabeth Retzer, Helge Myklebust, Anne M Barry, Nicholas O'Hearn, Terry L Vanden Hoek, and Lance B Becker. Cpr quality improvement during in-hospital cardiac arrest using a real-time audiovisual feedback system. *Resuscitation*, 73(1):54–61, 2007.
- [11] Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.
- [12] Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen. Context-dependent sound event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013(1):1–13, 2013.
- [13] 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.
- [14] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Tut database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132. IEEE, 2016.
- [15] Heikki V Huikuri, Agustin Castellanos, and Robert J Myerburg. Sudden death due to cardiac arrhythmias. *New England Journal of Medicine*, 345(20):1473–1482, 2001.
- [16] Geoffrey Arthur Rose, Henry Blackburn, RF Gillum, RJ Prineas, et al. *Cardiovascular survey methods.*, volume 56. Geneva, Switzerland; WHO, 1982.
- [17] JB Cooper and VR2004 Taqueti. A brief history of the development of mannequin simulators for clinical education and training. *Postgraduate medical journal*, 84(997):563–570, 2008.
- [18] Sallie J Weaver, Sydney M Dy, and Michael A Rosen. Team-training in healthcare: a narrative synthesis of the literature. *BMJ quality & safety*, 23(5):359–372, 2014.
- [19] Ashley M Hughes, Megan E Gregory, Dana L Joseph, Shirley C Sonesh, Shannon L Marlow, Christina N Lacerenza, Lauren E Benishek, Heidi B King, and Eduardo

- Salas. Saving lives: A meta-analysis of team training in healthcare. *Journal of Applied Psychology*, 101(9):1266, 2016.
- [20] JA Drezner. Preparing for sudden cardiac arrest—the essential role of automated external defibrillators in athletic medicine: a critical review. *British Journal of Sports Medicine*, 43(9):702–707, 2009.
- [21] Anouk P van Alem, Rob H Vrenken, Rien de Vos, Jan GP Tijssen, and Rudolph W Koster. Use of automated external defibrillator by first responders in out of hospital cardiac arrest: prospective controlled trial. *Bmj*, 327(7427):1312, 2003.
- [22] Sherry L Caffrey, Paula J Willoughby, Paul E Pepe, and Lance B Becker. Public use of automated external defibrillators. *New England journal of medicine*, 347(16):1242–1247, 2002.
- [23] Malcolm Woollard, Richard Whitfield, Anna Smith, Michael Colquhoun, Robert G Newcombe, Norman Vetter, and Douglas Chamberlain. Skill acquisition and retention in automated external defibrillator (aed) use and cpr by lay responders: a prospective study. *Resuscitation*, 60(1):17–28, 2004.
- [24] Karen Birckelbaw Kopacek, Anna Legreid Dopp, John M Dopp, Orly Vardeny, and J Jason Sims. Pharmacy students’ retention of knowledge and skills following training in automated external defibrillator use. *American journal of pharmaceutical education*, 74(6), 2010.
- [25] Ruth M Fanning and David M Gaba. The role of debriefing in simulation-based learning. *Simulation in healthcare*, 2(2):115–125, 2007.
- [26] Bhanu Prasad and SR Mahadeva Prasanna. *Speech, audio, image and biomedical signal processing using neural networks*, volume 83. Springer, 2007.
- [27] Ben Gold, Nelson Morgan, and Dan Ellis. *Speech and audio signal processing: processing and perception of speech and music*. John Wiley & Sons, 2011.
- [28] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.
- [29] E Oran Brigham. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.
- [30] GD Bergland. A guided tour of the fast fourier transform. *IEEE spectrum*, 6(7): 41–52, 1969.
- [31] William T Cochran, James W Cooley, David L Favin, Howard D Helms, Reginald A Kaenel, William W Lang, George C Maling, David E Nelson, Charles M Rader,

- and Peter D Welch. What is the fast fourier transform? *Proceedings of the IEEE*, 55(10):1664–1674, 1967.
- [32] 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.
- [33] Jont B Allen and Lawrence R Rabiner. A unified approach to short-time fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11):1558–1564, 1977.
- [34] Stanley S Stevens and John Volkman. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329–353, 1940.
- [35] Pavel Matějka, Ondřej Glembek, Ondřej Novotný, Oldřich Plchot, František Grézl, Lukáš Burget, and Jan Honza Cernocký. Analysis of dnn approaches to speaker identification. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5100–5104. IEEE, 2016.
- [36] Arun Narayanan and DeLiang Wang. Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7092–7096. IEEE, 2013.
- [37] Jianfeng Zhao, Xia Mao, and Lijiang Chen. Speech emotion recognition using deep 1d & 2d cnn lstm networks. *Biomedical Signal Processing and Control*, 47:312–323, 2019.
- [38] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 IEEE international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.
- [39] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.
- [40] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.
- [41] Ya-Ti Peng, Ching-Yung Lin, Ming-Ting Sun, and Kun-Cheng Tsai. Healthcare audio event classification using hidden markov models and hierarchical hidden markov models. In *2009 IEEE International conference on multimedia and expo*, pages 1218–1221. IEEE, 2009.

