University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Data Science	Spring semester, 20..... Open / Restricted access
Writer: Prava Thapa	…… (Writer's signature)
Faculty supervisor: Øyvind Meinich-Bache External supervisor(s):	
Thesis title: Real time click detection for Cardiopulmonary Resuscitation (CPR) Training with MiniAnne	
Credits (ECTS): 30	
Key words: Sound Event Detection, Deep Neural Network, Multiclass classification, Mini Anne	Pages: ..53…… + enclosure: ..10…… Stavanger, June 15, 2022 Date/year

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

Real time click detection for Cardiopulmonary Resuscitation (CPR) Training with MiniAnne

Master's Thesis in Computer Science

by

Prava Thapa

Internal Supervisors

Øyvind Meinich-Bache

June 15, 2022

"Programming is a nice break from thinking."

Leslie Lamport

Abstract

Cardiopulmonary resuscitation (CPR) is a life-saving procedure for patients whose hearts have stopped beating. Blood is circulated to keep the heart and brain alive by pressing hard and fast on the chest of a lifeless person.

The main goal of this thesis is to use deep neural networks (DNN) to detect clicks in a compression of a raw audio signal, which can then be used in CPR feedback during team training. The Mini-Anne from Laerdal is an inflatable CPR mannequin that can be used to practice CPR techniques like chest compressions. A clicking sound is produced when the mannequin's chest is compressed deeply enough. Mini-Anne is used to record the raw audio signal, which includes 30 compressions with a down click followed by an up click for each compression. These compressions are detected in raw audio signals and separated into down clicks and up clicks to create audio sub samples. These separated down clicks and up clicks are also the two classes. Then, to add more variation to the data set, background noises are overlaid on the audio sub samples. To create another dataset, audio sub samples are converted to log-scale mel spectrograms. To create yet another data set, a third class, noise, is also introduced. As a result, four datasets are available: normal data with two classes, noise augmented data with two classes, normal data with three classes, and noise augmented data with three classes. Three models Multilayer Perceptron (MLP), long short-term memory (LSTM) networks and convolutional neural networks (CNNs) are used with multi class classification.

The best result, with an accuracy of 0.9978 and an F1-score of 0.9978, is for a CNN with a normal data set (no augmented noise) and three classes.

Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor, Øyvind Meinich-Bache, for giving me the opportunity to do this project and providing invaluable guidance throughout this thesis. His fantastic enthusiasm, vision, sincerity, and continued support have deeply inspired me. It was a great privilege and honor to work and study under his guidance. I would also like to thank him for always inspiring me, encouraging me to try new ideas, providing feedback and always being available for meetings and discussions.

To my family, for their love, continuous support, and faith during the entirety of my study and thesis. Your encouragement when the times got rough are much appreciated and duly noted. My heartfelt thanks.

To my friends, for being my biggest cheerleaders. My completion of this project could not have been accomplished without the support of my dear friend, Nissan.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter discusses the motivation for this thesis, problem definition, sound event detection and related works, and thesis outline. Figure 1.1 shows the overall framework and implementation of this thesis.

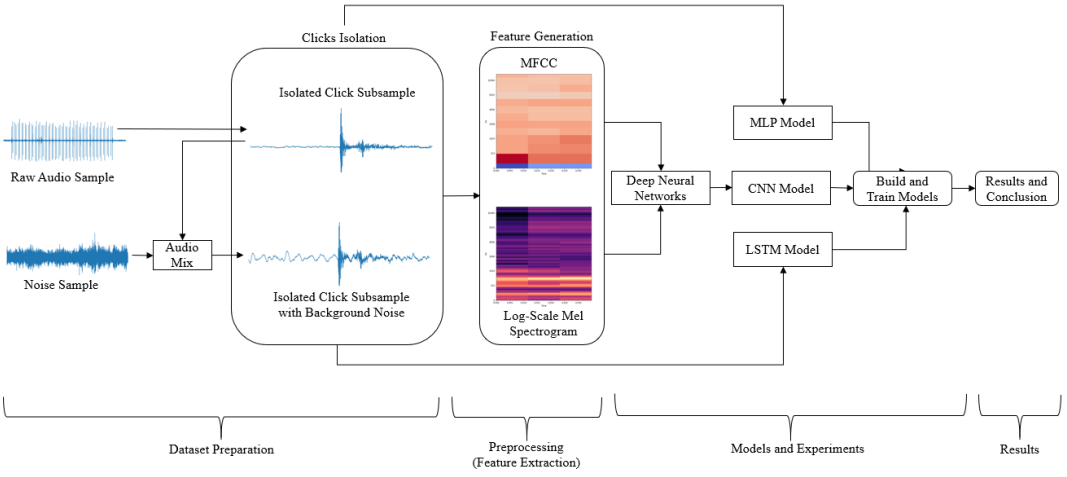

Figure 1.1: Overview of all the steps involved in this thesis

Cardiac arrest occurs when the electrical impulses that control a heartbeat become disorganized. This can also occur in persons who have never been diagnosed with heart disease or have no risk factors. In Europe alone, there are between 370,000 and 740,000 out-of-hospital cardiac arrest (OHCA) episodes, with a low average survival rate of 7.6%. OHCA is regarded as a major cause of death. Quality cardiopulmonary resuscitation (CPR) can improve survival rates until emergency medical services (EMS) arrive [7]. When CPR is performed, a person's heartbeat is simulated to restore blood flow and oxygenation to the body's system. CPR needs to be performed correctly. Blood flow might be stalled or ineffectual if CPR is performed incorrectly. CPR with properly

conducted chest compressions, cycle, compression point and ventilations has been shown to improve survival in both animal and human trials [8][9][10][11][12]. Interruptions in CPR or inability to give compressions during cardiac arrest, on the other hand, have been shown in animal tests to have a negative impact on survival [12]. Although consensus recommendations explicitly outline how CPR should be administered[13], the parameters and quality of CPR in real practice are not frequently monitored [14]. Monitoring chest compression depth, and providing a feedback would improve the quality of CPR [7] and increase the performance of chest compression[15]. When the chest compression performance improves, a quality CPR can be expected. The compression feedbacks can be useful in conducting CPR as well as in CPR training. Studies shows that CPR training helps to improve the quality of resuscitation and CPR methods during actual emergency resuscitation[16][17]. Mannequins are used in CPR training courses to plot the graphical representation of compression depth and provide real-time feedback.

1.1 Problem Definition

The following tasks must be completed in order to provide quality CPR feedback that can be used in CPR training:

- To begin, detect clicks in a compression. A complete compression would consist of a down click, then an up click
- We must separate the aforementioned clicks. Once the clicks have been separated, the goal is to detect down and up clicks separately.
- If we are successful, the next step is to increase the number of clicks by augmenting background noise and then detecting down and up clicks.
- We try to detect individual down clicks, up clicks, and noise after detecting down and up clicks in an augmented data set.
- Finally, being able to detect down and up clicks separately will aid in detecting compression. This, in turn, will aid in the monitoring of chest compression performance during CPR training.

1.2 Sound Event Detection (SED) and Related Works

Sound event detection is the identification of individual sound events in an audio, which may necessitate the estimation of onset and offset for the identified instances in the

sound event. SED has several applications. They include everything from healthcare and wildlife monitoring to audio and video content indexing and retrieval [18]. SED is a method of supervised learning in which sound events are labeled as classes. This method of sound event detection could be monophonic or polyphonic. Monophonic SED is concerned with detecting the most prominent sound events at each time interval of an output sequence, whereas the polyphonic SED is concerned with detecting overlapping sounds[44]. A benchmark has been established for datasets for polyphonic SED [19] [20].

Polyphonic sound event detection research has received a lot of attention in deep learning [20][21][22]. Recent research has focused on CNN's ability to learn features from images and make predictions based on the input data. On various publicly available datasets, CNNs are widely used for polyphonic sound event detection [23][24].

All of these methodologies and applications have produced varying results depending on the type of dataset and data preparation steps used in reaching a conclusion. MFCC and log-scaled mel spectograms are the most commonly used feature extraction method [25]. These characteristics would then be studied in Chapter 5.

1.3 Thesis Outline

Figure 1.1 depicts the overall framework and implementation of the thesis study. There are 7 chapters in total. A brief synopsis of each chapter is given below:

Chapter 1 provides a brief introduction to the thesis and related work.

Chapter 2 covers the medical background, basics of polyphonic SED, as well as the technological basis of deep neural networks and Data Augmentation.

Chapter 3 describes the dataset that was used for the thesis study. It contains description of the CPR audio clips obtained from the Laerdal Medical AS, background noise data, polyphonic audio mixing (data preparation), Feature Extraction and the manual splitting of train, test and validation set.

Chapter 4 describes the classification method employed, as well as the MLP, CNN, and LSTM models' algorithmic flow and model topologies.

Chapter 5 provides the findings of the experiments and their analysis.

Chapter 6 discusses the dataset, model performance, results, limitations and future work.

The final chapter of the thesis offers observations on the overall findings.

Chapter 2

Background

This chapter provides a brief explanation of CPR training with Mini Anne, audio signal processing, SED, and the technical background of DNNs. These explanations are useful in understanding why the proposed methods were chosen and how these solutions may be applicable.

2.1 Medical Background

This section provides a brief overview of the medical background related to the thesis work.

2.1.1 Cardiac Arrest

Cardiac arrest occurs when your heart stops beating or beats so quickly that it stops pumping blood. People usually collapse and become unresponsive during cardiac arrest. Cardiac arrest can be fatal in minutes. This is why one should call for assistance and begin CPR immediately. With immediate assistance, one's chances of survival improve. One of the most important treatments for improving cardiac arrest survival is immediate CPR. CPR is frequently used until an automatic or external defibrillator is available. CPR replaces the heart's pumping action with chest compressions. It transports small quantities of blood from the heart to the brain. However, knowing CPR guidelines and skills is insufficient. To hone their skills, medical and nursing professionals [16], including general public must practice and train on a regular basis. The Mini Anne, developed by Lærdal Medical, is the best way to learn CPR skills at home. The Mini Anne kit is an inflatable mannequin with an integrated adult/child compression clicker. This, in

conjunction with the free instructional video, is the ideal way to learn CPR from the comfort of home [1]. Figure 2.1 represents a mannequin, Mini Anne.

Figure 2.1: MiniAnne developed by Lærdal Medical for learning CPR skills at home [1]

2.1.2 Mini Anne in Training

The medical industry places a high value on the development of mannequin simulators for education, training, and research [26] [27]. Healthcare processes and patient outcomes are greatly improved through team training [28][29]. Using a mannequin in team training teaches people correct hand positions and required compression depth which leads to a successful CPR. Training medical professionals, emergency response teams, and others has also been demonstrated to improve survival rates following out-of-hospital cardiac arrest[30]. Furthermore, after detecting a reduction in their basic skill competences after 4-6 months after initial training, routine training of care workers is recommended[30] [31]. Debriefing is the process of facilitating or directing a reflection within the experimental learning cycle. The essence of the simulation experience is debriefing. There is also a growing corpus of research into the role and usefulness of objective debriefing in the learning process [27] [32]. This thesis work presents a method for recognizing click sound events from MiniAnne captured with a phone while in a team-training simulation to make the debriefing process more efficient. It has the advantage of providing insights into compression feedback, which can lead to improved results.

2.2 Audio Signal Processing

The reproduction of sound is known as audio.. A change in air pressure over time is referred to as sound. Sound is measured in volts or milli-amperes[mA] and is called an audio signal. A digital audio signal is a sampled and quantized form of an audio signal with bits per sample representation. Hearing gives information about the environment, including the qualities and location of sound sources [33]. There is a requirement for humans (auditory scene analysis) or machines to separate and classify this data. In digital audio signal processing, there are three major technological applications. Audio data compression, audio effects synthesis, and audio classification are only a few of the options. In the subsections that follow, we'll look at some audio signal processing approaches in connection to audio classification.

2.2.1 Audio Signal Properties and Characteristics

The wavelength, frequency, amplitude, velocity, and period are all significant properties that define a sound wave. The **wavelength** is the shortest distance traveled by a Wave before it begins to repeat itself. The number of cycles completed by a sound wave in one second is known as its **frequency**. The distance traveled by a sound wave per unit of time is defined as its **velocity**. The wave's **amplitude** indicates how much energy is contained in the wave. The loudness of a sound wave is defined by the highest height (vertically) achieved by the wave; the greater the magnitude, the louder the wave. The amplitude of a sound wave is defined as the magnitude of one rarefaction or compression. A sound wave's time **period** is defined as the amount of time it takes to complete one cycle. One complete rarefaction and compression in the vibration defines one wave cycle. The duration is expressed in seconds. The reciprocal of the time period is used to find the wave's frequency. The visual representation of these properties of sound wave in given in Figure 2.2.

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*20 =40 [34].

The other parameters of sound that are used in this thesis are: sample rate, bit resolution channel and volume.

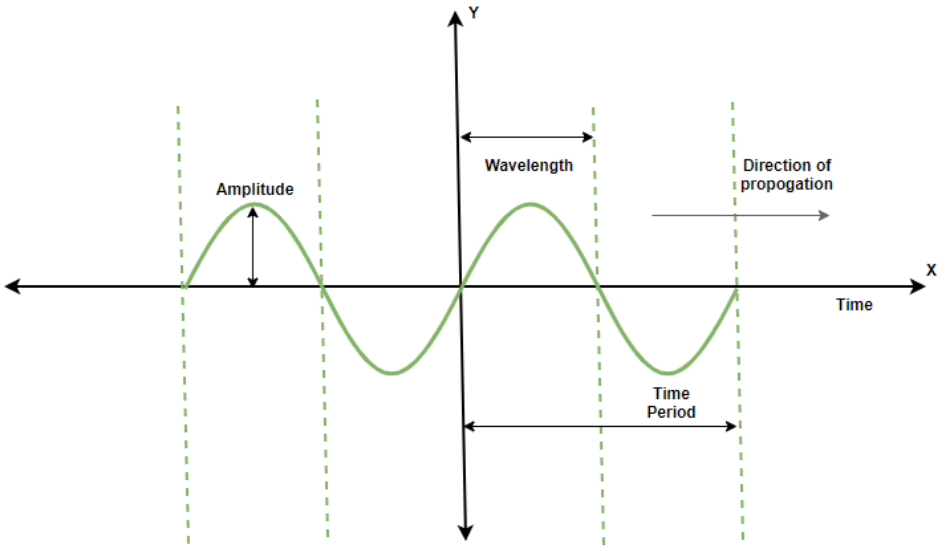

Figure 2.2: Properties of a sound wave [2]

The number of sample points per second, measured in Hertz [Hz], is known as the **sample rate**. A greater sample rate means better sound quality, but it also means more storage space. The sampling rate for phone and toy voice quality is 8 kHz, 16 kHz for speech recognition, and 44.1 kHz for CD quality. The amount of bits utilized to represent each sample point of audio signals is known as **bit resolution**. For 8-bit sound, the bit resolution is 0 - 255 or -128 - 127, while for 16-bit sound, it is -32768 - 32767. The path or communication channel by which a sound signal is carried from the player source to the speaker is referred to as a **channel**. One, two, or even more channels can be found in an audio file. Mono is commonly denoted by one channel, whereas stereo and surround sound are denoted by many channels. The **volume** of a sound is how loud or quiet the sound is.

The raw audio samples used in this thesis have following parameters:

- Sample rate = 44100 Hz.
- Bit resolution = 16 bits(2 bytes) .
- Channel = Mono.
- Average time duration = 45 secs
- Number of samples = 2054145

Figure 2.3 shows the visualization of amplitude over time for one of the click audio sample. Throughout this report, this audio signal will be used as an example.

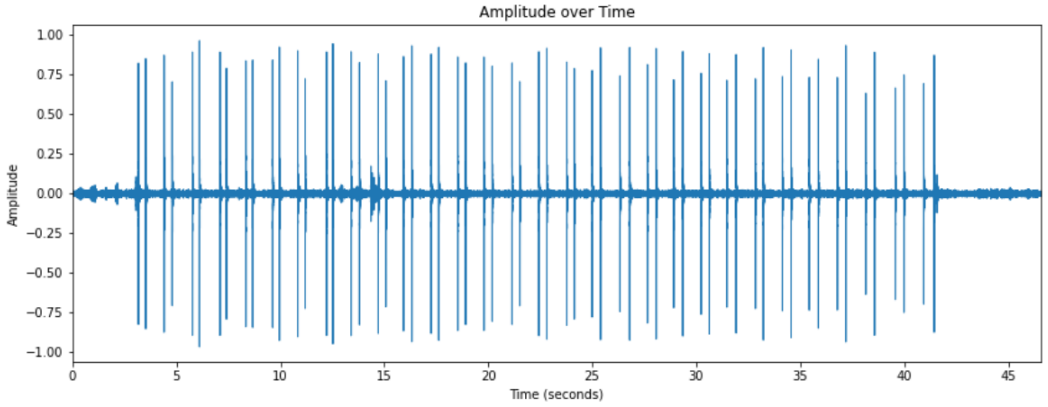

Figure 2.3: Visualization of Amplitude over Time of a raw audio sample recorded with Apple i-phone 10 in a quite environment, desired phone position, slow compression rate and good depth with click on a carpet floor

2.2.2 Audio Signal Representation

For machine-based interpretation and audio classification, the digital signal in Figure 2.3 can be further interpreted, changed, and evaluated. For the benefit of this thesis, we will look at how the Fourier Transform transforms signals from the time domain to the frequency domain.

Fourier Transform

The Fourier Transform is a method for converting a waveform (a function or signal) from the time domain to frequency domain (individual frequencies and amplitude of frequencies), which is defined by sine and cosine functions of changing frequencies. A frequency spectrum is created by decomposing a signal into a sequence of sine and cosine waves that sum up to the original signal [35]. We can see a visual representation of Fourier Transform in Figure 2.4.

A technique known as Discrete Fourier Transform (DFT) can be used to compute Fourier Transforms . Because the DFT is very expensive to compute, the FFT (Fast Fourier Transform) algorithm is used in practice, which is a more efficient way to implement the DFT. [36]. For a set of n frequency magnitudes from a vector of n input amplitudes such as $f0, f1, f2, ..., fn-2, fn-1$ and can be defined as:

$$x[K] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}} \tag{2.1}$$

The frequency domain ordinal is denoted by k, while the time domain ordinal is denoted by n. The length of the sequence to be changed is represented by N.

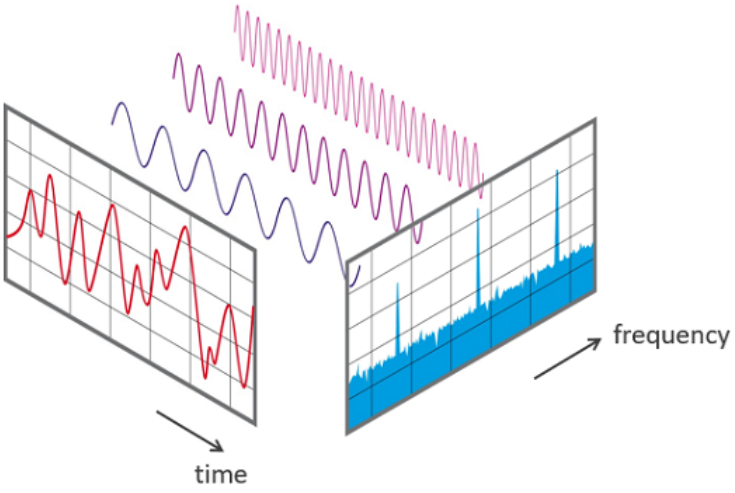

Figure 2.4: Visualization of Fourier Transform [3]

The FFT, on the other hand, will provide the overall frequency components for the entire time series of the audio signal. It will not reveal how the frequency components of the audio signal change over time.

The Short-Time Fourier Transform (STFT) algorithm can be used to obtain a more granulated view and to observe frequency fluctuations . This algorithm is another Fourier Transform variant that uses a sliding time window to divide an audio signal into smaller sections. It performs FFT on each section before combining them. As a result, it can capture frequency variations over time [37].

The waveform of Amplitude vs. Time in Figure 2.3 shows a time domain audio signal. Because an audio signal in time domain does not provide much information about the frequencies, we must represent an audio signal in frequency domain. Figure 2.5 shows the Frequency vs. Time graph.

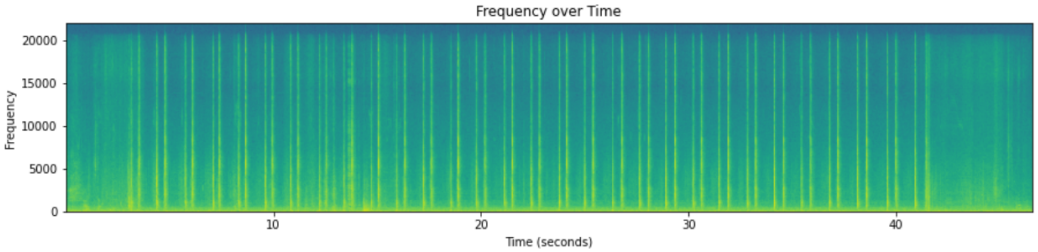

Figure 2.5: Visualization of Frequency over Time of a raw audio sample recorded with Apple i-phone 10 in a quite environment, desired phone position, slow compression rate. good depth with click on a carpet floor, n_ftt = 2048 and sampling rate = 44100

Spectrogram

Spectrograms are created by decomposing any signal into its constituent frequencies using Fast Fourier Transforms (FFT) and STFT [37] is used in this thesis to create spectrograms. A Spectrogram divides the duration of an audio signal into smaller time segments before applying the Fourier Transform to each segment to decide the frequencies stored in that segment. The Fourier Transforms for all of the segments are then combined into a single plot [38]. When a spectrogram is plotted in a graph, frequency is on the y-axis and time is on the x-axis, with different colors indicating the amplitude of each frequency which can be seen in Figure 2.6.

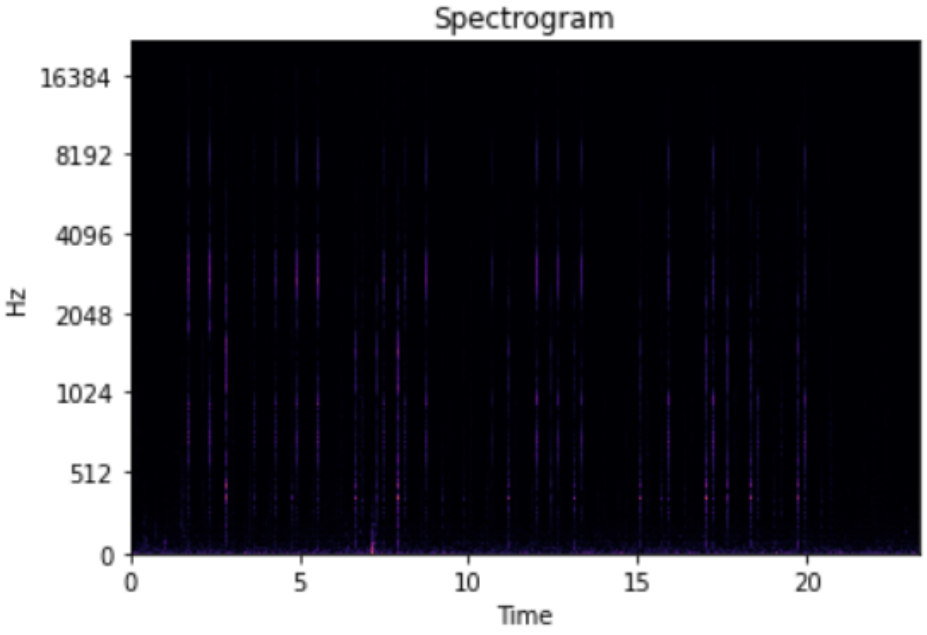

Figure 2.6: Spectrogram of a raw audio sample recorded with Apple i-phone 10 in a quite environment, desired phone position, slow compression rate. good depth with click on a carpet floor, n_ftt = 2048 and sampling rate = 44100

Figure 2.6 contains very little information. The reason for this is due to the manner humans recognize sound. The majority of what we can hear is concentrated in a small frequency and amplitudes range. This way of hearing frequencies is called *pitch*. Humans are more sensitive to variations in lower-pitched sound than to variations in higher-pitched sound. Humans hear frequencies on a logarithmic scale.

Mel Scale

The Mel Scale [39] was created in order to mimic how humans perceive frequencies through experiments with a huge number of listeners. It is a pitch scale in which listeners judge each unit to be equivalent in pitch distance from the next.

Decibel Scale

The decibel scale is how humans perceive amplitudes. According to this scale, humans hear nothing at 0 dB. As the scale goes up, there is an exponential growth in measurement units. For example, the sound intensity is doubled with an increase of 3 dB, with an increase of 10 dB, the change in sound intensity increased by a factor of 10, with an increase of 20 dB, the change in sound intensity increased by a factor of 100 and so on [40]. Like frequencies, humans hear amplitudes in a logarithmic scale too.

Because audios are dealt with realistically in this thesis, Mel Scale and Decibel Scale are used to represent our audio data.

2.2.3 Audio Features

The audio features that are used in this thesis are Log-scaled Mel Spectrograms and MFCC (Mel-Frequency Cepstral Coefficients).

Log-scaled Mel Spectrogram

A mel spectrogram is one in which the frequencies have been converted to the mel scale and the log-scaled mel spectrogram is the one where amplitudes have been converted to the mel scale to indicate colors. Mel spectrograms, rather than simple spectrograms, are preferable for deep learning models [41].

Figure 2.7 is slightly better than Figure 2.6 , but the spectrogram is still difficult to see. To see the majority of the spectrogram, the amplitude is changed to mel scale. Figure 2.8 clearly shows the difference between using a log-scaled mel spectrogram and not.

MFCC

The audio signal is first passed through a filter that emphasizes higher frequencies to obtain MFCC. The signal is then divided into small audio samples. A window is used to provide a more accurate representation of the frequency spectrum of the original signal. Then, FFT is applied to each frame to convert it from time domain to frequency domain. The log mel filter bank energies are obtained by applying the mel filter banks and taking the log of these spectrogram values. The log Mel spectrum is then converted into time domain using the Discrete Cosine Transform (DCT). The end result is a list of coefficients known as MFCCs [42] [43].

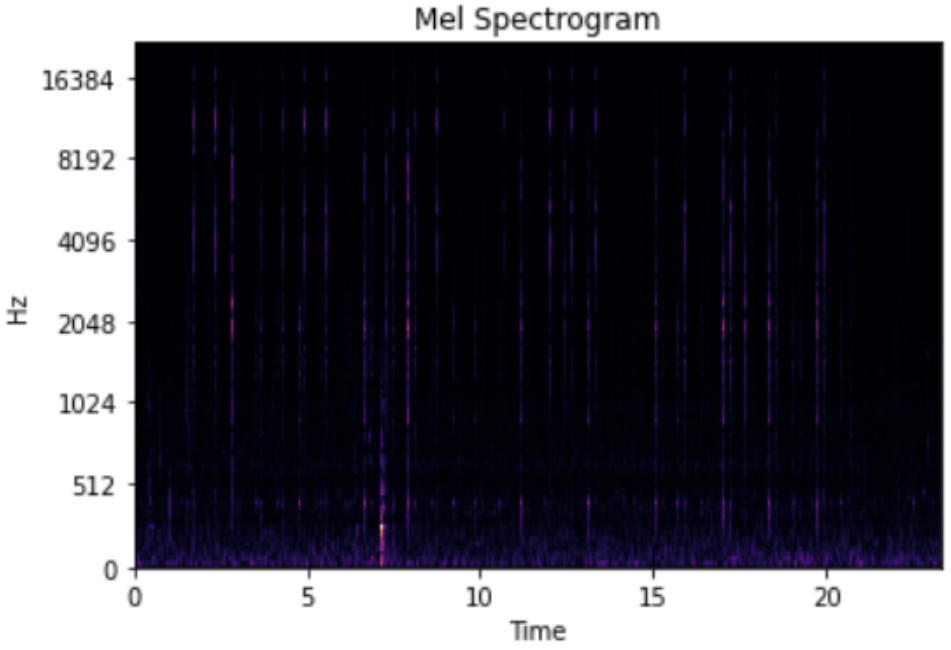

Figure 2.7: Mel Spectrogram of a raw audio sample recorded with Apple i-phone 10 in a quite environment, desired phone position, slow compression rate. good depth with click on a carpet floor, n_ftt = 2048 and sampling rate = 44100