- [42] Patrice Guyot, Julien Piquier, Xavier Valero, and Francesc Alias. Two-step detection of water sound events for the diagnostic and monitoring of dementia. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2013.
- [43] Rui Cai, Lie Lu, Alan Hanjalic, Hong-Jiang Zhang, and Lian-Hong Cai. A flexible framework for key audio effects detection and auditory context inference. *IEEE Transactions on audio, speech, and language processing*, 14(3):1026–1039, 2006.
- [44] Dimitrios Giannoulis, Emmanouil Benetos, Dan Stowell, Mathias Rossignol, Mathieu Lagrange, and Mark D Plumbley. Detection and classification of acoustic scenes and events: An ieee aasp challenge. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE, 2013.
- [45] Antti J Eronen, Vesa T Peltonen, Juha T Tuomi, Anssi P Klapuri, Seppo Fagerlund, Timo Sorsa, Gaëtan Lorho, and Jyri Huopaniemi. Audio-based context recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):321–329, 2005.
- [46] 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.
- [47] Donmoon Lee, Subin Lee, Yoonchang Han, and Kyogu Lee. Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input. *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.
- [48] 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.
- [49] Anurag Kumar, Maksim Khadkevich, and Christian Fügen. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 326–330. IEEE, 2018.
- [50] Ali Imran, Iryna Posokhova, Haneya N Qureshi, Usama Masood, Muhammad Sajid Riaz, Kamran Ali, Charles N John, MD Iftikhar Hussain, and Muhammad Nabeel. Ai4covid-19: Ai enabled preliminary diagnosis for covid-19 from cough samples via an app. *Informatics in Medicine Unlocked*, 20:100378, 2020.
- [51] Ting-Wei Su, Jen-Yu Liu, and Yi-Hsuan Yang. Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks.

- In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 791–795. IEEE, 2017.
- [52] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.
- [53] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.
- [54] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [55] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- [56] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.
- [57] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer, 1995.
- [58] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.
- [59] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.
- [60] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.
- [61] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.
- [62] Josiah Collier Hoskins and DM Himmelblau. Artificial neural network models of knowledge representation in chemical engineering. *Computers & Chemical Engineering*, 12(9-10):881–890, 1988.
- [63] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010*, pages 177–186. Springer, 2010.
- [64] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

- [65] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [66] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. 2013.
- [67] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.
- [68] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.
- [69] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.
- [70] Alessandro Giusti, Dan C Cireşan, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. In *2013 IEEE International Conference on Image Processing*, pages 4034–4038. IEEE, 2013.
- [71] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [72] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
- [73] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [74] Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. *arXiv preprint arXiv:1711.00108*, 2017.
- [75] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*, 2014.
- [76] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.
- [77] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

- [78] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [79] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.
- [80] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.
- [81] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.
- [82] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.
- [83] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
- [84] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.
- [85] Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pages 1015–1018. ACM Press. ISBN 978-1-4503-3459-4. doi: 10.1145/2733373.2806390. URL <http://dl.acm.org/citation.cfm?doid=2733373.2806390>.
- [86] Jean E Burgess. Youtube. *Oxford Bibliographies Online*, 2011.
- [87] James Robert, Marc Webbie, et al. Pydub. *Github*, 2011.
- [88] Guido Van Rossum and Fred L Drake. Python library reference, 1995.
- [89] Hyungui Lim, Jeongsoo Park, 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)*, pages 80–84, 2017.
- [90] Emre Çakir and Tuomas Virtanen. End-to-end polyphonic sound event detection using convolutional recurrent neural networks with learned time-frequency representation input. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2018.

- [91] 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*, volume 8, pages 18–25. Citeseer, 2015.
- [92] Vibha Tiwari. Mfcc and its applications in speaker recognition. *International journal on emerging technologies*, 1(1):19–22, 2010.
- [93] Ivan Himawan, Michael Towsey, and Paul Roe. 3d convolution recurrent neural networks for bird sound detection. In *Proceedings of the 3rd Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 1–4. Detection and Classification of Acoustic Scenes and Events, 2018.
- [94] Christopher T Lovelace, Barry E Stein, and Mark T Wallace. An irrelevant light enhances auditory detection in humans: a psychophysical analysis of multisensory integration in stimulus detection. *Cognitive brain research*, 17(2):447–453, 2003.
- [95] Michael Lippert, Nikos K Logothetis, and Christoph Kayser. Improvement of visual contrast detection by a simultaneous sound. *Brain research*, 1173:102–109, 2007.
- [96] Jack LeBien, Ming Zhong, Marconi Campos-Cerqueira, Julian P Velez, Rahul Dodhia, Juan Lavista Ferres, and T Mitchell Aide. A pipeline for identification of bird and frog species in tropical soundscape recordings using a convolutional neural network. *Ecological Informatics*, 59:101113, 2020.
- [97] Sharath Adavanne, Giambattista Parascandolo, Pasi Pertilä, Toni Heittola, and Tuomas Virtanen. Sound event detection in multichannel audio using spatial and harmonic features. *arXiv preprint arXiv:1706.02293*, 2017.
- [98] Jordi Laguarda, Ferran Hueto, and Brian Subirana. Covid-19 artificial intelligence diagnosis using only cough recordings. *IEEE Open Journal of Engineering in Medicine and Biology*, 1:275–281, 2020.
- [99] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [100] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [101] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

-
- [102] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.
- [103] Maalidefaa Moses Tantuoyir. Thesis code, 2021. URL <https://github.com/Maalde/ThesisCodes>. [Online; accessed 15-June-2021].
- [104] Pavel Yakubovskiy, Ganesh Anand et al. Image classifier library, 2019. URL https://github.com/qubvel/classification_models. [Online; accessed 28-April-2021].