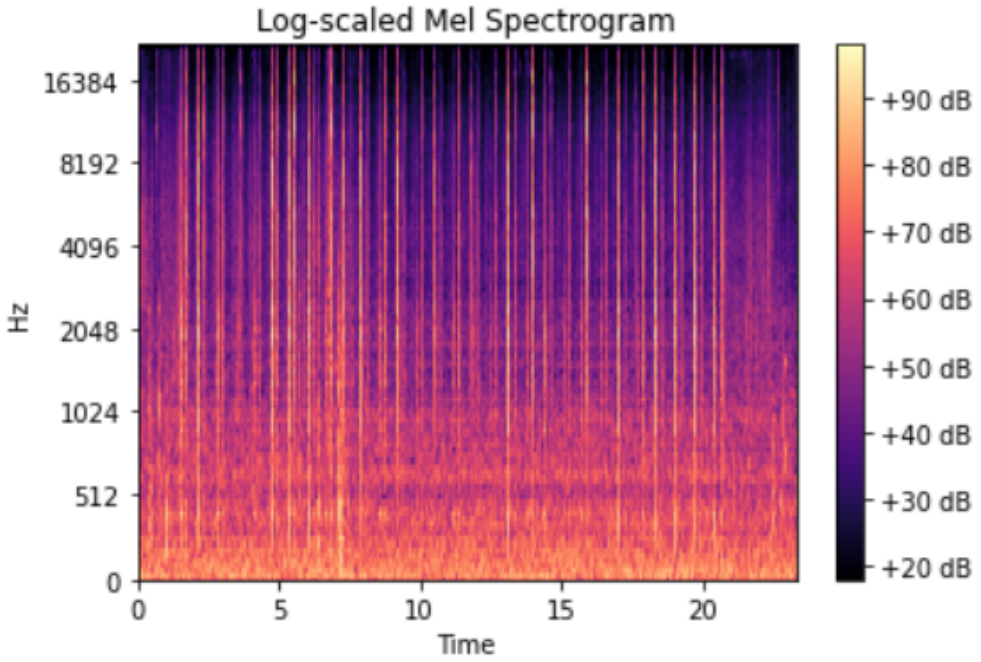

Figure 2.8: Log-scaled Mel Spectrogram of a raw audio sample recorded with Apple i-phone 10 in a quite environment, desired phone position, slow compression rate. good depth with click on a carpet floor, n_ftt = 2048 and sampling rate = 44100

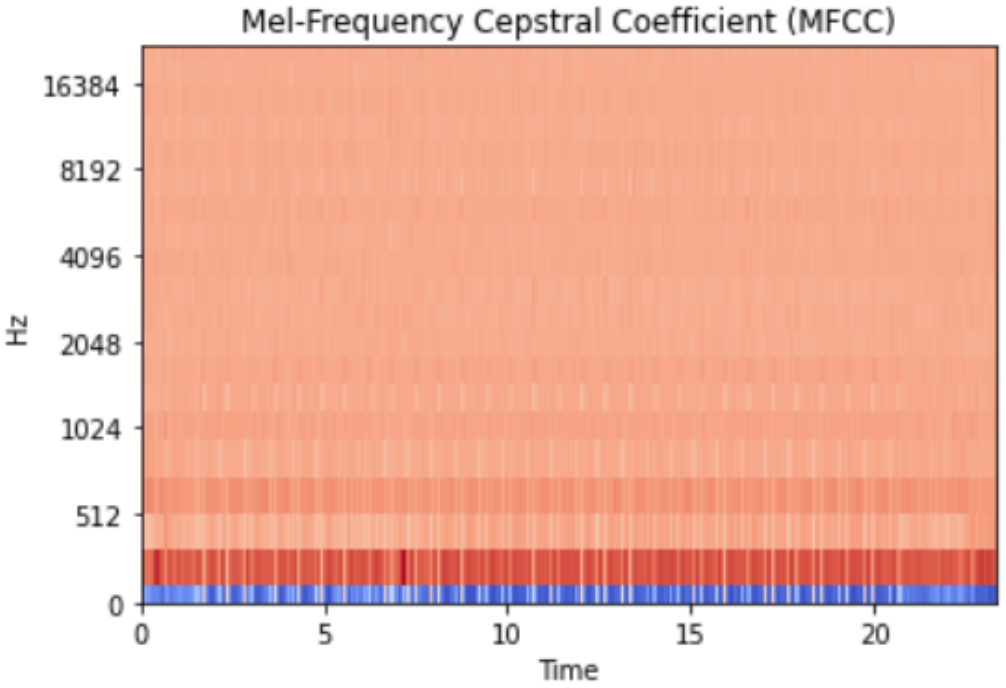

Figure 2.9: MFCC of a raw audio sample recorded with Apple i-phone 10 in a quite environment, desired phone position, slow compression rate. good depth with click on a carpet floor, n_ftt = 2048 and sampling rate = 44100

2.3 Deep Neural Networks

Deep neural networks are differentiated from single-hidden-layer neural networks by their depth, or the number of node layers through which data must transfer in a multi step pattern recognition process. Each layer of nodes in a deep neural network trains on a different set of features based on the output of the previous layer. Because nodes aggregate and recombine features from previous layers, the nodes can recognize more complex features as you progress through the neural net. Unlike most traditional machine-learning algorithms, deep neural networks extract features automatically and without human intervention.

Three deep neural networks, Multi layer perceptron (MLP), Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) are used in this thesis.

2.3.1 Multi layer perceptron (MLP)

A multi layer perceptron (MLP) is a feed forward neural network supplement. As shown in Figure 2.10, it is made up of three layers: the input layer, the output layer, and the hidden layer. The input signal to be processed is received by the input layer. The output layer is responsible for tasks such as prediction and classification. The true computational

engine of the MLP is an arbitrary number of hidden layers placed between the input and output layers. In an MLP, data flows from input to output layer in the forward direction, similar to a feed forward network. The back propagation learning algorithm is used to train the neurons in the MLP. MLPs are intended to represent any continuous function and to solve problems that cannot be solved linearly [4]. Pattern classification, recognition, prediction, and approximation are the most common applications of MLP.

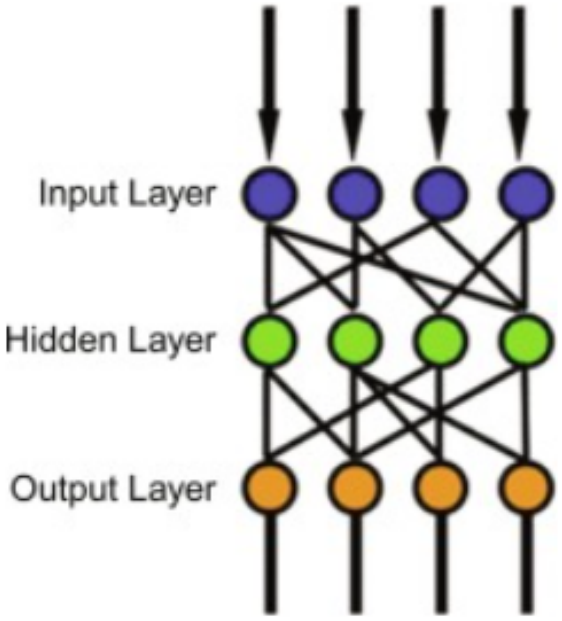

Figure 2.10: Common architecture of MLP with Input, Hidden and Output Layer in MLP [4]

2.3.2 Long Short-Term Memory (LSTM)

The LSTM network (Long Short-Term Memory) is a type of RNN (Recurrent Neural Network) that is commonly used to slove prediction problems using sequential data. LSTM, like any other neural network, has layers that help it learn and recognize patterns for improved performance. The underlining operation of LSTM can be thought of as retaining the useful information and dismissing the information that is not paticularly necessary for further prediction [44].

2.3.3 Convolutional Neural Network (CNN)

Convolutional Neural Networks mimic how humans perceive their surroundings with their eyes. Whenever people see an image, they automatically divide it into many smaller

sub-images and analyze each one individually. They process and perceive the image by assembling these sub-images [5].

CNN employs a *convolutional layer* as shown in Figure 2.11, where filter needs to be defined to determine the size of the image in interest. Then in order to know how close the number of pixels are to each other, a step length has to be defined .Convolutional layer reduces image's dimension.

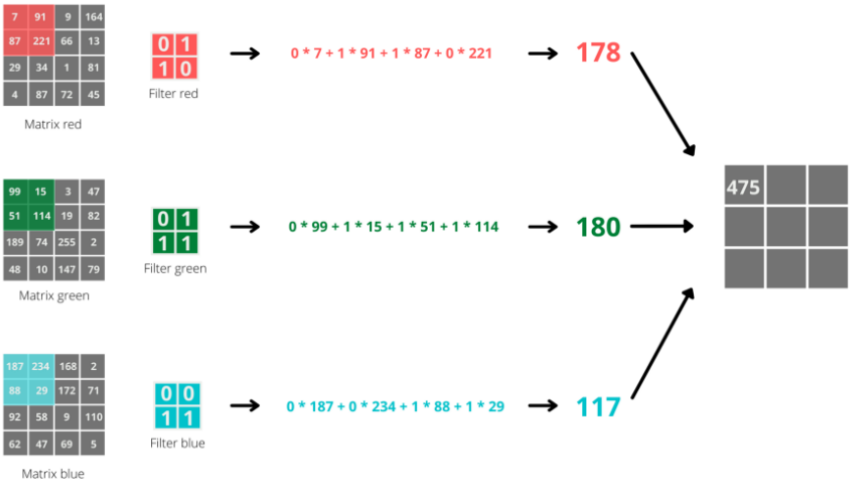

Figure 2.11: CNN Convolutional Layer [5]

Secondly, depending on the application, a *pooling layer* as presented in Figure 2.12 in CNN takes the average or maximum value of the result, which preserves small features in a few pixels that are critical for the task solution.

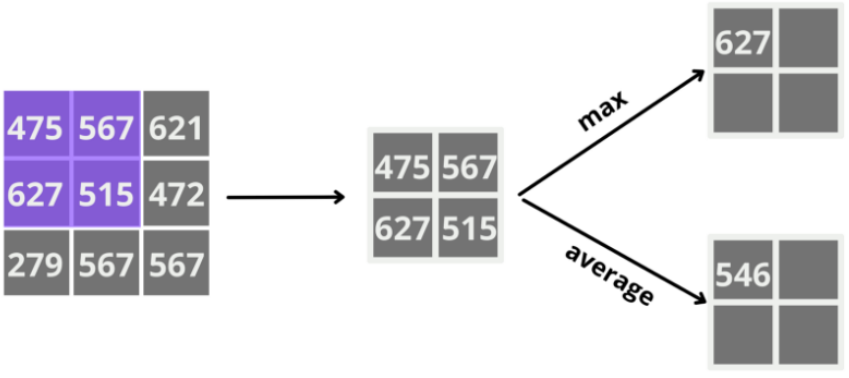

Figure 2.12: CNN Pooling Layer [5]

The *Dropout layer*[45], which helps prevent overfitting, sets input units to 0 at random, with a frequency rate at each step during training time. Inputs that are not set to 0 are upscaled by 1/(1 - rate) so that the sum of all inputs remains constant.

The *Flatten layer*[46] reshapes the input dimension to have a shape equal to the number of elements in the input. This is equivalent to creating a 1d-array of elements.

Finally, a *fully connected layer*[5] generates a neuron for each entry in the smaller matrix and connects it to all neurons in the next layer, resulting in considerably very few dimensions and requiring fewer training resources.

2.3.4 Hyperparameter Tuning

The process of selecting a set of optimal hyperparameters for a learning algorithm is known as hyperparameter tuning. A hyperparameter is a parameter whose value governs the learning process. This entails experimenting with various parameters and selecting the best one. Although it appears to be computationally expensive, our models will perform better with better parameters. Peak threshold, batch size, learning rate, input size, filter size, loss function, kernel initializer, bias initializer, dropout, and other parameters are all hypertuned in this thesis.

2.3.5 Performance Matrices

Performance metrics, also known as evaluation metrics, are metrics used to assess the quality of deep learning models. They assess the model's overall performance by measuring how well it performs on unseen data before deploying it in production. Confusion metrics is used to measure the performance of models for two class and three class data sets which is further explained in Chapter 3.

Confusion Metrix

A confusion matrix is a tabular summary of a classifier's correct and incorrect predictions. It is used to assess classification model's effectiveness and evaluate model's performance by computing performance metrics such as accuracy, precision, recall, and F1-score.

In the figure 2.13, the basic terms are defined as:

- **True positives (TP)** are the cases where we yes (this is a down click) and it was a down click.
- **True negatives (TN)** are the cases where we predicted no (this is not a down click) and it was not a down click, rather an up click.

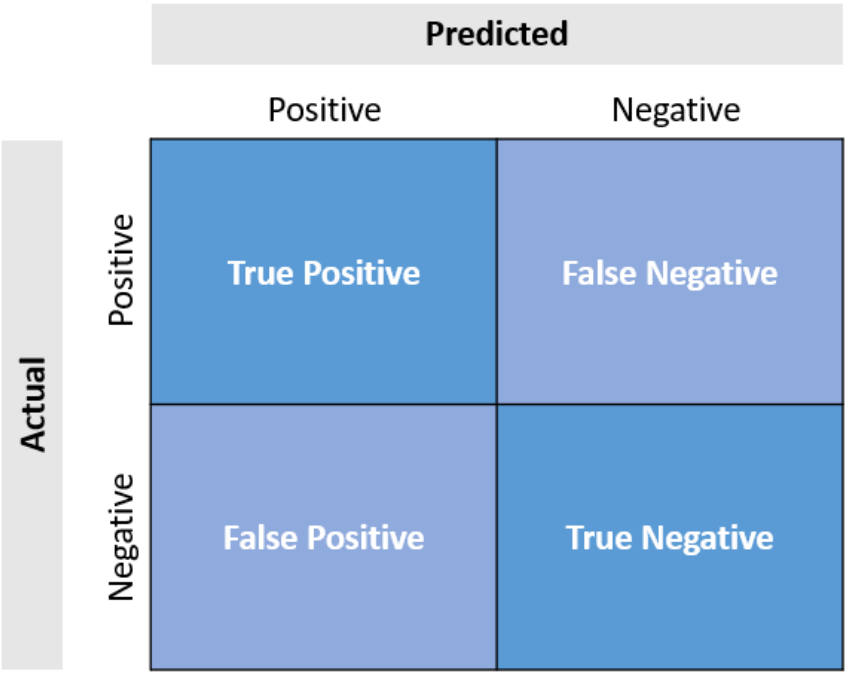

Figure 2.13: Confusion Matrix [6]

- **False positives (FP)** are the cases where we predicted yes (this is a down click), but it was not a down click. This is also known as *Type I error*.
- **False Negatives (FN)** are the cases where we predicted no (this is not a down click), but it was actually a down click. This is also known as *Type II error*.

Accuracy

Accuracy tells us how frequently the classifier is correct and is defined as:

$$Total = TP + TN + FP + FN$$

$$accuracy = \frac{TP + TN}{Total}$$

Precision

Precision indicates how often the classifier is correct when the prediction is yes and is given by:

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

Recall

Recall shows how often the classifier predicts yes and when it is actually yes. We can define recall as followed:

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

F1 Score

F1 Score computes the final score by taking into account both the precision and recall of the test; it is a measure of a test's accuracy. F1 score is given by:

$$F1score = 2\frac{precison * recall}{precision + recall}$$

2.4 Data Augmentation

In data analysis, data augmentation techniques are used to add slightly modified copies of previously existing data or newly created synthetic data from previously existing data to increase the amount of data. When we train a model, it acts as a regularizer and helps to reduce overfitting [47]. The augemntation process used in this thesis can be seen in Figure 2.14

For machine learning classification, synthetic data augmentation is critical. In signal processing problems, data scarcity is noticeable [48]. Wang et al. investigated the use of Deep Convolutional Neural Networks for EEG-Based Emotion Recognition, and the results show that emotion recognition improved when data augmentation was used [49]. One common method is to generate synthetic signals by rearranging real-world data components. According to current research, relatively simple techniques can have a significant impact. For example, Freer [50] discovered that adding noise to collected data to generate additional data enhanced the learning capacity of many models that would otherwise perform poorly.

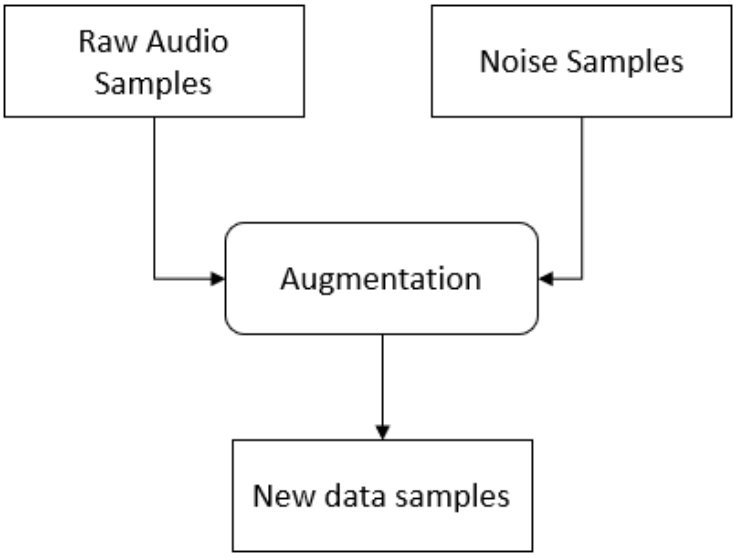

Figure 2.14: Augemntation preocess where new data samples are produced by adding noise samples to raw audio samples.

Chapter 3

Data set

This chapter discusses the data set, as well as its preparation and pre-processing. It is critical to understand how the dataset is used, created, and organized to solve our problem. The section highlighted in blue in Figure 3.1 corresponds to this chapter.

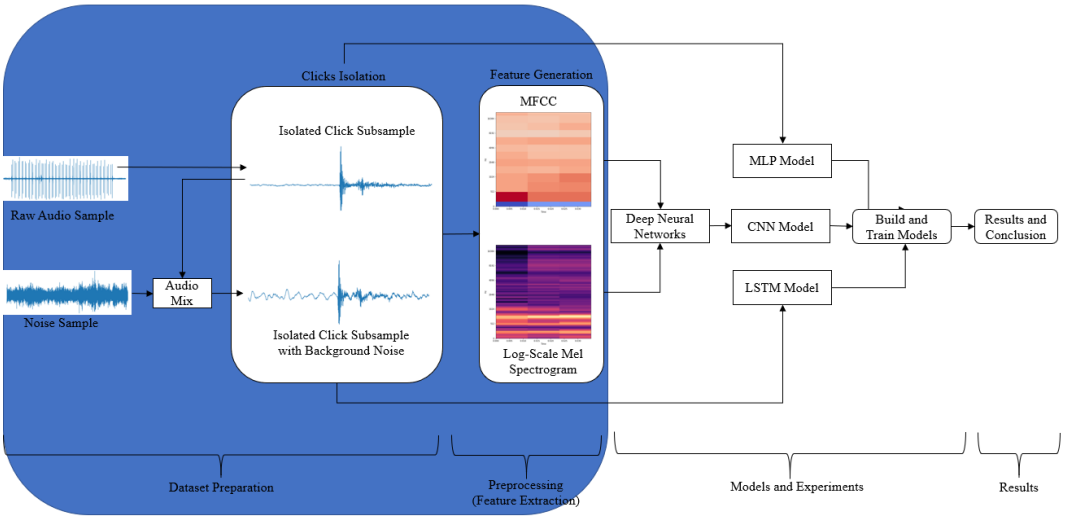

Figure 3.1: Overview of all the steps involved in this thesis highlighting data preparation and data pre-processing.

3.1 Overview of Data set

This thesis makes use of two data sets: raw clicks audio samples provided by Laerdal Medical and background noise samples. To create the final data set,

1. We first isolate the down and up clicks from the raw clicks audio samples and use as one of the data sets. It is divided into two classes: down clicks and up clicks.

2. We then combine background noise samples with isolated click sub-samples to create a new dataset. This has two classes as well.

3. We then extend 1 with a third class and use it as one of the data sets.

4. Finally, we extend 2 with a third class and use it as one of the data sets.

3.1.1 CPR Audio Data set

The CPR Audio data set was collected and provided by Laerdal Medical. This CPR audio data set consists of 40 different audio files that are recorded in different settings and variables which can be seen in Table 3.1 and Table 3.2.

Seven of the eight audio files were chosen for our work. The audio file from Test 7 is disregarded because no clicks were detected in it. These audio files are all about 45 seconds long, monophonic (1 channel), have a sample rate of 44100 Hz and 16 bits per sample.

Table 3.1: Settings and variables in which the audio data samples were recorded for Test1, Test 2, Test 3 and Test 4

Setting	Variables	Test 1	Test 2	Test 3	Test 4
	No.of compressions	30	30	30	30
Noise	Quiet	X	X	X	X
	Talking Surroundings	-	-	-	-
	Computer audio	-	-	-	-
Phone Position	Phone position 1 (desired)	X	X	X	-
	Phone Position 2 (30 cm away)	-	-	-	X
Compression Rate	Slow rate	X	-	-	-
	Normal Rate	-	X	-	X
	Fast Rate	-	-	X	-
Compression depth	Good depth with click	X	X	X	X
	Shallow depth without click	-	-	-	-
Floor Type	Carpet	X	X	X	X
	Hard floor	-	-	-	-
Phone Type	Phone Type	1-2	1-2	1-2	1-2
	Phone Type	3	3	3	3
	Phone Type	4-5	4-5	4-5	4-5

3.1.2 Background Noise Data set

The background data set is adapted from the background dataset used by MM. Tantuoyir in his master thesis [27]. The entire data set was not utilized. 3399 noise data samples of length 5 s, varying dBFS, sample rate 44100 Hz, single channelled, and compressed at

Table 3.2: Settings and variables in which the audio data samples were recorded for Test 5, Test 6, Test 7 and Test 8

Setting	Variables	Test 5	Test 6	Test 7	Test 8
	No.of compressions	30	30	30	30
Noise	Quiet	-	-	X	X
	Talking Surroundings	X	X	-	-
	Computer audio	-	X	-	-
Phone Position	Phone position 1 (desired)	X	X	X	X
	Phone Position 2 (30 cm away)	-	-	-	-
Compression Rate	Slow rate	-	-	-	-
	Normal Rate	X	X	X	X
	Fast Rate	-	-	-	-
Compression depth	Good depth with click	X	X	-	X
	Shallow depth without click	-	-	X	-
Floor Type	Carpet	X	X	X	-
	Hard floor	-	-	-	X
Phone Type	Phone Type	1-2	1-2	1-2	1-2
	Phone Type	3	3	3	3
	Phone Type	4-5	4-5	4-5	4-5

164 kbits per sample are used. Table 3.3 contains a summary of the background data set used in this thesis.

Table 3.3: Types of Noise audio samples and their count

Type of Noise Audio	No. of Noise Audio Samples
High Noise	1193
High Noise Background	1169
Low Noise	1037

3.2 Data Preparation

This section explains how individual down and up clicks were extracted from raw audio samples, how the data set was expanded by adding noise, and how the audio data was grouped.

3.2.1 Clicks Detection

A complete compression is made up of a down click and an up click. As each of the raw audio samples have 30 compression, there 30 down clicks and 30 up clicks in each raw audio sample. In order to be able to detect a compression, we must first detect down

clicks and up clicks within that compression. This is accomplished by calling *find_peaks* from *SciPy* [51], and the code for detecting clicks is available in Appendix A.1.

The raw audio samples are first loaded and converted into a 1D array. For all raw audio samples, a clicks threshold, the vertical distance to its neighboring clicks, is defined based on the length of clicks' vertical distances. This aids in the removal of non-clicking points in raw audio samples. Then, by comparing neighboring values, all local maxima or peaks (clicks) are found. A local maximum is any sample with two direct neighbors with smaller amplitudes. This method locates these peaks (rather than just their values) and, ideally, finds the true inter-sample peak rather than just the index with the highest value. Five audio samples from Test 7 are disregarded out of 40 because no clicks were detected. There were also some false positives that were ignored and were not used in this thesis. As a result, we have 26 audio files that have perfect click detection. The peaks in the audio signal are depicted in the Figure 3.2

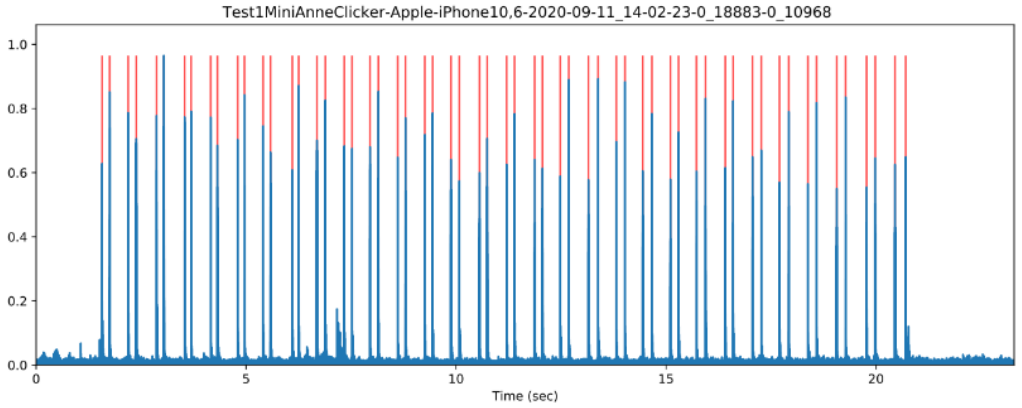

Figure 3.2: Visualization of peaks in a raw audio sample recorded with Apple i-phone 10 in a quite environment, desired phone position, slow compression rate and good depth with click on a carpet floor

3.2.2 Clicks Isolation

After locating the clicks, we determine how far apart these clicks are. Then we compute the median of all the distances. We define a window size to separate down and up clicks using this median. Because a down click always comes first, we apply a simple logic that classifies all peaks in odd positions as down clicks and the rest as up clicks. Appendix A.2 contains the code for click isolation. Figure 3.3 shows one example of how an isolated click appears.

These isolated click audio sub-samples used in this thesis have following parameters:

- Sample rate = 44100 Hz.

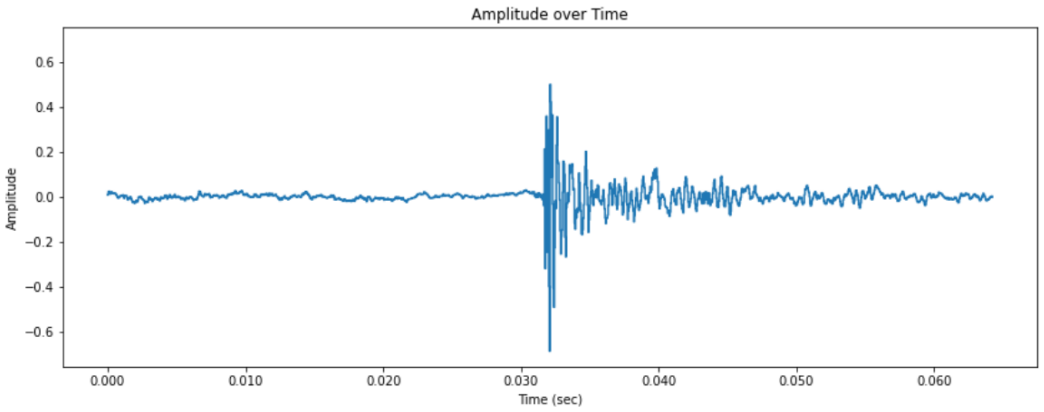

Figure 3.3: Visualization of isolated down click sub-sample in Apple i-phone 10 in a quite environment, desired phone position, slow compression rate and good depth with click on a carpet floor

- Bit resolution = 16 bits(2 bytes) .
- Channel = Mono.
- Average time duration = 64 milliseconds (ms)
- Number of samples = 2833

3.2.3 Polyphonic Audio Mixing

The background noise samples with their original volume in dBFS are overlaid on the isolated click sub-samples to produce polyphonic audio. Python's PyDub[52] library is used to accomplish this. The resulting augmented click sub-sample is of length 64 milliseconds. Appendix A.3 provides the code for polyphonic audio mixing.

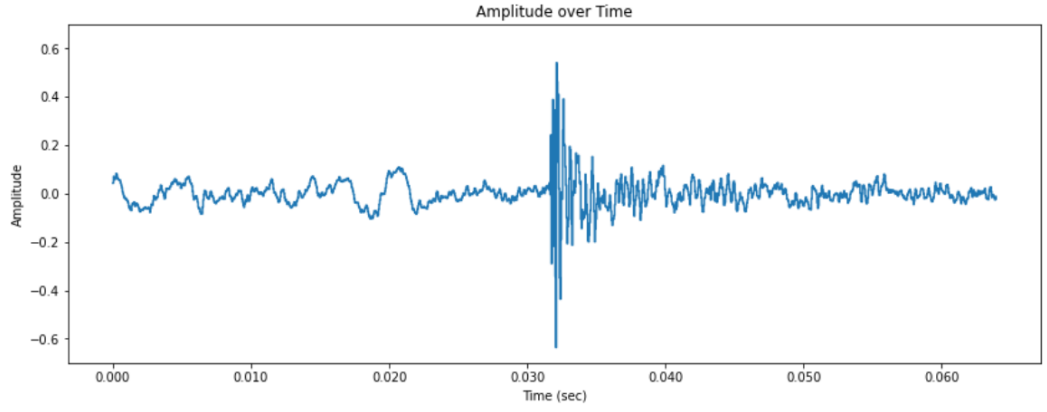

Figure 3.4: Visualization of noise augmented down click sub-sample in Apple i-phone 10 in a quite environment, desired phone position, slow compression rate and good depth with click on a carpet floor

3.2.4 Normal Data Set

From all raw audio samples, 1560 sub-samples were extracted, with 780 down clicks and 780 up clicks for two classes. There are 780 up clicks, 780 down clicks, and 780 noise sub-samples for three classes. The information in table 3.4 includes the number of clicks in various tests as well as the number of isolated down and up clicks for two classes. Subsections 3.2.6 and 3.2.7 provide descriptions of two and three classes, respectively.

Table 3.4: Overall description of Normal Data Set for two classes with the number of raw sub-samples used, no. of clicks generated from each raw audio sample and total number of isolated down and up clicks

Tests	Click Type	No. of clicks isolated from each raw audio sample (a)	No. of raw audio samples used (b)	No. of isolated clicks (a * b)	Total no. of isolated clicks (down and up)
Test 1	down click up click	30 30	5	150 150	300
Test 2	down click up click	30 30	5	150 150	300
Test 3	down click up click	30 30	5	150 150	300
Test 4	down click up click	30 30	2	60 60	120
Test 5	down click up click	30 30	2	60 60	120
Test 6	down click up click	30 30	4	120 120	240
Test 8	down click up click	30 30	3	90 90	180
Total	down click up click	30 30	26	780 780	1560

3.2.5 Augmented Data Set

The augmented data set was created to provide more samples as input to our models. 30 randomly selected background noise samples with their original volume in dBFS were overlaid on the 1560 (780 down clicks and 780 up clicks) isolated click sub-samples to produce a total of 46800 (23400 down clicks and 23400 up clicks) augmented sub-samples.

3.2.6 Two Class

The two classes used in this thesis are obtained from raw audio samples as explained in Subsection 3.2.1 and 3.2.2. The said classes are:

- down clicks
- up clicks

3.2.7 Three Class

The reason for creating three classes is that as the number of classes increases, so does classification precision. Because minor significant features that aren't strong enough to manifest in a coarse classification but are shared across classes have a significant impact as the number of classes increases[53]. The noise class was created using the data set explained in 3.1.2 and the code can be found at Appendix A.4.

For the normal data set, 780 noise samples were chosen at random from a pool of 3399 noise samples and clipped to be 64 seconds long, similar to down and up clicks. Because there were only 3399 samples, we took the first 64 seconds, the last 64 seconds, the middle 64 seconds, increased the first and last 64 seconds' loudness by 6 dB, and decreased the first and last 64 seconds' queitness by 3 dB to generate a noise dataset with 23,400 noise samples for the augmented data set.

The three classes used in this thesis are:

- down clicks
- up clicks
- noise

3.2.8 Final Data Set

The final dataset is made up of four datasets. There are 1560 samples in the two-class normal dataset. There are 46800 samples in the two class augmented data set. The normal data set has 2340 samples, while the augmented data set has 70200 samples.The summary of the dataset used in this thesis is given in Table 3.5.

Table 3.5: Final dataset used in this thesis.

Datasets	No. of down clicks	No. of up clicks	No. of noise samples	Total no. of subsamples
Two class Normal Data set	780	780	-	1560
Two class Augmented Data set	23400	23400	-	46800
Three class Normal Data set	780	780	780	2340
Three class Augmented Data set	23400	23400	23400	70200

3.3 Feature Extraction

In polyphonic sound event detection research, a variety of feature extraction methods have been used. The log-scaled mel spectrogram and MFCC are the most commonly used features [54][55]. MFCC as well as log-scaled mel spectrograms were generated and tested to identify the best features for this dataset for which the code can be found at Appendix A.5. The python library, Librosa [56] is used to extract these features.

3.3.1 MFCC

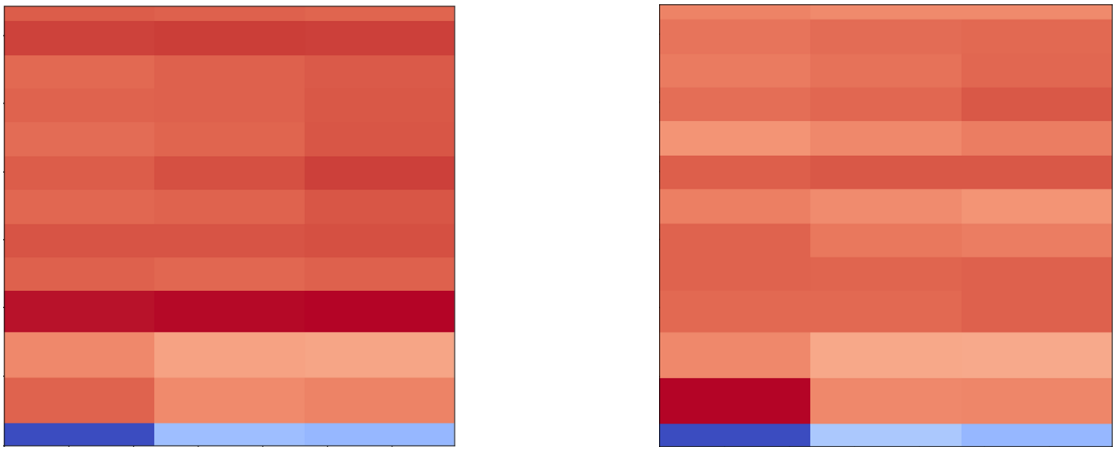

Figure 3.5: Examples of MFCCs obtained from isolated click subsamples

Isolated click subsamples are recorded at 44100 Hz or 44.1 kHz.The following procedure is used to generate log-scaled features:

- Short frames are used to frame the signal.
- Divide the STFT of the audio signal according to the mel-scale.
- The Discrete Fourier Transform is computed. Transform the coefficients of 40 to get the result.

- As a result, 40 MFCCs with a 50% overlap of size 574*574 are calculated for each time window. Some examples of MFCCs of isolated clicks subsamples can be seen in Figure 3.5

3.3.2 Log-Scaled Mel Spectrogram

The following procedure is used to generate log-scaled features:

- Sample the audio datasets with a small sample at a fixed sample rate of 44100.
- Then, with a hop length of 512, windowing is performed.
- The STFT of the audio signal is then divided by the Mel-scale.
- Then we figure out how many FFTs are needed for the audio samples. The maximum number of FFTs used is 2048.
- Then we make the required number of mel bands. 128 mel bands were used in this case.
- The log of the resulting spectrogram is then computed. The images that result are 2D and 574*574 in size. Figure 3.6 depicts some of the generated images.

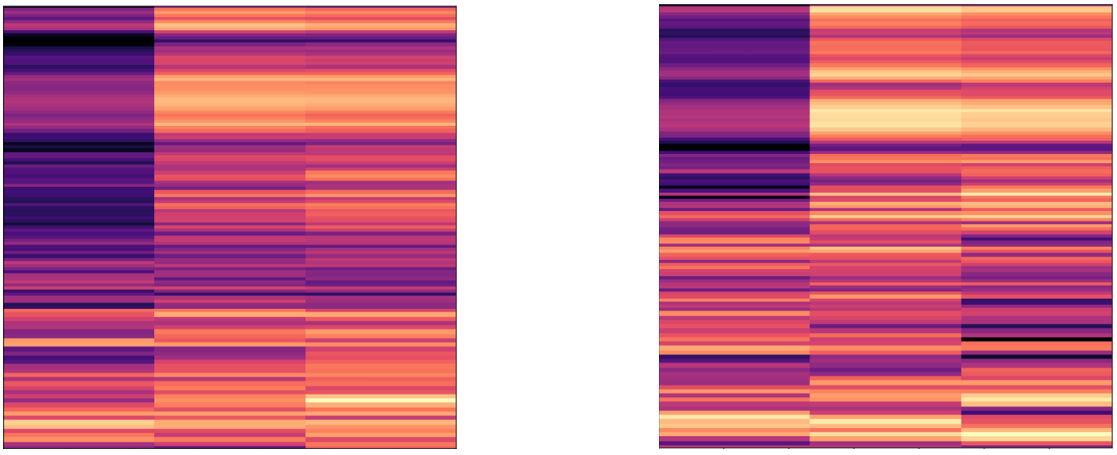

Figure 3.6: Examples of log-scale mel spectrograms obtained from isolated click subsamples

3.4 Train, Validation and Test Set

We take a novel approach to separating the normal and augmented data sets manually. For example, a test folder containing data sets from all five phones will be distributed so

that the training set contains samples from three phones (60 percent), the validation set contains samples from one phone (20 percent), and the test set contains samples from one phone (20 percent) => (3:1:1). Where data is not available from all five phones, we distribute the test folders based on the overall dataset, ensuring that our training set comprises 60% of the dataset, 20% of the test set, and 20% of the validation set. Because of the nature of our dataset, we have chosen 60:20:20.

The test dataset has datasets that are not in the training or validation sets. We did this so that the test set included data that the model had not seen before.

Table 3.6 summarizes the distribution of data sets into train, validation and test sets.

Table 3.6: Distribution of Data sets into Train, Validation and Test Sets

Datasets	Click Type	Train Set	Val Set	Test Set	Train Total	Test Total	Val Total	Overall Sample Size
Two Class Normal Dataset	down click up click	480 480	150 150	150 150	960	300	300	1560
Two Class Augmented Dataset	down click up click	14400 14400	4500 4500	4500 4500	28800	9000	9000	46800
Three Class Normal Dataset	down click up click noise	480 480 480	150 150 150	150 150 150	1440	450	450	2340
Three Class Augmented Dataset	down click up clicks noise	14400 14400 14400	4500 4500 4500	4500 4500 4500	43200	13500	13500	70200

Chapter 4

Methodology

This chapter describes the methods and architectures used in this thesis for multiclass classification tasks. Figure 4.1 depicts the section of the thesis addressed in this chapter.

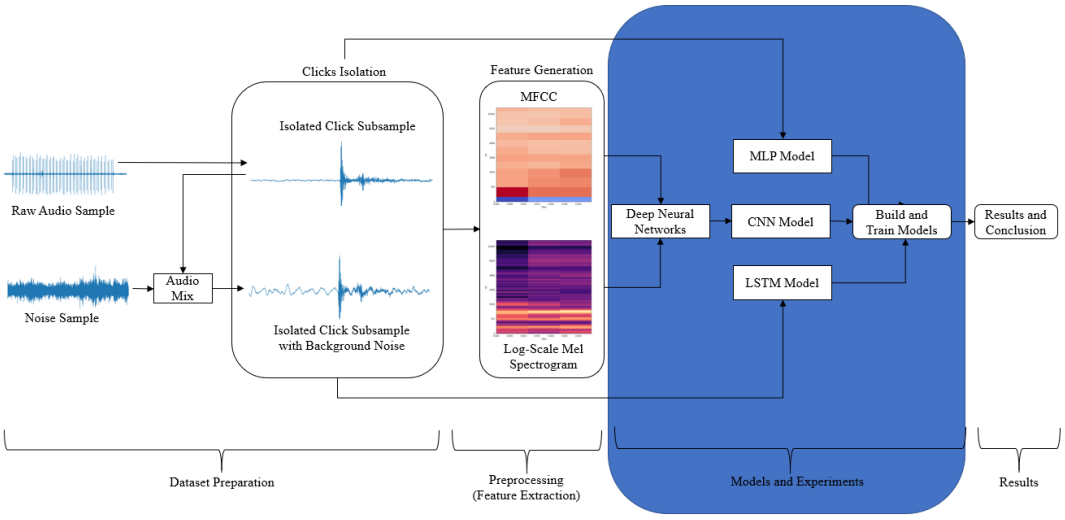

Figure 4.1: Overview of all the steps involved in this thesis highlighting the methodolody.

4.1 Proposed Methods

The proposed methods investigate the use of MLP, LSTM, and CNN to perform multi-class classification tasks. The loss function used in multi-class classification is categorical cross-entropy. Each model is run on the same number of epochs with early stopping. The early stop is used to determine whether or not the validation accuracy improves, and if it does not, training is terminated even if it has not reached the final epoch. As a result, we avoid widening the gap between training and validation accuracy and terminate training

before the model memorizes the dataset further. If the validation accuracy doesn't really improve after 15 epochs, the training is stopped.A checkpoint is used to capture the most accurate validation. Adam is the optimizer that we employ. All of the models use a learning rate of 0.00001. The models in this thesis were all created from scratch.

4.1.1 MLP Model

The MLP model architecture has five layers: an input layer, three hidden layers (512 nuerons in hidden layer 1), (256 neurons hidden layer 2), and (64 neurons hidden layer 3), and a output layer with 2 neurons for a two-class problem. With the exception of a output layer with 3 neurons, the same MLP model architecture is used for three class problems. The model is fed an array of MFCCs and class labels as input. Before feeding the input into the model, it is flattened to make it one-dimensional. The three hidden layers have a 'relu' activation function. This activation function is used to avoid Vanishing Gradient Problem. Dropout is used after all three hidden layers, and the kernal regularizer L2 is used in all hidden layers. These regularizers are used to avoid overfitting. A normal distribution weight initializer with mean = 0 and standard deviation = 0.01 is used. A bias initializer is used to generate tensors with initial values of 0.

For multiclass classification, a softmax activation is used. The total number of parameters in this model for two class classification is 155,074, with 155,074 trainable parameters and 0 non-trainable parameters. Table 4.1 contains detail information about MLP model architecture for two class classification and the code can be found at A.6. The same model architecture is used for normal and augmented dataset for two class classification.

Table 4.1: Layers Summary of MLP model architecture for two class classification

Layer Type	Output Shape	Param
flatten (Flatten)	(None, 13)	0
dense (Dense)	(None, 512)	7168
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 64)	16448
dropout_2 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 2)	130

The total number of parameters in MLP model for three class classification is 155,139, with 155,139 trainable parameters and 0 non-trainable parameters. Table 4.2 contains detail information about MLP model architecture for three class classification and the code can be found at A.7. The same model architecture is used for normal and augmented dataset for three class classification.

Table 4.2: Layers Summary of MLP model architecture for three class classification

Layer Type	Output Shape	Param
flatten (Flatten)	(None, 13)	0
dense (Dense)	(None, 512)	7168
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 64)	16448
dropout_2 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 3)	195

4.1.2 LSTM Model

The LSTM model architecture has five layers: an input LSTM layer (128 neurons in LSTM layer), one lstm layer (64 neurons lstm layer 1), two hidden layers (64 neurons hidden layer 1) and (64 neurons hidden layer 2) and a output layer with 2 neurons for a two-class problem. With the exception of a output layers with 3 neurons, the same LSTM model architecture is used for three class problems. The model is fed an array of MFCCs and class labels as input. Before feeding the input into the model, it is one-dimensional. The two hidden layers have a 'relu' activation function. This activation function is used to avoid Vanishing Gradient Problem. Dropout is used after all two hidden layers and LSTM layer 1, and the kernal regularizer L2 is used in all hidden layers. These regularizers are used to avoid overfitting. A normal distribution weight initializer with mean = 0 and standard deviation = 0.01 is used. A bias initializer is used to generate tensors with initial values of 0.

For multiclass classification, a softmax activation is used. The total number of parameters in this model for two class classification is 128,418, with 128,418 trainable parameters and 0 non-trainable parameters. Table 4.3 contains detail information about LSTM model architecture for two class classification and the code can be found at A.8. The same model architecture is used for normal and augmented dataset for two class classification.

Table 4.3: Layers Summary of LSTM model architecture for two class classification

Layer Type	Output Shape	Param
lstm (LSTM)	(None, 1, 128)	72704
lstm_1 (LSTM)	(None, 64)	49408
dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 64)	4160
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2080
dropout_2 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 2)	66

The total number of parameters in LSTM model for three class classification is 128,451, with 128,451 trainable parameters and 0 non-trainable parameters. Table 4.4 contains detail information about LSTM model architecture for three class classification and the code can be found at A.9. The same model architecture is used for normal and augmented dataset for three class classification.

Table 4.4: Layers Summary of LSTM model architecture for three class classification

Layer Type	Output Shape	Param
lstm (LSTM)	(None, 1, 128)	72704
lstm_1 (LSTM)	(None, 64)	49408
dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 64)	4160
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2080
dropout_2 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 3)	99

4.1.3 CNN Model

The CNN model architecture has five layers: an input covolutional with 32 neurons and a filter kernel size of 3*3., three other convolutional layers (layer 1 with 64 neurons, layer 2 with 128 neurons and layer 3 with 128 neurons, all with a filter kernel size of 3*3.) and a output layer with 2 neurons for a two-class problem. The output layer also has a hidden layer with 64 neurons With the exception of three output layers, the same CNN model architecture is used for three class problems. As input, the model is fed 2 dimensional (2D) images of size 574 * 574. For the convolutional layers and hidden layer, we used ReLU activation, and for the final output layer, we used softmax. 2D max pooling was added after each convolutional layer. Dropout was added after each convolutional and hidden layer to prevent overfitting. Our network has also received some L2 regularization. The normal distribution weight initializer is used, with mean = 0 and standard deviation = 0.01. To generate tensors with zero initial values, a bias initializer is used.

For multiclass classification, a softmax activation is used. The total number of parameters in this model for two class classification is 10,857,858, with 10,857,858 trainable parameters and 0 non-trainable parameters. Table 4.5 contains detail information about CNN model architecture for two class classification and the code can be found at A.10. The same model architecture is used for normal and augmented dataset for two class classification.

The total number of parameters in CNN model for three class classification is 10,857,923, with 10,857,923 trainable parameters and 0 non-trainable parameters. Table 4.6 contains detail information about CNN model architecture for three class classification and

Table 4.5: Layers Summary of CNN model architecture for two class classification

Layer Type	Output Shape	Param
conv2d (Conv2D)	(None, 574, 574, 32)	896
max_pooling2d (MaxPooling2D)	(None, 287, 287, 32)	0
dropout (Dropout)	(None, 287, 287, 32)	0
conv2d_1 (Conv2D)	(None, 287, 287, 64)	18496
max_pooling2d_1 (MaxPooling 2D)	(None, 144, 144, 64)	0
dropout_1 (Dropout)	(None, 144, 144, 64)	0
conv2d_2 (Conv2D)	(None, 144, 144, 128)	73856
max_pooling2d_2 (MaxPooling 2D)	(None, 72, 72, 128)	0
dropout_2 (Dropout)	(None, 72, 72, 128)	0
conv2d_3 (Conv2D)	(None, 72, 72, 128)	147584
max_pooling2d_3 (MaxPooling 2D)	(None, 36, 36, 128)	0
dropout_3 (Dropout)	(None, 36, 36, 128)	0
flatten (Flatten)	(None, 165888)	0
dense (Dense)	(None, 64)	10616896
dropout_4 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 2)	130

the code can be found at A.11. The same model architecture is used for normal and augmented dataset for three class classification.

Table 4.6: Layers Summary of CNN model architecture for three class classification

Layer Type	Output Shape	Param
conv2d (Conv2D)	(None, 574, 574, 32)	896
max_pooling2d (MaxPooling2D)	(None, 287, 287, 32)	0
dropout (Dropout)	(None, 287, 287, 32)	0
conv2d_1 (Conv2D)	(None, 287, 287, 64)	18496
max_pooling2d_1 (MaxPooling 2D)	(None, 144, 144, 64)	0
dropout_1 (Dropout)	(None, 144, 144, 64)	0
conv2d_2 (Conv2D)	(None, 144, 144, 128)	73856
max_pooling2d_2 (MaxPooling 2D)	(None, 72, 72, 128)	0
dropout_2 (Dropout)	(None, 72, 72, 128)	0
conv2d_3 (Conv2D)	(None, 72, 72, 128)	147584
max_pooling2d_3 (MaxPooling 2D)	(None, 36, 36, 128)	0
dropout_3 (Dropout)	(None, 36, 36, 128)	0
flatten (Flatten)	(None, 165888)	0
dense (Dense)	(None, 64)	10616896
dropout_4 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 3)	195

4.2 Classification Method

The classification method used in this thesis is multiclass classification because we are using a categorical loss function. When there are two or more label classes, the categorical

loss function is used. Labels are expected to be one-shot representations. Labels can be provided as 'categorical' or 'integer' labels. When the labels are provided as 'integers,' the sparse categorical function is used [57].

Another reason for using the loss function was the plan to expand our work from two to three classes from the start. There would be little change in the code and model configuration, and it would work in both cases.

4.2.1 Multiclass Classification

The inputs are divided into K distinct classes. K = 2 for two classes and K = 3 for three classes in our case. All of the experiments in this thesis were conducted with K = 2 and K = 3.

Chapter 5

Experiments and Results

This section presents the results of various experiments conducted with various datasets and models to demonstrate model performance. The section of the thesis address in this chapter is depicted in Figure 5.1.

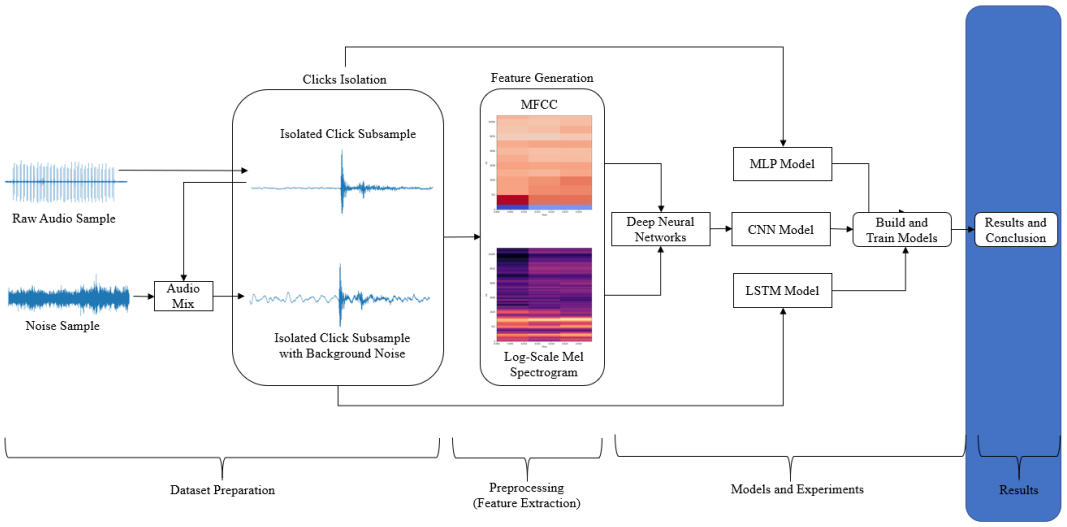

Figure 5.1: Overview of all the steps involved in this thesis highlighting the results.

5.1 Experimental Setup

The experiments begin with the initialization of the image shapes. The MLP and LSTM models were tested with MFCCs in the form of an array. This array contains MFCCs as well as labels. The CNN model was tested using the original 'image' sizes generated by feature extraction. Following that, the datasets were loaded and normalized. Because the images were in 'RGB' format, they were divided by 255. Each experiment was

then assigned an optimizer, and the hyperparameters were tuned. Following that, the model was trained over a predetermined number of epochs. The resultant model was saved and loaded. To evaluate the model, a validation dataset was used. To validate model performance (overfitting and underfitting), the validation loss, validation accuracy, training loss, and training accuracy were compared . If the model was overfitting, underfitting, or inaccurate, new hyperparameters were tried until the best results were obtained for each experiment. Finally, test dataset was used to predict the classes, compute the best model and to compute confusion matrix.

Three models were tested, each with four experiments: two class normal, two class augmented, three class normal, and three class augmented.

5.1.1 Hyperparameter Tuning

The hyperparameters were fine-tuned by training multiple models with various combinations of optimizers, learning rates, batch sizes, loss functions, and dropout. The validation datasets were used to evaluate the best combination of hyperparameters. Grid search was the hyperparameter optimization strategy used in the multiclass classification experiment.

5.1.2 Learning Rate

The learning rates were set during the initial grid search, and subsequent adjustments were made based on the observed results. Adam has the best learning rate of 0.00001.

5.1.3 Dropout

Dropout was implemented to address overfitting issues. The dropout rate for the MLP model was 0.3, while it was 0.2 for the LSTM and CNN models.

5.2 MLP Experiments and Results

The MLP model was used to experiment with four datasets: two class normal, two class augmented, three class normal and three class augemented. This experiment was run with 100 epochs with early stopping at 15. It stopped learning at epoch 71 for two class normal and for two class augmented it stopped at epoch 61. For three class normal, it ran for 100 epoch and for three class augmented it stopped at 89. The optimizer used

was Adam and learning rate = 0.00001. The results for prediction on test datasets are shown in Table 5.1.

Table 5.1: Test results with MLP

Datasets	Accuracy	Precision	Recall	F1 Score
Two Class Normal	0.8267	0.8297	0.8220	0.825
Two Class Augmented	0.8197	0.7984	0.8553	0.8259
Three Class Normal	0.8653	0.8635	0.8653	0.8637
Three Class Augmented	0.8431	0.8412	0.8431	0.8420

The confusion matrices are given in figure 5.2. In these matrices, label 0 represents down clicks, label 1 represents up clicks and label 2 represents noise.

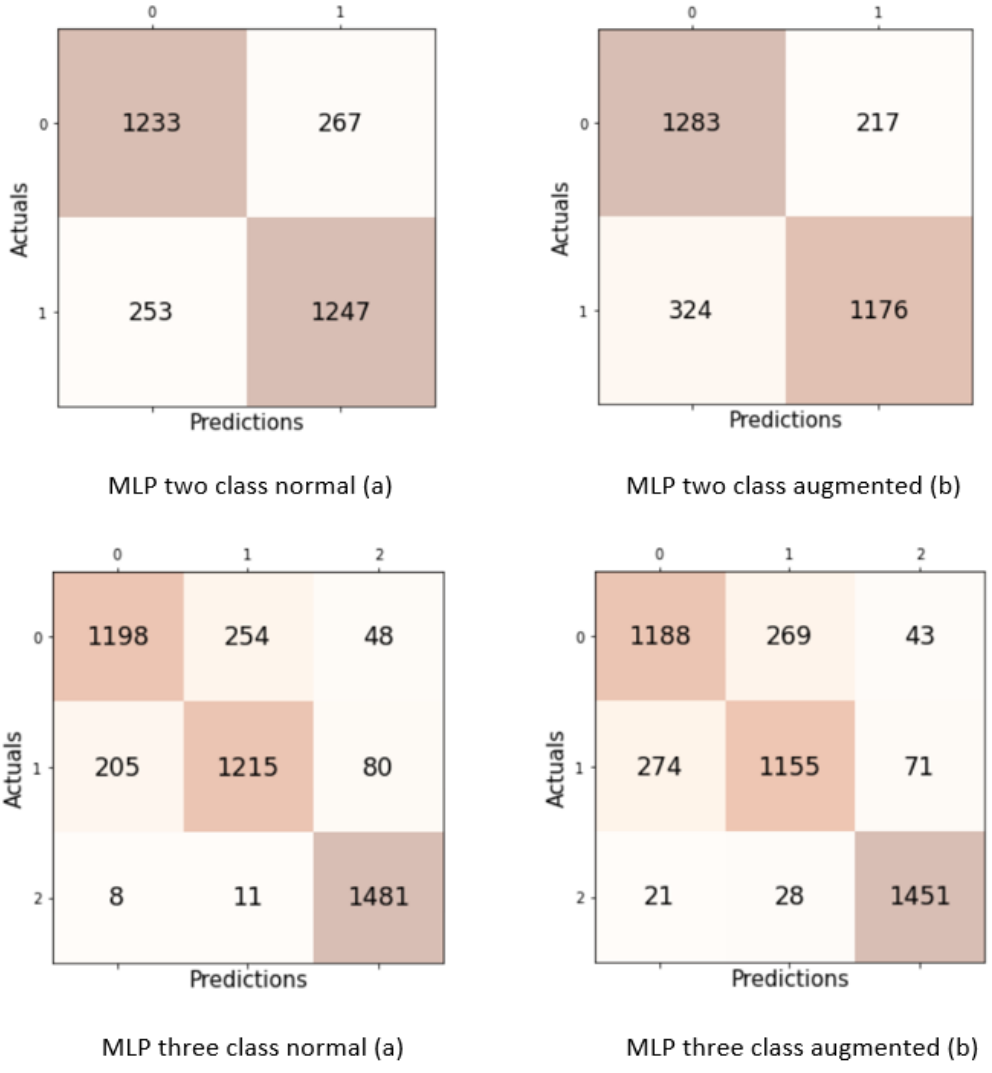

Figure 5.2: Confusion matrix for MLP model.

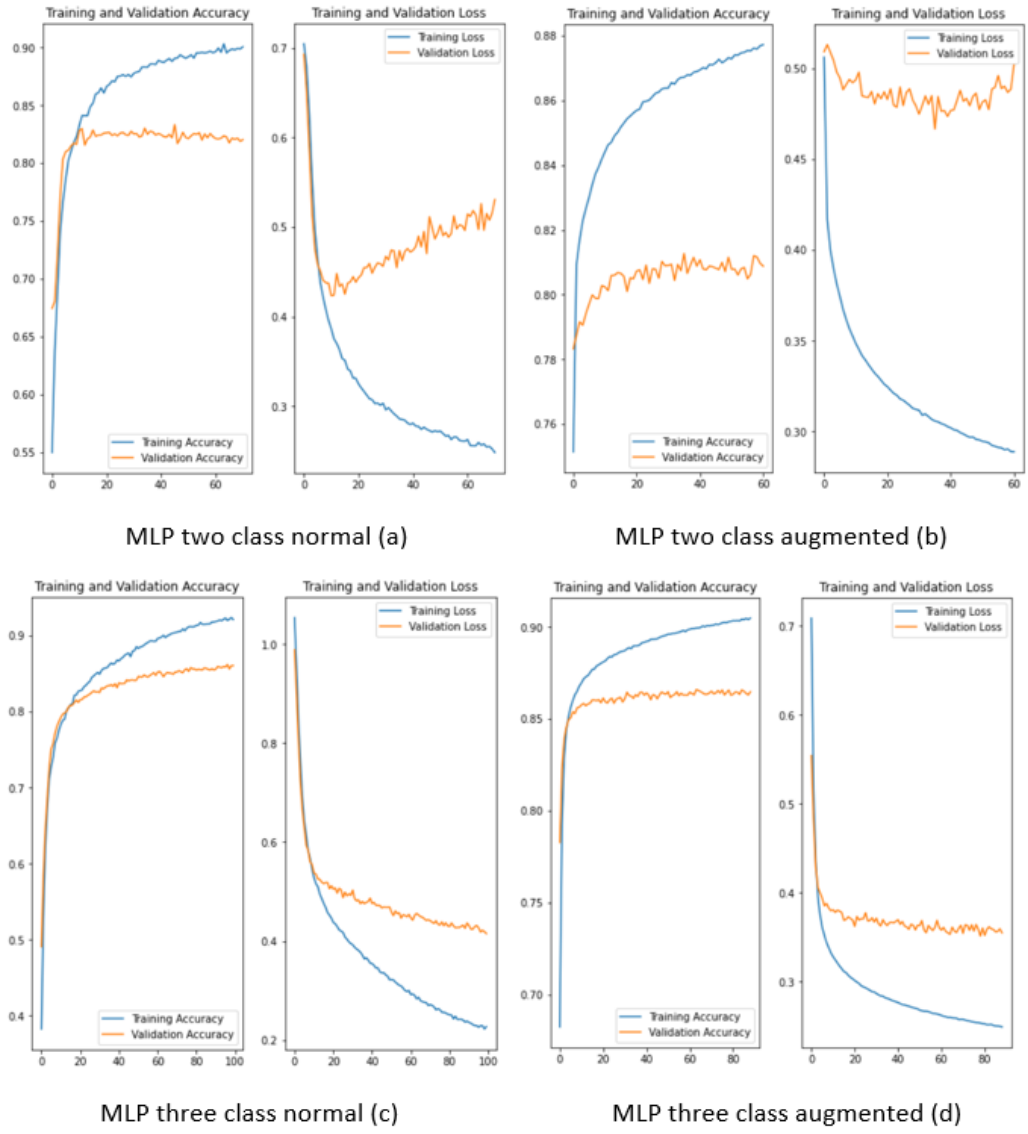

Figure 5.3: Training and Validation Plots for MLP.

5.3 LSTM Experiments and Results

The LSTM model was used to experiment with four datasets: two class normal, two class augmented, three class normal and three class augemented. This experiment was run with 100 epochs with early stopping at 15. The learning stopped at epoch 60 for two class normal and at epoch 84 for two class augmented. Three class normal ran for all 100 epoch and at 72 epoch, the three class augmented stopped learning. The optimizer used was Adam and learning rate = 0.00001. The results for the validation and test datasets are shown in Table 5.2.

The confusion matrices are given in figure 5.4. In these matrices, label 0 represents down clicks, label 1 represents up clicks and label 2 represents noise.

Table 5.2: Test results with LSTM

Datasets	Accuracy	Precision	Recall	F1 Score
Two Class Normal	0.8517	0.8450	0.8613	0.8531
Two Class Augmented	0.8153	0.7964	0.8473	0.8211
Three Class Normal	0.8391	0.8414	0.8391	0.8389
Three Class Augmented	0.8364	0.8370	0.8364	0.8347

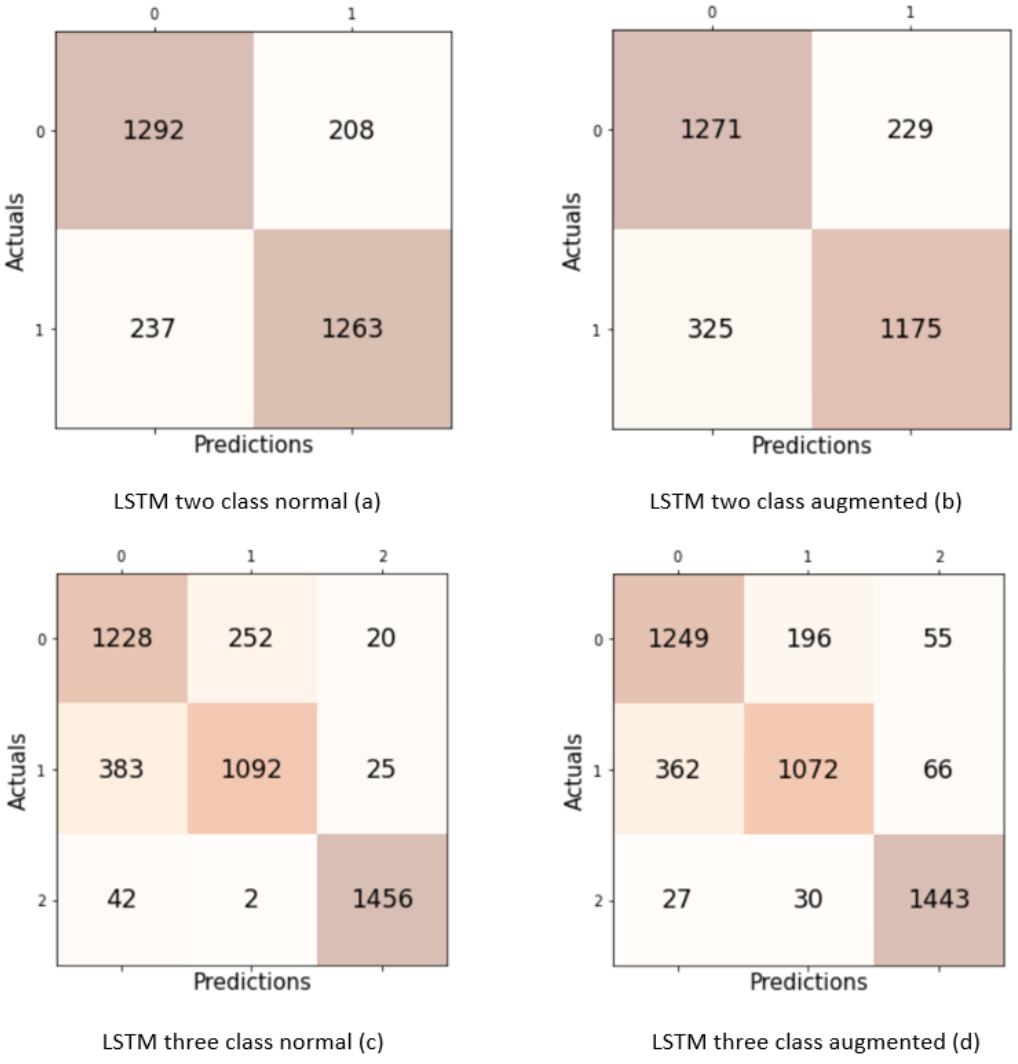

Figure 5.4: Confusion matrix for LSTM model.

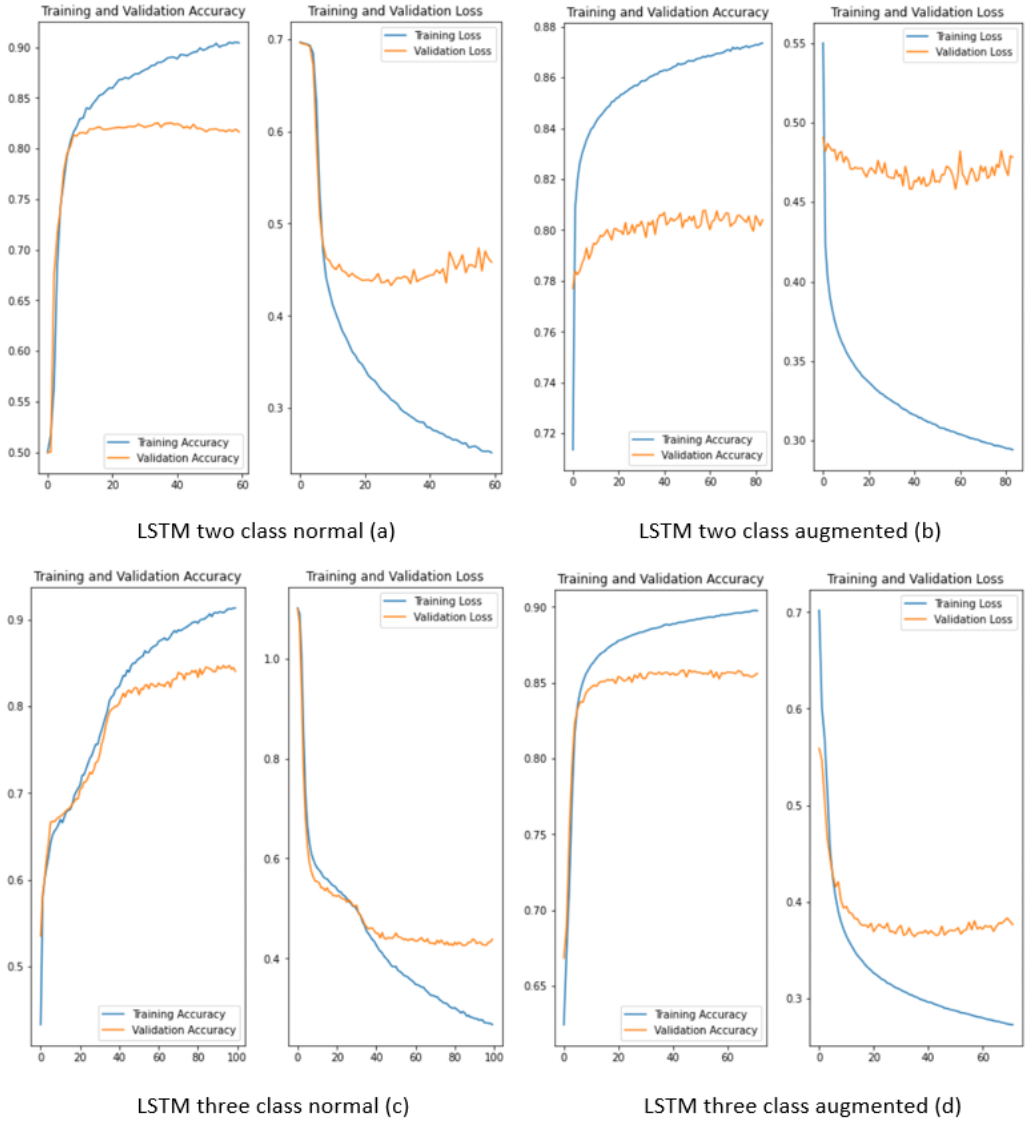

Figure 5.5: Training and Validation Plots for LSTM.

5.4 CNN Experiments and Results

The CNN model was used to experiment with four datasets: two class normal, two class augmented, three class normal and three class augemented. This experiment was run with 100 epochs with early stopping at 10. It stopped learning at epoch 31 for two class normal and at epoch 15 for two class augmented. The three class normal stopped learning at epoch 27 and the three class augmented stopped learning at epoch 21. The optimizer used was Adam and learning rate = 0.00001. The results for the validation and test datasets are shown in Table 5.3.

The confusion matrices are given in figure 5.6

Table 5.3: Test results with CNN

Datasets	Accuracy	Precision	Recall	F1 Score
Two Class Normal	0.9967	0.9934	1.000	0.9967
Two Class Augmented	0.8933	0.8242	1.000	0.9036
Three Class Normal	0.9978	0.9978	0.9978	0.9978
Three Class Augmented	0.9533	0.9583	0.9533	0.9533

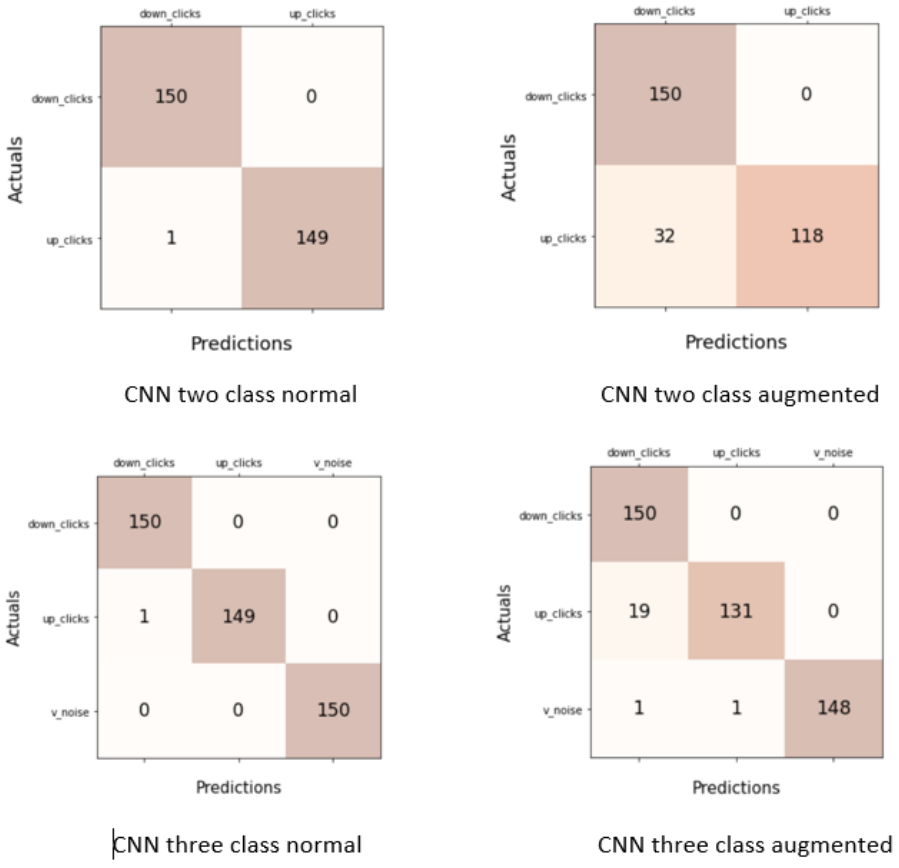

Figure 5.6: Confusion matrix for CNN model

5.5 Analysis of the Result

Four experiments were conducted for each of three models.

1. For MLP model, the accuracy of three class normal was the highest.
2. For LSTM model, the accuracy of two class normal was the highest.
3. For CNN model, the accuracy of three class normal was the highest.
4. The accuracy decreases for augmented dataset in all three models.

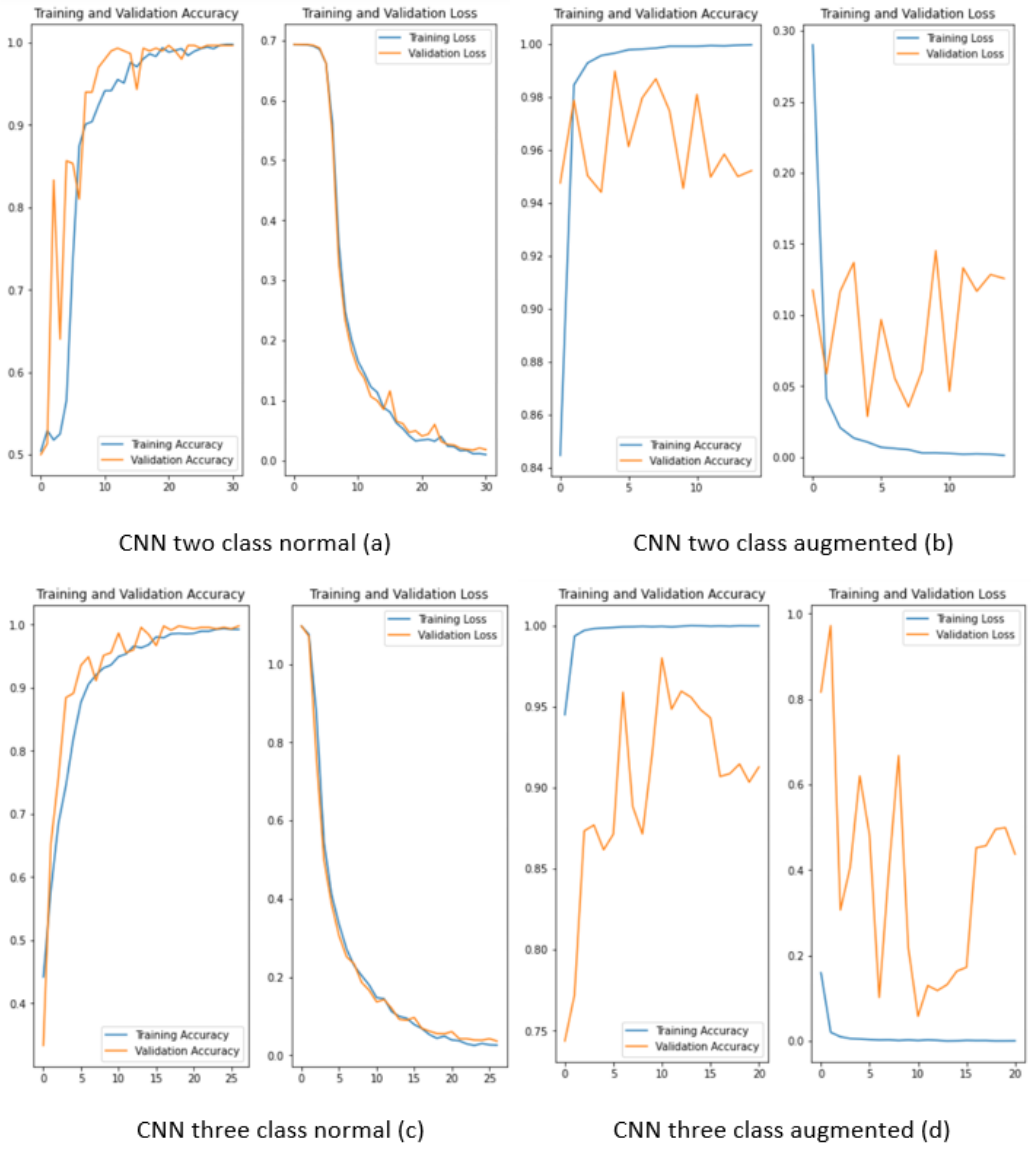

Figure 5.7: Training and Validation Plots for CNN

The prediction is performed using the test dataset, and the confusion matrix is computed. The confusion matrix shows that more up clicks were predicted as down clicks than down clicks were predicted as up clicks or noise was predicted as down click or up click.

Overall, the CNN model outperforms the others.

Chapter 6

Discussions

This chapter discusses about the model's performnace, limitations and future work.

6.1 Model Performance

Because the datasets in this work are balanced, the accuracy is the main performance matrix regarded for these models. The F1 score is also used to assess the robustness of models. Furthermore, the cost of miss-classification in the case of model implementation is discussed by taking the precision and recall scores from each experiment into account.

Overall, the CNN model performed well. However, when switching from normal to augmented data, the model's performance decreased. The reason we augmented and expanded the dataset was to avoid overfitting and improve model performance. However, we are getting the opposite result. This could be due to the noise samples we're using to supplement the dataset. It is becoming more difficult for the model to recognize the audio samples as we add noise to the existing dataset.

6.2 Limitations

The models are tested with datasets that the model has never seen before, but not with new test data.

6.3 Future Work

- The use and testing of pre-trained models may aid in improving results.

- Additionally, the models' results can improve further fine tuning.

Chapter 7

Conclusions

This thesis aimed to detect clicks in compressed data. A down click is followed by an up click to form a complete compression. To detect compression, the down and up clicks were separated and detected. To accomplish this, three DNN models were tested: MLP, LSTM, and CNN.

Many steps were involved in heading from a raw audio sample to detecting clicks in those samples. First, a data set suitable for the models had to be made. Four data sets were created. Two of these data sets were normal audio sub samples with isolated clicks and no additional noise, while the other two were poly polyphonic audio sub samples or augmented sub samples created by overlaying provided audio samples with background noise. Both the normal and polyphonic audio sub samples were then converted to log-scaled mel spectrograms, which were then fed into the aforementioned DNN models. Two classes were defined while feeding these data sets to the DNN models: down clicks and up clicks, and three classes: down clicks, up clicks, and noise. Each data set was split into three sections: training, validation, and testing. The two class normal data set contained 1560 subsamples, the two class augmented data set contained 46800 subsamples, the three class normal data set contained 2340 subsamples, and the three class augmented data set contained 70200 subsamples.

For two classes, CNN had the highest overall accuracy of 99.7 percent and the highest F1 score of 0.9967. Based on the findings of this thesis, it is concluded that isolating clicks from raw audio samples and using CNN to detect clicks is a promising method.

Appendix A

Appendix Codes

A.1 Click Detection

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.2 Click Isolation

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.3 Create Polyphonic Audio Mix

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.4 Create Third Class

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.5 Generate Features

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.6 MLP for two class

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.7 MLP for three class

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.8 LSTM for two class

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.9 LSTM for three class

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.10 CNN for two class

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

A.11 CNN for three class

A detailed description can be found at https://github.com/prava-thapa/masterthesis2022

Bibliography

[1] Lærdal Medical. Mini anne, 2022. URL https://laerdal.com/products/simulation-training/resuscitation-training/mini-anne/.

[2] GeeksforGeeks. What are the characteristics of sound waves?, 2021. URL https://www.geeksforgeeks.org/what-are-the-characteristics-of-sound-waves/.

[3] Wikimedia Commons, 2017. URL https://commons.wikimedia.org/wiki/File:FFT-Time-Frequency-View.png.

[4] S Abirami and P Chitra. The digital twin paradigm for smarter systems and environments: The industry use cases. In *Adavances in Computers*, 2020.

[5] Niklas Lang. Using convolutional neural network for image classification, 2021. URL https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4.

[6] dataaspirant.com. Confusion matrix, 2020. URL https://dataaspirant.com/wp-content/uploads/2020/08/3_confusion_matrix.png.

[7] Øyvind Meinich-Bache, Kjersti Engan, Tonje Søraas Birkenes, and Helge Myklebust. Real-time chest compression quality measurements by smartphone camera. *Journal of healthcare engineering*, 2018, 2018.

[8] Raf J Van Hoeyweghen, Leo L Bossaert, Arsene Mullie, Paul Calle, Patrick Martens, Walter A Buylaert, Herman Delooz, Belgian Cerebral Resuscitation Study Group, et al. Quality and efficiency of bystander cpr. *Resuscitation*, 26(1):47–52, 1993.

[9] E John Gallagher, Gary Lombardi, and Paul Gennis. Effectiveness of bystander cardiopulmonary resuscitation and survival following out-of-hospital cardiac arrest. *Jama*, 274(24):1922–1925, 1995.

[10] MICHAEL P Feneley, GEORGE W Maier, Karl B Kern, J WILLIAM Gaynor, SA Gall Jr, ARTHUR B Sanders, K Raessler, LH Muhlbaier, J Scott Rankin, and GA Ewy. Influence of compression rate on initial success of resuscitation and 24 hour

survival after prolonged manual cardiopulmonary resuscitation in dogs. *Circulation*, 77(1):240–250, 1988.

[11] Robin Dowie, Helen Campbell, Rachael Donohoe, and Patricia Clarke. 'event tree'analysis of out-of-hospital cardiac arrest data: confirming the importance of bystander cpr. *Resuscitation*, 56(2):173–181, 2003.

[12] Robert A Berg, Arthur B Sanders, Karl B Kern, Ronald W Hilwig, Joseph W Heidenreich, Matthew E Porter, and Gordon A Ewy. Adverse hemodynamic effects of interrupting chest compressions for rescue breathing during cardiopulmonary resuscitation for ventricular fibrillation cardiac arrest. *Circulation*, 104(20):2465–2470, 2001.

[13] Contributors and Reviewers for the Neonatal Resuscitation Guidelines. International guidelines for neonatal resuscitation: an excerpt from the guidelines 2000 for cardiopulmonary resuscitation and emergency cardiovascular care: international consensus on science. *Pediatrics*, 106(3):e29–e29, 2000.

[14] 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.

[15] 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.

[16] Raymond Farah, Eva Stiner, Zmora Zohar, Arie Eisenman, and Fabio Zveibil. The importance of cpr training for assessing the knowledge and skills of hospital medical and nursing personnel. *Harefuah*, 146(7):529–33, 2007.

[17] Diana M Cave, Tom P Aufderheide, Jeff Beeson, Alison Ellison, Andrew Gregory, Mary Fran Hazinski, Loren F Hiratzka, Keith G Lurie, Laurie J Morrison, Vincent N Mosesso Jr, et al. Importance and implementation of training in cardiopulmonary resuscitation and automated external defibrillation in schools: a science advisory from the american heart association. *Circulation*, 123(6):691–706, 2011.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] Donmoon Lee, Subin Lee, Yoonchang Han, and Kyogu Lee. Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input. *DCASE*, 1:14–18, 2017.

[24] 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.

[25] 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.

[26] 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.

[27] Maalidefaa Moses Tantuoyir. Sound event detection from aed inteam training situations using dnns. Master's thesis, uis, 2021.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Ruth M Fanning and David M Gaba. The role of debriefing in simulation-based learning. *Simulation in healthcare*, 2(2):115–125, 2007.

[33] Bhanu Prasad and SR Mahadeva Prasanna. *Speech, audio, image and biomedical signal processing using neural networks*, volume 83. Springer, 2007.

[34] Ben Gold, Nelson Morgan, and Dan Ellis. *Speech and audio signal processing: processing and perception of speech and music*. John Wiley & Sons, 2011.

[35] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-hill New York, 1986.

[36] E Oran Brigham. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.

[37] 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.

[38] 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.

[39] Stanley S Stevens and John Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329–353, 1940.

[40] SoundEar A/S, 2022. URL https://soundear.com/decibel-scale/.

[41] 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.

[42] 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.

[43] 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.

[44] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International conference on artificial neural networks*, pages 799–804. Springer, 2005.

[45] Keras. Dropout layer, 2022. URL https://keras.io/api/layers/regularization_layers/dropout/.

[46] Keras. Flatten layer, 2022. URL https://keras.io/api/layers/reshaping_layers/flatten/.

[47] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[48] Rafael Anicet Zanini and Esther Luna Colombini. Parkinson's disease emg data augmentation and simulation with dcgans and style transfer. *Sensors*, 20(9):2605, 2020.

[49] Fang Wang, Sheng-hua Zhong, Jianfeng Peng, Jianmin Jiang, and Yan Liu. Data augmentation for eeg-based emotion recognition with deep convolutional neural networks. In *International conference on multimedia modeling*, pages 82–93. Springer, 2018.

[50] Daniel Freer and Guang-Zhong Yang. Data augmentation for self-paced motor imagery classification with c-lstm. *Journal of neural engineering*, 17(1):016041, 2020.

[51] SciPy.org. scipy.signal.find_peaks, 2018. URL https://docs.scipy.org/doc/scipy-1.1.0/reference/generated/scipy.signal.find_peaks.html.

[52] James Robert, Marc Webbie, et al. Pydub. *Github*, 2011.

[53] Felix Abramovich and Marianna Pensky. Classification with many classes: challenges and pluses. *Journal of Multivariate Analysis*, 174:104536, 2019.

[54] 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.

[55] Hyungui Lim, Jeong-Soo Park, and Yoonchang Han. Rare sound event detection using 1d convolutional recurrent neural networks. In *DCASE*, pages 80–84, 2017.

[56] Brian McFee, Colin Raffel, Dawen Liang, Daniel P 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.

[57] tensorflow.org. tf.keras.losses.categoricalcrossentropy, 2022. URL https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy.