2020-03-16

RECOGNITION OF ENVIRONMENTAL ALARM SOUND USING ANDROID-BASED SMARTPHONES FOR THE HEARINGIMPAIRED

DEMILIE, YENGUSIE

BAHIR DAR UNIVERSITY

BAHIR DAR INSTITUTE OF TECHNOLOGY

SCHOOL OF RESEARCH AND POSTGRADUATE STUDIES

FACULTY OF COMPUTING

RECOGNITION OF ENVIRONMENTAL ALARM SOUND USING ANDROID-BASED SMARTPHONES FOR THE HEARING-IMPAIRED

YENGUSIE DEMILIE ALENE

BAHIR DAR, ETHIOPIA

July 30, 2019

RECOGNITION OF ENVIRONMENTAL ALARM SOUND USING ANDROID-BASED SMARTPHONES FOR THE HEARING-IMPAIRED

YENGUSIE DEMILIE ALENE

A Thesis submitted to the school of Research and Graduate Studies of Bahir Dar Institute of Technology, Faculty of Computing BDU in partial fulfilment of the requirements for the degree of Master of Science in Information Technology

Advisor Name:

Asrat Mulatu (PhD)

BAHIR DAR, ETHIOPIA

July 30, 2019

DECLARATION

I declare that this thesis titled, Recognition of Environmental Alarm Sound Using Android-Based Smartphones for the Hearing-Impaired and the work presented in it are my own, and all sources of materials used for the thesis has been clearly stated and attributed. In compliance with internationally accepted practices, I have acknowledged and refereed all materials used in this work. I understand that non-adherence to the principles of academic honesty and integrity, misrepresentation/ fabrication of any idea/data/fact/source will constitute sufficient ground for disciplinary action by the University and can also evoke penal action from the sources which have not been properly cited or acknowledged.

Name of the student_____ Signature _____

Date of submission: _____

Place: Bahir Dar

This thesis has been submitted for examination with my approval as a university advisor.

Advisor Name: __Asrat Mulatu_____

Advisor's Signature: __ 

Bahir Dar University

Bahir Dar Institute of Technology-

School of Research and Graduate Studies

Faculty of Computing

THESIS APPROVAL SHEET

Student:

Yengusie Demilie Alene_____

Name Signature Date

The following graduate faculty members certify that this student has successfully presented the necessary written final thesis and oral presentation for partial fulfilment of the thesis requirements for the Degree of Master of Science in Information Technology.

Approved By:

Advisor:

_____

Name Signature Date

External Examiner:

_____

Name Signature Date

Internal Examiner:

_____

Name Signature Date

Chair Holder:

_____

Name Signature Date

Faculty Dean:

_____

Name Signature Date

ACKNOWLEDGMENTS

First, I would like to thank the Almighty GOD, who have blessed and give me wisdom to guide my work so that I am able to achieve the success of this thesis.

I would like to thank my thesis supervisor, Dr. Asrat Mulatu for showing me a brilliant research path, gave me his valuable time, focus for each step I take, guidance, comments and correction to the thesis from the initiation of the idea till the completion of the work. He was always available for my questions and he truly changed my insight on academic research.

I would like to thank Bair Dar Institute of Technology School of Research and Postgraduate Studies and Faculty of Computing for providing me with financial help and funding this thesis and MSc class.

I am also express grateful thanks to my instructors, classmates and friends at Bahir Dar University Computing Faculty who have provided me great motivation, guidance and suggestions for this thesis.

Finally, I would like to express my deepest thankfulness to my family (Demilie, Yezibe, Andualem, Belayneh, Kelemu, Hilena, Mekdes and my husbad (Konjo)) for encouraging me to think me as that doing things are possible from the early stage of my studies till the completion of this thesis. I am very thankful to my mother Yezibe Yiheys and my father Demilie Alene, my brothers Andualem Demilie, Belayneh Demilie and Kelemu Demilie, and my sisters Hilena Demilie and Mekdes Demilie for giving me blessed family love, encouragement and support in all my studies from early schools till the completion of this work.

Yengusie Demilie

July, 2019

ABSTRACT

Environmental alarm sound (EAS) is a rich source of information that can be used to infer awareness of the surrounding, especially for events out of sight and attention. The events may include fire alarm, car horn, dog barking and others. Most of human beings can recognize these alarm sounds and can take a prompt action. However, hearing-impaired (HI) individuals get difficulty to detect these sound events.

This thesis addresses the design and development of a model for detection and recognition of environmental alarm sound (EAS) on mobile phones for the awareness of hearing impaired. We compare performance of different Artificial Neural Network (ANN) sound classifiers (Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP)) and sound feature extraction techniques (Mel Frequency Cepstral Coefficient (MFCC), Mel Spectrogram (Mel), Spectral contrast (SC) and Chroma gram (Chroma)) to select the best ones in the case of MLP classifier. The performance of classifiers evaluated on a set of five EAS classes (dog barking, car horn, gun shot, engine idling and fire alarm). The offline training and testing result show that CNN classifier with log-mel-spectrogram feature outperforms than MLP classifier, in terms of recognition accuracy. The audio feature vector is selected based on its suitability for CNN classifier for offline training and testing and verified to be optimal features. Android-based EAS recognition model is then implemented using CNN classifier on the selected feature.

The recognition performance of the EAS recognition technique is evaluated under **20**% of the EAS dataset for 10 times trial. In this case, EAS recognition technique achieve **89**% recognition accuracy, which outperforms the recognition performance of previously investigated efforts on the area of environmental sound recognition (ESR).

Key Words: Environmental alarm sound recognition, ***Multilayer Perception***, ***Convolutional Neural Network***, ***Mel-frequency cepstral coefficient***, ***Spectral contrast***, ***Mel-spectrogram***, ***Protocol buffer***

Table of Contents

List of Abbreviations

ANN	Artificial Neural Network
BLSTM-RNN	Bidirectional Long Short-Term Memory- recurrent Neural Network
BN	Bayesian Network
Chroma	Chroma gram
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
EAS	Environmental Alarm Sound
EASR	Environmental Alarm Sound Recognition
ESR	Environmental Sound Recognition
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
HI	Hearing Impaired
HMM	Hidden Markov Model
KNN	K-Nearest Neighbour
Mel	Mel-Spectrogram
MFCC	Mel Frequency Cepstral Coefficient
MLP	Multilayer Perceptron
MP	Matching Pursuit

SC	Spectral Contrast
STE	Short Time Energy
STFT	Short Time Fourier Transform
SVM	Support Vector Machine
ZCR	Zero Crossing Rate

List of Figures

List of Tables

CHAPTER ONE

1. Introduction

1.1 Background

Sound is one of human beings most important tool to gather information about the environment next to vision. It is an important and subtle way to gain awareness of the surroundings, especially for events out of sight and attention. The environment surrounding us is composed of variety of sounds. It is relatively easy for most people to make sense of what they hear or to discriminate where they are located in the environment based on the sound alone [1]. For example, alerts such as fire alarms are broadcast to people in homes and public places to notify potential dangers. Household appliances such as microwave ovens beep to indicate that the food is cooked. These sounds are important as alarm that warn people of hazardous situations called danger signals [2]. However, this recognition typically is not the case with machines or hearing-impaired individuals. Nowadays, with the advancement of technology and research efforts, machines can recognize sounds like human speech and musical sounds.

As in [3] mentioned, unlike speech and music, which have formanic and harmonic structures, environmental sounds are considered unstructured since they are variably composed from different sources. That makes it difficult to recognized by machines. In the past decades, several research works investigated on environmental sound recognition (ESR) for different purposes such as hearing aid technology, noise monitoring and measurement, animal bioacoustics, assisting robotics, navigation, long-term audio monitoring, audio surveillance and security applications [4]. Due to the fact that ESR is still in its infancy, the majority of research efforts in this area utilized techniques that were originally developed for the recognition of speech and music [4]. Because of randomness, high variance, comprised of different sources and other difficulties in working with environmental sounds, the recognition accuracy rate falls rapidly with increasing number of sound classes [5].

According to recent survey data reported by the World Health Organisation (WHO, 2013), 360 million people worldwide have disabling hearing loss, and two-thirds of them live in developing countries [6]. It is non-trivial for the hearing-impaired (HI) people to perceive such important sound awareness. While visual clues such as facial expression and sign language used by most HI peoples to interpret speech, for non-speech sound awareness they usually relied on bulky assistant infrastructure. For instance, vibration clocks are used to communicate tangible alarms during sleep. Wearable displays notify the HI people by encoding acoustic events into visual signals. Hearing aids and cochlear implants are essential to improve both speech and non-speech sound awareness. Nevertheless, numerous studies such as [7] report low usage satisfaction with hearing aids due to poor benefit, background noise, fit and comfort problems. For these individuals, missing these contexts such as fire alarm can not only be inconvenient but also life threatening.

In [8], technology has advanced to a stage where a powerful computational device can be easily carried in our pocket. Hence, one of the emergent tendencies is the growing need for sound recognition applications to be available on these portable devices.

So, the objective of this research is to construct Android-based EASR model. In this case, we perform offline model training and testing to select the best classifier among CNN and MLP and also the optimal parameter value of each classifier. To determine the effect of both frequency (MFCC, Chroma and SC) and time-frequency (Mel) domain features on the recognition performance of MLP different experimental analysis have done. The combination of MFCC, Croma and Mel feature vectors outperformed in the case of MLP. In the case of CNN, we used log-mel-spectrogram that represent the audio in bi-dimensional matrix and suitable for CNN's convolutional layer to filter useful feature and for the corresponding recognition purpose.

Generally, we perform different evaluation and comparison of each classifier with different configuration parameter values (feature extraction technique, number of hidden neurons, learning rate, dropout rate, activation function and others) to select the best one. Then, CNN classifier with its outperformed parameter values selected for the development of Android-based EASR model due to its good performance (93.8%) in the offline recognition of EAS. The trained feature matrix, corresponding class label and checkpoints

(weights and bias value for each training phase) of CNN classifier freeze to a compressed graph using protocol buffer in order to use it in Android-based EASR model.

Android-based EASR model constructed using sound recorder (using smartphone's microphone) for record the EAS occurred in real-time, audio pre-processing, feature extraction and recognition. Frozen graph is used for crosscheck unknown feature and the trained feature for recognition purpose in Android-based EASR model.

We evaluate the performance result of Android-based EASR model by using 20% of EAS dataset for 10 times trail and scored 89% recognition accuracy. In addition to this, the platform effect on the performance of EASR model (offline EASR in desktop and Android-based EASR in Android smartphones) evaluated. Thus, the Android-based EASR model performs less than offline EASR model due to noise interference when the smartphone's microphone record EAS and audio edge boundaries (audio overlapping). Finally, the running time of smartphone's battery when Android-based EASR model performs recognition, measured and evaluated. The result shows that the developed model can perform smoothly with the battery resource of the smartphone and it is good.

1.2 Statement of the Problem

Most of HI individuals in the Ethiopia gather information about their environment by their sight or communicate with other people's using sign language. This indicates that these HI peoples have no capability to recognize or hear the EAS which makes them challenged to their day to day life activities. For instance, they cannot evacuate a building on hearing a fire alarm [8], move appropriate position on hearing car horn, or take right action on hearing dog barking. In [7], such activities depend on human hearing which intelligently filters out sounds and quickly recognizes it thereby signalling the brain to take the next step.

Few investigations have been developed to create awareness about environmental sounds surrounding the HI individuals. By using sound visualization based on spectrograph and positional ripples [9]. However, this is impractical since it requires prior knowledge of the surrounding place (e.g. office); also, it is expensive in terms of equipment setup (array of microphones placed at certain corners in the room) and is also not portable (bound to the workplace environment). To address the issue on problems mentioned in [9], exploiting 3D coloured visualization of the acquired sound and displaying them as symbols of the recognized sound proposed by [10] using MFCC feature extraction technique and KNN and DTW classifier algorithm. But the 3D colour as well as the icons require further description and also it is for experienced user who have previous knowledge about the sound. And the other effort is investigated real-time ESR system for Android mobile devices was developed using MLP classifier and MFCC, ZCR, Spectral Centroid and Spectral Flatness feature extractors [8]. However, its recognition performance is lower as it is real-time system. Since lower recognition accuracy indicates the system misclassification rate is higher. This is due to the selected classifier algorithm and feature extraction techniques selected without any comparison regarding to its recognition accuracy and configuration parameters with other classifier algorithm and feature extraction techniques.

Hence, to address the above issues Android-based EASR model proposed to assist HI individuals. Essentially, the study is intended to answer the following research questions:

- What are the key feature extraction techniques that score highest possible recognition accuracy and noise robust for EASR?
- Which feed-forward ANN classifier fits and performs well in the recognition of EAS?
- To what extent recognition effectiveness is registered for Android-based EAS recognition?

1.3 Objectives of the Research

The objective of this research work is presented in terms of general and specific objectives which are presented as follows.

1.3.1 General Objective

The general objective of this study is to design and develop Android-based EASR model for hearing impaired individuals to minimize their risk of accident exposure.

1.3.2 Specific Objectives

To achieve the above general objective, the following specific objectives are drawn: -

- To pre-process audio dataset to make it suitable for further processing
- To extract representative features of audio dataset using feature extraction techniques.
- To apply classification algorithms to the individual extracted audio features.
- To select better performed algorithm and robust combination feature vectors for further Android-based implantation.
- To analyse and evaluate the accuracy and battery usage of Android-based EASR model under real-life environment.

1.4 Methodology

This research is experimental research, with the main aim to design and develop an Android-based EASR model using Android smartphones for the main purpose to reduces risk accident exposure of HI peoples. Like many other pattern classification tasks, EAS recognition is made up of three basic components. (i) detecting/sensing component: for measuring/recording sound event; (ii) audio processing component: for extracting the characteristic features of the recorded sound signal; and (iii) classification component: for recognition of the class of the sound event.

In audio-based EASR model, the detection (recording) is typically done using microphones. The audio processing component mainly deals with the extraction of features from the recorded audio signal. Feature extraction quantizes the audio signal and transforms into various characteristic features. This is done using different audio feature extraction techniques (frequency domain, time domain or a combination of time-frequency domain). The result of feature extraction in n dimensional feature vector usually representing each audio frame. A classifier then takes this feature vector and determines wat it represents (i.e. it determines the class label of the audio event).

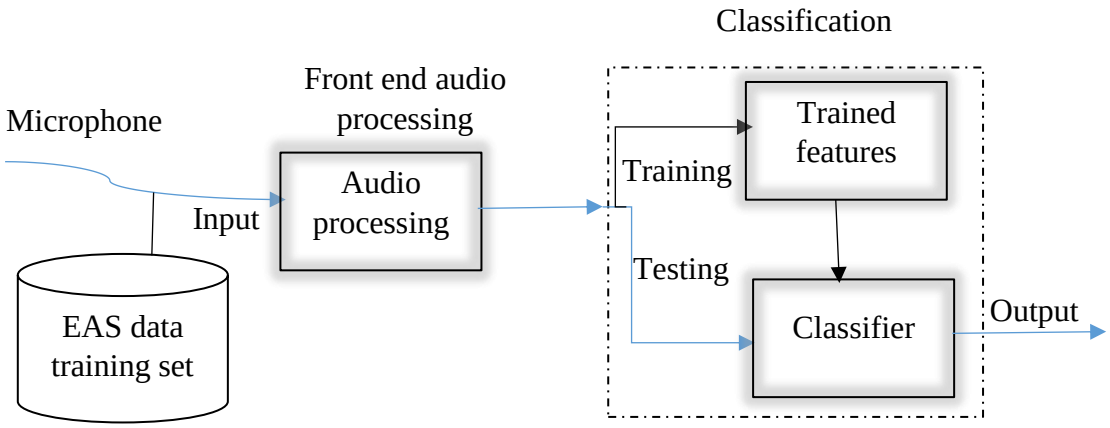

Figure 1-1: General architecture of EASR model

Figure 1-1, illustrates the general architecture of EASR system. In this architecture input represents the raw audio data whereas output represents the EAS class label.

The EASR model constructed in two phases: training and testing phase. During the training phase, the system receives its inputs from pre-recorded audio data training set and generates representative model for each audio event (EAS). In case of testing, the system receives its audio inputs directly from the smartphone's microphone. The testing phase uses the models generated during training phase for matching and determining the type of audio received by the microphone. The detail discussion of these two phases described in Chapter 4.

The main procedures that have been followed during the design of EASR model described as follows:

- First, thorough literature study of audio feature extraction techniques and classification algorithms is conducted. The main goal of this preliminary literature study is to preselect the best set of audio feature extraction and classification algorithms that can provide the highest possible recognition accuracy. This step is performed during the research topic selection.
- Offline training and testing is performed in order to compare the performance of each of the pre-selected techniques and then select the best one. Unlike speech and music recognition the research on EASR is not yet well matured. This makes it difficult to obtain standard procedure and well-organized information to determine the optimal audio feature extraction techniques and classification algorithms. Hence, it is important to make further experimental analysis to determine the best techniques. All experimental analysis (offline training and testing) are performed using Python codes. The training and testing results compare the performance of audio features and classification algorithms based on their recognition accuracy and computational speed. The offline training and testing result is discussed in Chapter 5 in detail.
- Android-based EASR model is developed using the best audio feature extraction and classification techniques chosen based on Python (offline) training and testing result, during Python experiment. The Android-based EASR model process the audio data online and provides in-time classification result. The developed Android-based EASR model is discussed in detail in Chapter 6.

1.4.1 Dataset Preparation

One major problem in the ESR field is the lack of a single consolidated audio database. Most papers in this field presents its results with its own dataset consisting of an arbitrary number of environmental sound classes which have different structures, sound type and quality. In the absence of a standard universal database like TIMIT [11] any performance benchmarking of the various approaches is not effective. Due to this, the data used to evaluate developed model, by recording different EAS using smartphone's microphone and collecting selected alarm sound from different environmental sound dataset from web. The collected EAS class type is same as to the recorded EAS. This is because the recorded audio data set is not enough for training the classifier to get the highest possible recognition accuracy.

This dataset is prepared by recording those EAS around Bahir Dar city in normal working environment in different scenarios: bus station, road, house and from the web by taking EAS only from existing environmental sound datasets. In addition to this from different films and different recorded audio clips. The recorded sound re-prepared to have same format as the audio data that collected from the web by loading and edit the format using *Format Factory* [12] application. This re-preparation enables the recorded audio compatible with the collected data from the web to reduce bias when developing a compressed dataset for the offline training and testing of this work.

1.4.1.1 Classes of EAS

We have collected 1,732 data samples with 5 different classes of EAS each with 2 to 5sec duration. Because, the training data becomes more the classifier model become more accurate. During data collection sound recording used different approaches such as record EAS very nearly to the occurred sound, distant to the occurred sound and with noisy environment. In this case, the distance of the occurred sound did not measure. However, we can say that the occurred sound heard loudly then the listener is near to the sound. While, the occurred sound heard to the minimum loud then the sound occurred to the distant place. Because, this approach useful in learning the environment to the EASR model used for as a classifier algorithm in this research design. This dataset has prepared for both training and testing. Even if, in the environment diverse unlimited number of sounds there,

the aim of this study is on EAS. Due to this, we limit our dataset to have 5 classes of EAS. The followings are sound classes in EAS-dataset.

1. Fire alarm
2. Car horn
3. Gun shoot
4. Engine idling
5. Dog barking

These sound classes are selected based on the impact that deliver on HI. And also, these sounds are important as alarm that warn people of hazardous situations called danger signals [2].

1.4.2 Implementation Tool

The sound recording conducting in Bahir Dar city during normal working hours (with some possibility of background noises) where various alarm sounds are heard. Unlike, previous works which uses high quality standalone microphones, it is intended to use commercial grade smartphone internal microphone for recording the sounds with a sampling rate of 44.1 kHz, mono-channel, 16-bits per sample. The model training and testing conducted using a computer with Intel(R) core (TM) i3_4000M CPU of 2.4GHz, 4.00GB (RAM), and 600 GB hard disk capacity with Microsoft window operating system that experimental offline training and testing and comparison be performing using the Python programming language.

The mobile application is developed in Android Studio with Android SDK (System Development Kits) and JDK (Java Development Kits) on windows platform. Android is a mobile operating system (OS) developed by Google, based on Linux kernel, and primarily designed for touchscreen mobile devices such as smartphones and tablets [13].

1.4.3 Evaluation Criteria of the Research

This study result is evaluated based on the different evaluation criteria that determine the recognition performance of the proposed study model. These criteria selected to measure how much the work give significance and reduce risk exposure of HI individuals. Based on these criteria, the proposed study model analysed and evaluated. In this work two most important evaluation criteria are selected: recognition accuracy and running time.

- Recognition Accuracy: recognition accuracy of the proposed study model is evaluated based on testing the system on different EAS occurred in the environment.
- Running time: this parameter evaluated by measuring the running time of the smartphone during running the Android-based EASR model.

1.5 Scope and Limitation of the Research

The main focus of this research work is to design and evaluate EAS recognition using Android-based smartphones for HI individuals. The study mainly focusses on the investigation of the implementation of the recognizer model that deals how to it minimize risk exposure of HI individuals surrounding their environment. This is basically investigated based on selecting the outperformed feature extraction technique (MFCC, Chroma, Mel and SC) and feed-forward ANN classifier algorithms (MLP and CNN) by offline training and testing (comparison and analysis). In the case of feature extraction techniques, audio feature extraction techniques are categorized into frequency domain or time domain features. From these, MFCC, Chroma, SC and Mel selected based on their performance in the representation of characteristic features of EAS (better recognition accuracy, robustness to noise, low computational complexity: i.e. simplicity))

Since, there is no organized environmental sound dataset, for training and testing; in real time environmental alarm sound using Android-based smartphones, the proposed system will use limited number of sound classes. The performance of the proposed system is evaluated only for frequently occurring alarm sound that happen on the environment which are very crucial for the HI individuals on their day to day activities.

In addition, the designed model in this study do not able to recognise the source direction of the occurred environmental alarm sound. We will proceed it in the future works because knowing the direction of occurred sound (EAS) have great significance.

One of the main goals of this study is to ensure the Android-based EASR is location-independent. However, the recognition performance of the model in noisy environment and on simultaneous sound occurring situations decreases slightly.

1.6 Significance of the Research

Auditory cues provide people with information about events outside of their field of view and can so help detecting potential hazards (e.g. EAS). People with severe hearing inabilities, i.e. HI, often cannot or can only partially benefit from auditory cues, which can lead to a lowered quality of life. Even though hearing aids are available, not everyone can benefit from them. To provide HI people with auditory cues from environmental sounds different assistive devices have been introduced, usually for use at home [9]. A flexible and outperformed Android-based EASR model is presented in this contribution. The model of this work implemented on Android platform device like mobile phones and tablets, which are more useful for practical applications of being able to move anywhere thereby supporting the freedom of HI individuals.

Hence, the following significances are provided by this effort:

- Allows to create awareness of environmental sounds especially those early warning events (EAS) to those HI individuals.
- Allows to provide cognitive assistance in taking of decisions by giving hints, suggestions and reminders about EAS occurring around ones surrounding.
- To further the research in area of environmental sound recognition and apply the results to local contexts, among others.

1.7 Organization of the Research

This thesis is organized in to six chapters, as sown below.

Chapter One: Describes an overview of the general background of the study, the statement of the problem, objective of the study, the scope of the study, the methodology of the study, significance/contribution of the study that provides to the user.

Chapter Two: Provides detail literature overview under the study domain area, background and principles in digital audio analysis, basic sound processing techniques for environmental alarm sound recognition and review of previous efforts related to the research domain area.

Chapter Three: Covers theoretical aspect of all technique and algorithms used to design the research work.

Chapter Four: Covers all technique and algorithms used to design the research work, and it also provides the overall architecture the design made in the study.

Chapter Five: Discusses the experimentation activity undertaken to implement the methods and techniques in chapter three. The experiment conducted, the dataset used and experimental setup, comparison of feature extraction techniques and classifier algorithms included. Based on the experiment result the selection of outperform feature extraction techniques and classification algorithm used for Android-based system development also comprised.

Chapter Six: Based on the selected feature extraction techniques and recognition algorithm Android platform experimental activities performed, result of the experiment presented, different aspect of performance measurement undertaken.

Chapter Seven: This chapter discusses conclusions decided from observing the experiment conducted in this study and the derived recommendations for future research work directions in the domain area.

CHAPTER TWO

2. Literature Review

2.1 Introduction

This section discusses about existing literatures that investigated with the recognition of environmental sound. The challenges and shortcomings associated with environmental sound recognition methods and techniques which include different algorithms and feature extraction techniques used in previous efforts in the area of environmental sound recognition is illustrated.

In [14], Sound is generated when an object (such as bumping cars) provides a riot in the density of the medium in which it resides (usually air). This riot proliferates through the medium. When the riot reaches a human ear, it is converted into electrical signals that the brain interprets as sound.

Surrounding around us is composed of infinite sound classes in addition to music and speech, due to it consists of numerous object disturbance. These sounds could be things like (dog barking, rain, rooster, sea waves) and artificial sounds (helicopter, siren etc.,) [8] called ES. In addition to this sounds like fire alarm or gunshot which is called environmental alarm sound. They could befall not only in controlled environments like the workplace or home but also in public places such as in entertainment or whilst walking in the street.

[3] investigated ES based situation analysis has been studied for building autonomous agents that are able to understand their surrounding environment through the use of both audio and visual information , [15] for automatic detection and recognition of impulsive sounds in noisy environment, [14] for recognizing ES in an autonomous mobile surveillance robot for surveillance purpose and others. In addition to this ES detection and recognition investigated in mobile computing for different purpose such as [8], [10], [11], [16].

According to Angelos Pillos et al. [8] the development of ESR in Android OS technology was proposed with different approach and methods [8]. This technology includes acoustic

event detection, feature extraction and classification for determination of most suitable category of the event.

Over the last few years, researchers used environmental sound recognition feature extraction methods such as frequency domain feature extraction (example: Mel frequency cepstral coefficient (MFCCs), spectral centroid, spectral flatness and spectral entropy (SE)); temporal domain features (example: zero crossing rate (ZCR), short time energy (STE)) and time-frequency features (example: matching pursuit (MP), Log-Gabor filters and Spectrogram). And also, several environmental recognition models proposed by different researchers such as K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), Artificial Neural Network (ANN). Several efforts also apply the combination the above feature extraction technique and recognition model, however SVM, KNN and ANN has been the most successfully recognition models [17].

The rest of this chapter devoted about to discuss background and basic principles used and literatures on environmental sound recognition basic procedures reviewed.

2.2 Background and Principles in Digital Audio Analysis

This section provides a detail background information and fundamental principles and techniques used in environmental sound processing and investigation.

The standard method of sound analysis at spectral level, is based on classical Fourier analysis to the whole audio signal. Even though, in audio signal an exact definition of Fourier transform cannot directly apply due to audio signals are time varying or non-stationary in the real world and, indeed, all their meaning is related to such time variability [18]. Hence, it is important to develop sound analysis techniques that allow to grasp at least some of the distinguished features of time-varying sounds, in order to ease the tasks of audio analysis such as feature extraction.

To address this problem audio signals split to a smaller sequence of audio segment called frames. This audio segment assumed to be pseudo stationary that the representative features could be extracted and further processed. The process of dividing audio signal into frames called framing. The length of each frame range between 10 and 50ms which audio segment

feature assumed not able to change significantly. Audio processing such as Fourier transform and feature extraction performed frame by frame. But, audio processing only in individual frames could not effective because there is discontinuity (sharp corner) between consecutive frames. To eliminate this discontinuity each frame should be multiplied by smoothing function such as Hanning or Hamming window [16] before we applying Fourier transform called windowing. This is the process of subdividing frames by overlapping some percentage such as 50% or 25% with each other called windowing.

The process of frame by frame analysis is known as short-time signal analysis. In the previous efforts there are various short-time signal analysis such as Short-Time Fourier Transform (STFT), Discrete Wavelet Transform (DWT) and Winger distribution (WD) [11]. Due to its computational simplicity Sort-Time Fourier Transform (STFT) is the most sort-time signal analysis technique [11]. Hence, we used STFT in our work. In this section we focus on review literatures about STFT, analysis parameters such as window type and length.

2.2.1 Short-Time Fourier Transform

The Short-Time Fourier Transform (STFT) is an influential general-purpose tool for audio signal processing. It defines a particularly useful class of time-frequency distributions which specify complex amplitude versus time and frequency for any signal. It performs a Fast Fourier Transform (FFT) analysis on short windows time to represent and analyses frequency content of the signal. We selected STFT for audio analysis because:

- ✓ For non-stationary signal processing and analysis, it is the simplest and effective technique.
- ✓ It has the capability to represent the spectra of the signal with spectral profiles that varying over time.
- ✓ It allows adaptive and other non-linear signal modifications.
- ✓ Time-Frequency (T-F), i.e., STFT analysis is what the human brain does.
- ✓ It allows processing and signal modification directly in the Time-Frequency domain

After a sliding window applied on a successive frame, Fourier transform operation is then applied. In other words, we can think of STFT as multiplying audio signal x(n) by a short-time window that is centred around the time frame n. The segment of the signal contained in the window is analysed using the Discrete Fourier Transform (DFT), which implies the evaluation of the Time-Frequency representation at a set of discrete frequencies. Because Applying the DFT over a long window does not reveal transitions in spectral content. Using the Fourier series representation, we have Discrete Fourier Transform (DFT) for finite length signal. DFT can convert time-domain discrete signal into frequency-domain discrete spectrum. Equation 2.1 deals the mathematical definition of STFT for finite set of N points. Assume that we have a signal$\{x[n]\}_{n=0}^{N-1}$. Then the DFT of the signal is a sequence $X_m(k)$ for k = 0, …, N – 1

$$\mathrm{X_m(k)} = \sum_{\mathrm{n=0}}^{\mathrm{N-1}} \mathrm{x(n).\, w(m-n).\, e^{-j2\pi nk/N}} \qquad 2\text{-}1$$

Where

X(n)= input signal at time n

w(m)= length of m window function (e.g., Hamming)

$X_m(k)$= DFT of windowed data (frame) centred about time n.

If we assume P to be the overlap size (in terms of number of samples) between successive frames, then we can compute the number of frames as follows

$$Number\ of\ Frames = \lfloor \left(\frac{N-m}{P}\right) \rfloor + 1 \qquad 2\text{-}2$$

Where $\lfloor \rfloor$ is a symbol for rounding down a fraction value to the nearest integer value, called flooring. The time taken to evaluate a DFT on a computer depends principally on the number of multiplications involved. DFT needs N^2multiplications. FFT only needs $N \log_2(N)$. It is relatively slow computation process. Hence, a very efficient algorithm for computing the DFT required. So that Fast Fourier Transform (FFT) is used. The fast Fourier (FFT) is an optimized implementation of a DFT that takes less computation to perform but essentially just deconstructs a signal. Figure 2-1 shows the process of STFT. Most researchers used FFT as a short time Fourier transform [8].

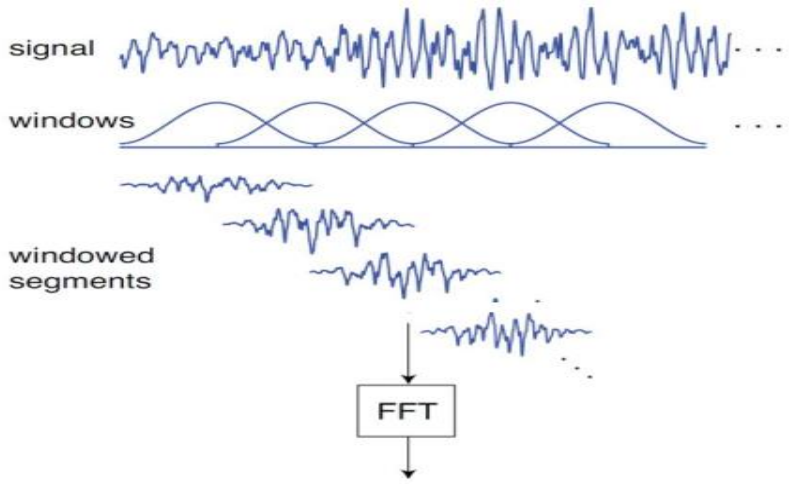

Figure 2-1 STFT processing

This work incorporates the concept of Fourier transform for processing environmental alarm sounds.

2.3 Sound Processing Techniques for Environmental Sound Recognition

Environmental sound recognition is the process of capturing, detecting, analysing and recognizing sound events that occur in the environment. Environmental sound recognition is exercised by many researchers to analyse environments, e.g., in surveillance [19], monitoring of people in need of care [20], or detecting, localizing, tracking and classifying sources of military interest in real time [21]. Recently, environmental sound recognition becoming hot research area with applications including audio surveillance, healthcare monitoring, urban sound analysis, multimedia event detection and bird call detection [22]. Environmental sound recognition is a technique for understanding or recognizing the context of sounds in the environmental surroundings by passing through successive audio signal processing techniques.

Now a day, sound processing technique for environmental sound recognition, which is able support to automatically detect and analysis unstructured nature of acoustic sound events and has the ability to distinguish the exact class or label of that sound and its context. Several experimentations has been performed using environmental sound, such as for understanding and predicting the context surrounding of an agent [3], understanding of a scene (context) of surrounding an audio sensor [21], for audio surveillance and security

application [19], audio forensic [23], deaf assistance using android smartphones [8, 10], user context recognizer [16].

Similar to other sound processing tasks, the basic steps of environmental sound recognition processing can be broadly classified in to three phases: pre-processing, feature extraction and recognition may be the post processing in few previous efforts.

2.3.1 Pre-Processing

It takes place at the raw audio signal to pre-pare it to the format that suitable for extracting useful information. This pre-processing performed into two categories. The first one is for training the model that audio dataset pre-processed and the other is for performing Android-based test the model that developed in mobile device. Now we discuss for training the model that takes place on computer.

Framing: the raw audio signal is non-stationary signal, which means its statistical value is not constant across over time and it is one dimensional representation. So, it is necessary to split it to smaller segments that seems to be stationary. This phase occurred when splitting audio signal into 20 to 50ms duration successive short segments with various sampling rate such as 16kHz [11] or 44.1kHz [8].

Windowing: In framing phase each frame has a sharp corner. Which means that there is a discontinuity between successive frames. To remove this sharp corner between frames applying windowing function on the whole frames of a signal is required. The window of audio frame takes place by multiplying the value of the frame of the signal at time n, $S_{frame}[n]$, with the value of the window at time n, $S_w[n]$ [11]. Usual windowing functions are Hamming window [11] and Hanning window [8].

$$Y[n] = S_{\mathrm{frame}}[n] \times S_w[n] \qquad 2\text{-}3$$

Then apply a short time Fourier transform (STFT) [11]. A typical STFT as mentioned earlier, is Discreet Fourier Transform (DFT) and Fast Fourier Transform (FFT). FFT is nothing but it performs DFT computation in efficient and effective manner. Applying STFT on successive frames convert short frame audio signal to a time-frequency representation that makes it suitable for extracting useful features.

2.3.2 Feature Extraction

Feature extraction is the crucial stage of audio signal processing. This task takes place to extract useful representative set of features from the audio signal frame. Two broad categories of features extracted in feature extraction stage. Those are temporal and spectral domain features. The former, temporal (time- domain) features extracted after windowing process takes place, without additional audio processing required such as STFT takes place. The latter, spectral (frequency domain) features extracted after DFT takes place. Several researchers used different feature extraction algorithm (both time and frequency) domain features.

- ✓ Time domain features used previously: zero crossing rate (ZCR) [8, 24], short time energy (STE)
- ✓ Frequency domain features: MFCCs [8, 10, 16, 17], spectral flatness [8], spectral centroid [8, 11], spectral entropy (SE) [11], log Mel energies [17, 25], log spectrogram [26], SC [27, 28], Chroma [29, 30, 31]
- ✓ Time-frequency features: Matching pursuit (MP) [5], Log Gabor filters-based [19], Mel [32, 33]

2.3.3 Recognition

Once the audio signal is processed and useful feature vectors extracted then the next step is recognition. So, this process is the final task performed in audio signal processing which aims to classify the input feature vector to its correct label or to get the final output of the proposed model. The recognition task performed by training the model with training audio dataset feature vector and then perform testing or put testing dataset feature vector to the trained model. The recognition models used in the previous effort from simple recognition model up to advanced (artificial neural network) models. Most principal classifier algorithms for environmental sound recognition are:

- ➢ Gaussian Mixture Model (GMM) [5, 11]
- ➢ KNN [5, 10, 11]
- ➢ Support Vector Machine (SVM) [16]
- ➢ Multilayer Perceptron Neural Network (MLP) [8, 34, 35]
- ➢ Convolutional Neural Network (CNN) [26, 36, 34]

- Deep Neural Network (DNN) [17]
- Bidirectional Long Short-Term Memory-Recurrent Neural Network (BLSTM-RNN) [25]

2.4 Previous Efforts on Environmental Sound Recognition

Table 2-1: Summery of environmental sound recognition studies

Author	Title	Techniques used in the work	Obtained Result	Limitation of the work
Alhabbash et.al, [10]	Sound visualization for deaf assistance using mobile computing	MFCC for feature extraction; KNN and Dynamic Time Wrapping (DTW) for classification	-	The 3D colour as well as the icons require further description and also it is for experienced user who have previous knowledge about the sound.
Angelos Pillos et.al, [8]	A real-time environmental sound recognition system for the android OS	MFCCs, ZCR, Spectral flatness, and spectral centroid for feature extraction; MLP for classification	74.5%	Lower performance as it is real-time system. Since lower recognition accuracy indicates the system misclassification rate is higher. Techniques used in the work selected without any comparison regarding to its recognition accuracy and configuration parameters with other
Wai-ling Ho-Ching et.al, [9]	Can you see what I hear? The Design and Evaluation of a Peripheral Sound Display for the Deaf	Spectrogram design and Positional replies design	-	It is impractical since it requires prior knowledge of the surrounding place (e.g. office); also, it is expensive in terms of equipment setup (array of microphones placed at certain corners in the room) and is also not portable (bound to the workplace environment)

Table 2-1 shows literatures have been proposed for EAR with their recognition accuracy, feature extraction and classification approaches used and limitation of each study.

Among these issues, how to deal with EAS detection and recognition is the particular focus in this thesis work. In addition to that, this research work adopts the necessary techniques and methods from these research efforts to achieve its mentioned objective.

CHAPTER THREE

3. Machine Learning Algorithms for EAS Recognition

Sound classification is the process which takes a sound sample as input and gives the respective class label as output. There are many types of classifiers in machine learning paradigm that can perform such a task. Machine learning can be used to effectively classify sound based on the source. In the case of this work, recognition can be defined as learning different features of EAS and based on the trained features predicting the corresponding class of unknown EAS. Hence, the proposed approach is to develop EAS recognizer for HI peoples to support them in the way that they can able to react with their environment accordingly. The recognition process (i.e. detect and classifying EAS) is done in either MLP or CNN classifier based on the recognition performance result during the offline training and testing. In other words, the training and testing takes place for both classifiers and then by comparing the performance of them and thus the good performing classifier is selected. In the literature several environmental sound recognition methods are mentioned in section 2.3. In ANN environmental sound recognition/classification algorithms categorized into three groups based on their learning behaviour [37], namely; supervised learning (supervised classification), semi-supervised learning (semi-supervised classification) and unsupervised learning (unsupervised classification). Supervised classification algorithm recognizes patterns based on learning each training instance that associated with a set of labels [37]. The network provided the desired output for the given input. These supervised classification algorithms include HMM, GMM, K-NN, SVM, ANN, BN and Dynamic Time Wrapping (DTW). Semi-supervised classification algorithm recognizes patterns based on learning a small amount of labelled data associating with a huge amount of unlabelled data [37]. It is applicable when the need for labels during the training is critical and the amount of testing data is huge. Unsupervised classification that the recognition performed based on measuring the similarities between inputs and represent them in efficient way due to the absence of neither explicit teacher or labelled samples [37]. Unsupervised recognition algorithms widely used in clustering and mixture modelling.

3.1 Construction of The Recognition Model

Various environmental sound recognition models are proposed [11] for classifying types of sound from the occurred sound event in the environment. Namely: HMM, ANN, SVM, K-NN, DTW, GMM are the most widely used techniques. In this study we selected a effective and powerful classifier; ANN because of it has strong abilities to explore inherit and hidden patterns through huge amount of training data [25]. As previous efforts stated that it is massively parallel interconnected networks of simple (usually adaptive) elements and their hierarchical organizations which are intended to interact with the objects of the real world in the same as biological nerves system do [22, 26, 38]. ANN are straight forward model of real-world system comprising of a large number of neurones like processing elements and a large number of weighted connections between the elements [17, 25].

Because of MLP neural network and CNN are effective techniques for environmental sound classification [17, 25], they are similar for solving non-linear problems, by learning the given data samples and then recognize the unknown data by using the trained feature. However, they are not use the same approach and network model. MLP use feed-forward fully connected layers of neurons and different activation function for solving non-linear complicated problems. While, CNN instead of fully connected, it uses convolutional layers and a group of pooling layers with activation function. CNN have ability to extract features from raw data (EAS) using the convolutional layer while MLP perform recognition by learning the pre-extracted features of EAS and can able to recognize unknown one.

In order to simulate, compare the performance of recognition model and to select the better classifier for the main aim to develop smartphone-based recognition model, the proposed approach MLP and CNN are used.

3.1.1 Multi-Layer Perceptron (MLP) Classifier

MLP is a feed-forward layer network of artificial neuron, where the data circulates in one way, from the input layer to the output layer [34, 35]. MLP is one of Artificial Neural Networks (ANNs), are mimics of human brain to process real world systems comprising of large number of neurons like processing elements and a large number of weighted connections between elements. Neural Network approach applied for recognition of environmental alarm sound because it adapts some knowledge from the data and can able to generalized. MLP is an appropriate algorithm and classification model. The basic components [34]

- **Learning Paradigm**: it is a model of the environment in which the neural network operates to learn or train the system. Hence, MLP is a supervised learning, the desired output is available for all the samples needs to be trained.
- **Network Graph Architecture**: it is the topology of the network that describes the pattern/ layer of connections between neurons. As depicted in the Figure 3-1, MLP comprised three layers: input layer, hidden layer and output layer.
  - Input layer: input layer of the model is the first layer and its main task is receiving input data/features that represent audio signals from the environment and corresponding weights feed to the network (hidden layer).
  - Hidden layer: layers of neurons other than input layer and output layer, its purpose is processing the input feature receives from the input layer. Computing weighted sum of the input features and its corresponding weight performed at the hidden layer. Hidden layer process input features by calculating the weighted sum of input data being activated by activation function and forward the output to the output layer. In this work we used one hidden layer neural network model.
  - Output layer: the output layer receives the output of the hidden layer output and by using activation function provides the output of the network (i.e. the output layer provides the classification result of the network). Hence, the output of recognition model prediction presented in output layer.

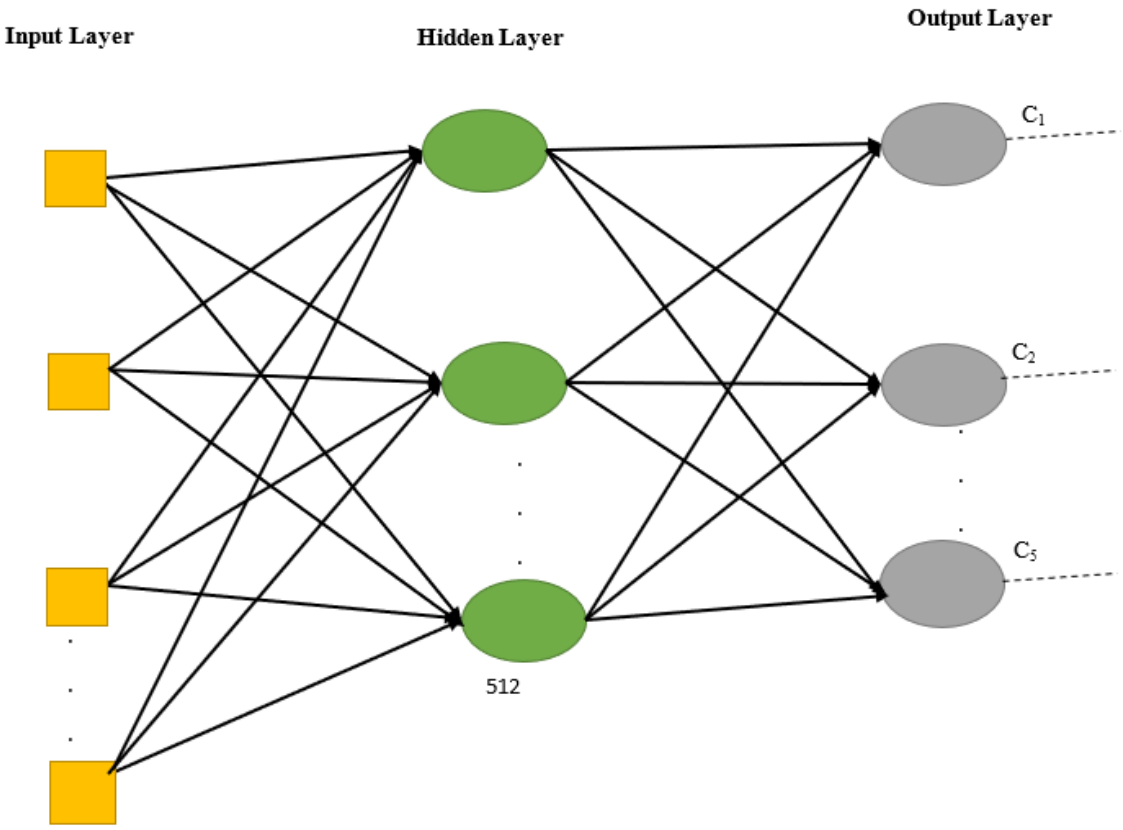

Figure 3-1:Multilayer Perceptron architecture

- **Learning Algorithm**: learning in MLP means a method that determining the adaptation of weights between the connections of two neurons. MLP uses a backpropagation algorithm for adjusting its weights to a sequence of training samples during a learning phase. Backpropagation algorithm is a supervised learning technique that performs four main tasks: initializing weights, feed forward, backpropagation of errors and weight update to minimize error rate of the network [17, 25, 34].
- **Activation Function:** is a fundamental block of neural network that determine the neuron to be activated or not by calculating weighted sum and further adding bias with it. The activation function introduces non-linearity of the output of the neuron. In this thesis work we used Rectified Linear Unit (ReLu) activation function in the hidden layer and SoftMax activation function in the output layer.
    - Rectified Linear Unit (ReLu): It has become very popular in the past couple of years [36]. It calculated as

$$R(x) = \max(0, x) \begin{matrix} 0, x < 0 \\ x, x \geq 0 \end{matrix} \qquad 3\text{-}1$$

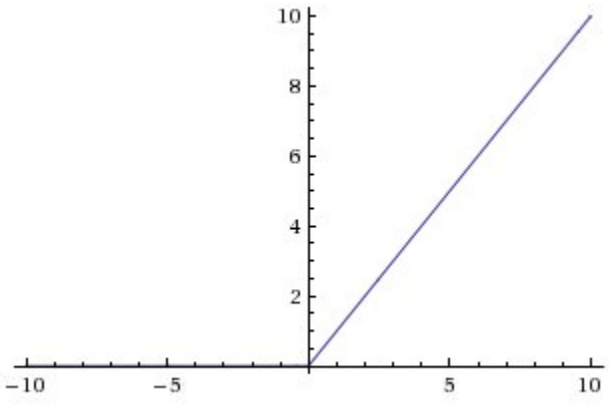

Figure 3-2: Rectified Linear unit (ReLu) activation function sample graph

Hence, seeing the mathematical form of this function we can recognize that it is very simple and efficient. It should only be used within the hidden layers of the neural network model. It can able to avoids the two major problems arise in deep learning, saturation and vanishing of gradients of neural network [36]. Because ReLu promotes sparse representations inside the network due to the hard 0 for negative values of x . In this study we used ReLu activation function in the hidden layer for the recognition model.

- SoftMax: The SoftMax activation function is mostly used to normalize the output of each unit to be between 0 and 1, just like a sigmoid function. But it also divides each output such that the total sum of the output is equal to 1 as shown:

$$\begin{bmatrix} 1.2 \\ 0.9 \\ 0.4 \end{bmatrix} \longrightarrow SoftMax \longrightarrow \begin{bmatrix} 0.04 \\ 0.34 \\ 0.62 \end{bmatrix}$$

For a vector x, its SoftMax is defined for each of its components x_j as:

$$\text{SoftMax}(x_j) = \frac{e^{x_j}}{\sum_m e^{x_m}} \qquad 3\text{-}2$$

Such that $SoftMax(x_m) > 0 \ \ \forall m$ and $\sum_m SoftMax(x_m) = 1$.

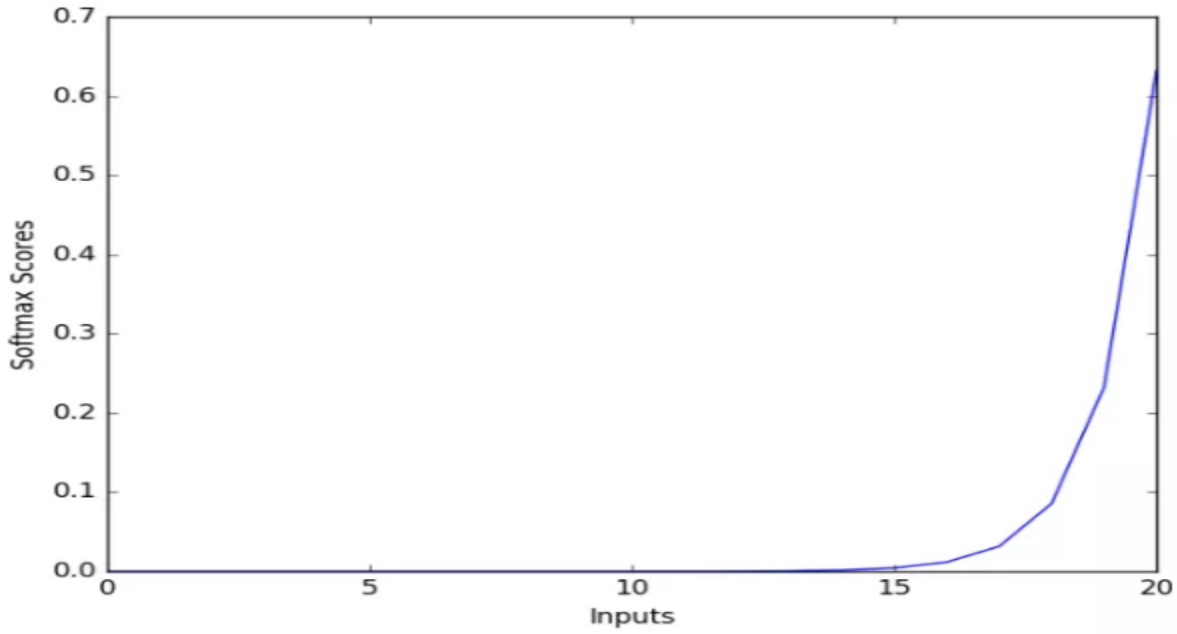

Figure 3-3: SoftMax activation function sample graph

The output of SoftMax function is equivalent to a categorical probability distribution shown in Figure 3-3, it tells us the probability that any of the classes are true. The SoftMax function determines a discrete probability distribution for K classes [39] as shown in Equation 3-2. For this reason, we used SoftMax activation function on the output layer of our environmental alert sound recognition neural network model.

3.1.2 Convolutional Neural Network (CNN)

CNNs are a derivative of standard MLP neural networks optimized for two-dimensional pattern recognition and not only image but also audio data [34]. Instead of fully connected hidden layers, the CNN introduces a special network structure, which consists of alternating named convolutional and subsampling layers. In the case of audio data, CNN (2D CNN) analyzes in chunk level rather than frame level [40]. The model general overview comprised the following components.

- **Learning Paradigm:** the learning paradigm of CNN is similar with MLP; which is based on providing the desired output available for all the samples needs to train by the model.

- **Network Graph Architecture:** it is a network topology that describes how the neurons connected each other in the neural network. The graphical architecture of CNN is a subset of a feed-forward network, which is each neuron send data to the next neuron in the forward direction within the network. A typical CNN consists of a number of different layers stacked together in a deep architecture: an input layer, a group of convolutional and pooling layers (which can be combined in in various ways) and a fully connected output layer.
    - Input layer**:** is a collection of neurons that aims to receive input pattern from the environment and provides to the next layer (convolutional).

      Convolutional layer: contain neurons that take their synaptic inputs from a local receptive field and generate feature maps. The main characteristics of the CNN is that the weights of the neurons within the same feature map are shared. This characteristic allows to have replicated units share the same configuration. Hence, features of the audio data can be detected regardless of its frequency or time position and increases learning efficiency by reducing the number of parameters being learnt. In our work we used two convolutional layers as depicted in Figure 3-4 below.

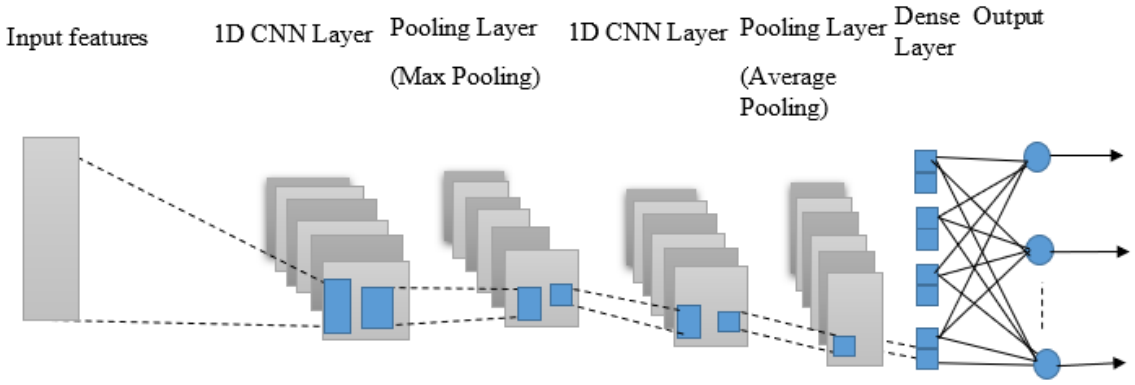

Figure 3-4: Convolutional Neural Network architecture

    - Pooling layer: contains neurons that involved between sequential convolutional layers; and between convolutional and fully connected output layer for the main aim to subsampling each feature map, hence reduces it dimension [34]. We used

two types of pooling layer: Max pooling and Average pooling between two convolutional layer and convolutional layer and fully connected output layer respectively.

- ✓ Max pooling: perform dimension reduction of feature map by simply extracting the highest value from a window that slides along each feature map.
- ✓ Average pooling: reduces the output of convolutional layer by computing the average result of the generated feature map.

- Output layer: ensures the classification of the input audio signal. In this layer neurons are fully connected and have a unique set of weights so they can detect complex features and perform classification.

- **Learning Algorithm**: Naturally CNN is a sub set of feed-forward neural network, learning algorithm of CNN is depend on Backpropagation rule that allows the information flow backwards from the output to the next layer through the network. For the main aim to compute the gradient and to adjust weights of the hidden neuron to minimize the difference between the predicted output and actual output.
- **Activation Function**: neurons in the CNN operates its processes by applying a non-linear filtering to a weighted sum of its inputs, with the weights being the network parameters learned during training. We used ReLu activation function for the convolutional layer and SoftMax activation for output layer similar to as MLP (Section 3.1.1).

CHAPTER FOUR

4. Design of Environmental Alarm Sound Recognition System

4.1 Introduction

To recognize environmental alarm events from different environmental situations several sound processing phases are conducted. The process of sound recognition takes several complicated phases starting from detecting sound from the surrounding to the output text description of occurred sound. In this study EAS recognition model developed into two platforms. The first, is computer based EASR model developed, in especial target to train and test the model through necessary EAS audio dataset. Because performing model training in Android based smartphone is impossible due to its resource (battery power, CPU usage and memory storage) limited device. The second, is based on the output of computer based EASR model (trained model and feature vectors), then the mobile based EASR model is developed. Hence, this chapter illustrates the overall architecture of EASR model design and development.

4.2 Environmental Alarm Sound Recognition Processes

Three generic steps/activities are performed for EAS recognition processes. First, audio signals pre-processed through three basic steps i.e. framing, windowing and short time Fourier transform to make ready it for extracting useful feature vectors from it. Second, feature extraction process applied on short segment of audio frames to extract useful representative feature vectors from each frame of audio signal. The feature extraction performed in this study focusing on extracting spectral (frequency domain) features using MFCCs, Chroma, SC; and time-frequency feature using Mel. Because, the first three frequency-based features extract the power spectrum of the input signal in each frequency band. It enables to discriminate one input sound from the other easily [21]. Mel is two-dimensional feature extraction technique that represents both frequency and time information of audio signal, which allows to detect and recognize effectively such flat spectrum EAS [32]. Finally, EAS classification is takes place using CNN and MLP ANN classifier algorithm that able to adapt to the difference ways of EAS situations. The details of these steps in this study are depicted in Figure 4-1 below.

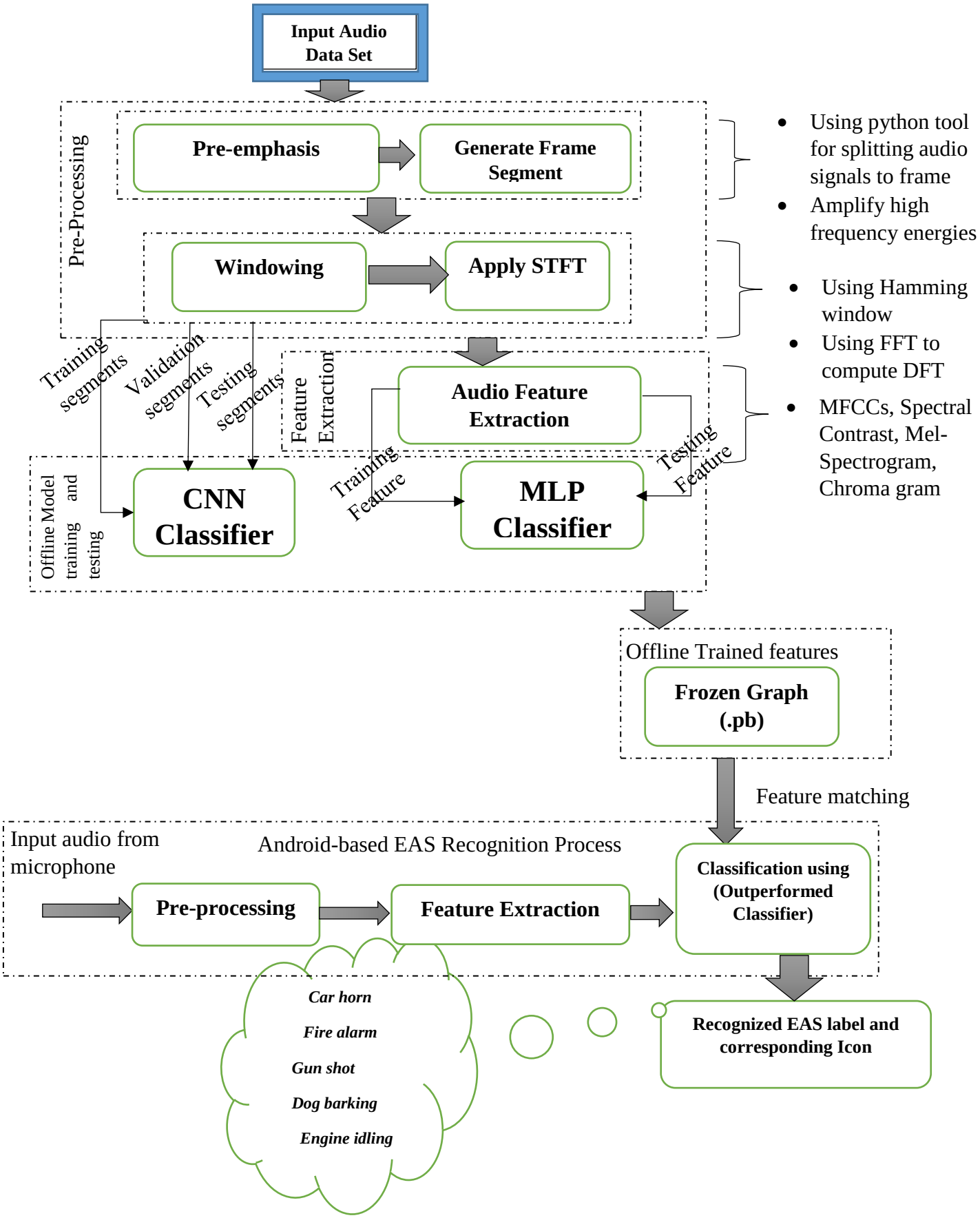

Figure 4-1 :Architecture of the proposed model for EASR

4.2.1 Pre-processing

The input audio data for EASR captured is in Wav (Waveform) audio format. Because it is popular that it able to kept the file first digital copy completely uncompressed, high quality and to avoid introducing artefacts (e.g., from the MP3 format) [5]. After capturing and collecting EAS in mobile standalone microphone, the next pilot phase is sound pre-processing. It improves the sound in the ways that increase the chances for success of other further processes. It is very important for other sound processing operation; feature extraction. Several activities performed in sound pre-processing such as splitting continuous non-stationary audio signal to a stationary short audio segment (framing), amplifying signal energy which is in high frequency band, removing sharp corners of each frames (windowing) and converting a one-dimensional audio signal or time domain frame to time-frequency format (compute STFT on each frame [41]).

The basic operations performed in pre-processing are: pre-emphasis, framing, windowing and compute short-time Fourier transform.

4.2.1.1 Pre-Emphasis

Usually audio signal should pre-emphasised before any further processing. The audio signal energy distribution is higher at lower frequency band than higher frequency. So, in higher frequency there is lower energy. But the human hearing system is sensitive at higher frequency than lower frequency which is usually greater than 1kHz. To achieve good recognition accuracy, it is necessary to pre-emphasis/amplifying high frequency band energy of the audio signal before any further processing applied [21, 42]. Therefor pre-emphasis task applied on audio signal on the main target for magnifying the portion of spectrum energy of the signal that is in higher frequency band, while leaving the lower frequency energy in its original state [42] or spectrally flatten the signal. The process of pre-emphasis computed using the formula with time domain audio signal input x(n) and amplifying factor $\boldsymbol{\alpha}$, it is usually $0.9 < \alpha < 1.0$ [42] , is given below.

$$y(n) = x(n) - \alpha . x(n-1) \qquad 4\text{-}1$$

4.2.1.2 Framing

As mentioned in Section 2.3 audio signals are continuous and non-stationary which is its statistical values is not constant over time. This property of the signal makes difficult to get audio signal representative feature vectors. For this reason, splitting the signal to a smaller consecutive segment/frames which is enough to describe features [18] . In this thesis specifically in the computer based EASR we used 50ms frame length for framing with 44.1kHz sample rate and 16-bit resolution [8, 10]. Features are extracted from each frame and this set of features is used as one instance of training or testing. Some sound events are short-lived (e.g. gunshot) as compared to other longer events (e.g. fire alarm). So, the frame length should be considered audio events that has small duration as well as long duration. The sampling rate is selected because of most of previously investigated efforts in environmental sound recognition is used 44. 1kHz. The process of splitting an audio signal to a smaller portion/window is known as framing. Feature extraction and other further processing is performed frame by frame. The statistical value of the spectrum of the signal per frame assumes to be constant over time. Figure 4-2 shows framing, each frame has 50ms length with 25% overlapping.

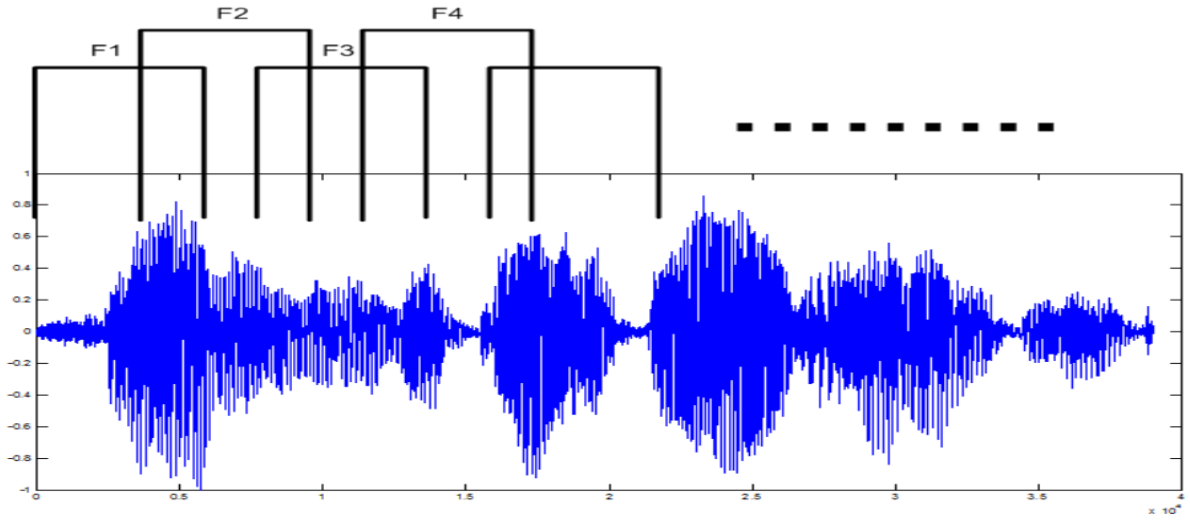

Figure 4-2: Framing process of audio signal

4.2.1.3 Windowing

When audio signal fragmented, each frame has a sharp corner. This leads to there a discontinuity between successive frames. Extracting a feature vector on discontinuous frames loses some useful descriptive features. So, to reduce the discontinuity between successive frames and to avoid sharp corners at the end of each frame, windowing operation takes place [21]. So that the signal after windowing becomes continuous, each frame shows the characteristic of periodic function. In this study we used Hamming windowing function [8] with 25% overlap to smooth the edge of each frame segment. If the window length is too small, the long-term variations in the signal would not be well captured by the extracted features, and the framing method might chop events into multiple frames. On the other hand, if the window length is too large, it becomes difficult to locate segmental boundaries between consecutive events and there might be multiple sound events in a single frame.

The windowing operation should be performed before STFT operation and it is applicable for only training purpose. As mentioned in Equation 2.3 the windowing of audio frame computed by multiplying the value of the frame signal at time n, $S_{frame}[n]$, with the value of the window at time n, $S_w[n]$ where the value of $S_w[n]$ is defined as shown below.

$$S_w[n] = 0.54 - 0.46\cos(\frac{\pi n}{N}) \qquad 4\text{-}2$$

Figure 4-3, shows windowing process framed segment of audio signal using Hamming window function.

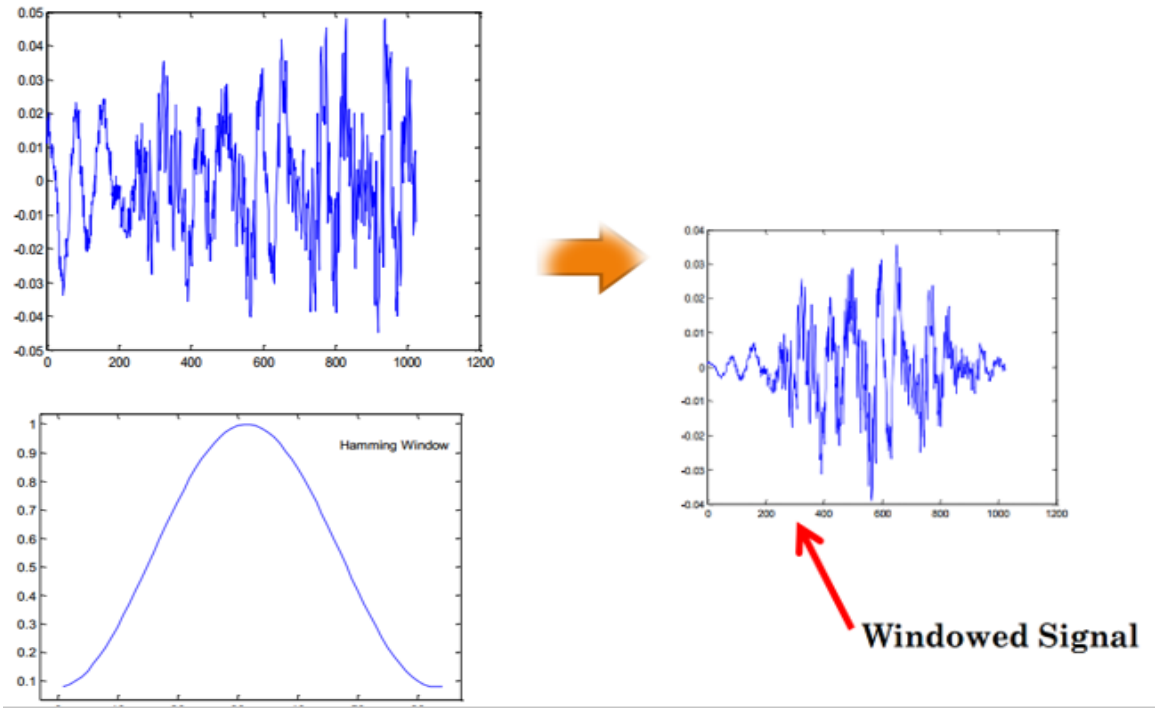

Figure 4-3: Windowing process using Hamming window function

4.2.1.4 Short Time Fourier Transform

Before we got into short time Fourier transform (STFT), we should know first: what is Fourier transform? Fourier transform is a tool to transforming a wave function or signal from a time domain (how a signal change over time) to frequency domain (how much of the signal lies within each given frequency band over a range of frequencies). As mentioned in section 2.2.1 STFT is applied to each windowed frame to convert it to a frequency domain representation (spectral estimation). The windowed frame is in discrete time domain signal which needs Discrete Fourier Transform (DFT) to transform it to frequency domain signal for easier analysis. To perform STFT, in this study FFT applied to compute DFT. Because applying DFT on windowed frame directly takes too much time. But, FFT is a very efficient algorithm to make computer doing the DFT in a much shorter time. FFT is able to divide a compound complex signal to a simpler frequency domain signal, as shows below in Figure 4-4:

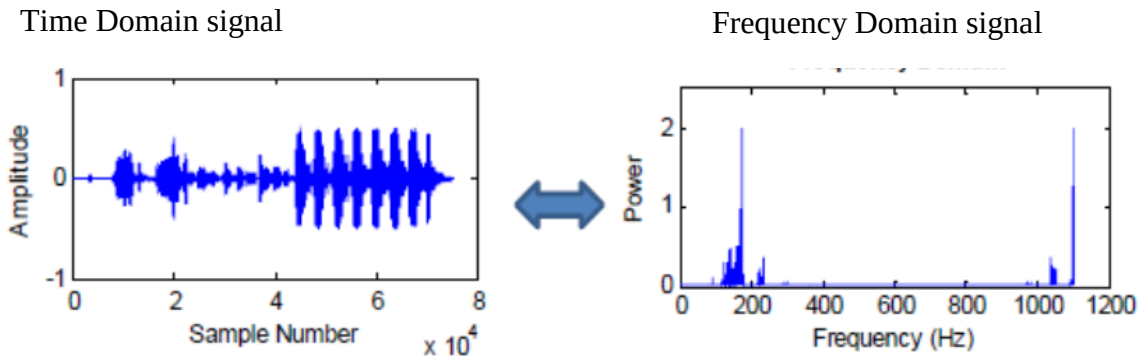

Figure 4-4: Time domain and Frequency domain signal respectively

4.2.2 Feature Extraction

Feature extraction is a crucial and highly impactful process in environmental sound recognition approach [14, 19, 21]. Because good features extracted from the audio signal precisely represents the given environmental sound; on the other hand, bad features misrepresent the given input signal or environmental sound [5]. This causes to the recognition algorithm degrades its recognition accuracy. To achieve the expected goal in the area of environmental sound recognition, selection of representative features and feature extraction techniques should take much focus [5, 19]. Selection of audio features depends on not only for achieving good recognition performance but also it depends on the device that experimental analysis takes place [11]. These devices may resource intensive or not. For instance, Android based smart phones are resource limited device. Audio features to be used in such smart phone based environmental sound recognition; should achieve a good recognition accuracy, require minimum computational complexity, low power consumption and should have a minimum memory requirement.

In this study a group of frequency and time-frequency domain features used for EASR due its representation power. And also, EAS are highly stochastic and noise additive due to this the combination of frequency and time-frequency domain features can represent useful features of the sound. Frequency domain features are MFCCs, SC, Chroma; and time-frequency domain feature is Mel.

4.2.2.1 Mel Frequency Cepstral Coefficient (MFCC)

Computation of MFCC includes: First, pre-process the signal and take DFT coefficients. Second, convert Fourier coefficients to Mel-scale filter bank to get how much energies in each filter bank, i.e. filter bank coefficients taken. Third, logarithmic the obtained Mel-scale filter bank energies, which give us a log filter bank energy. Fourth, the result of logarithmic value of Mel-scale decorrelated by discreet cosine transform (DCT) in order to remove redundant information, which give us cepstral coefficients. Finally, the discrete coefficients taken as MFCCs. The complete procedure for extracting MFCCs is as shown below.

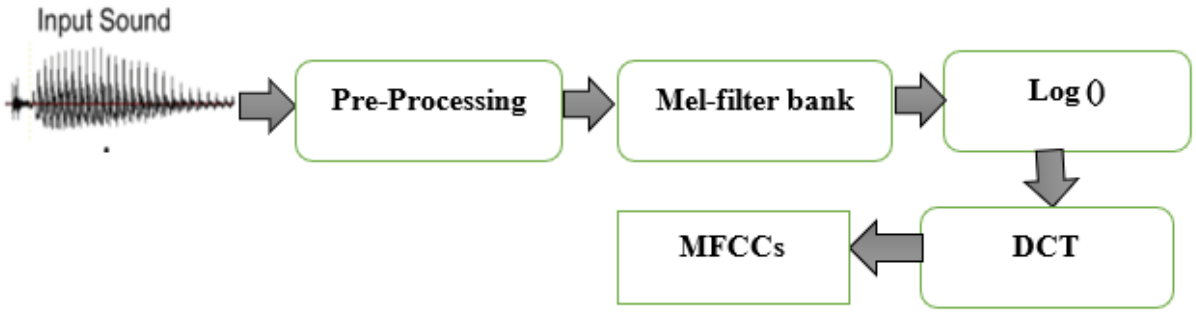

Figure 4-5: Block diagram for MFCCs extraction process [11]

In the above Figure 4-5, the first step is input sound pre-processing; which includes pre-emphasis, frame blocking, windowing audio frames and STFT i.e. compute DFT. The viewpoint of this phase is to represent audio signal in a statistical stationary manner by dividing it to a smaller section (in our case 50ms). Pre-emphasis takes place to amplify spectrum energies in high frequency band of the spectrum. Frame blocking with the aim of dividing audio signal to smaller segments. Windowing (Hamming window) to reduce the discontinuity between frames caused by sharp corners of each frame. STFT: i.e. FFT applied to compute DFT in efficient manner to obtain the magnitude frequency response of each frame.

The next step is Mel-scale filter bank which is a triangular bandpass filter used to get the log energy of each triangular bandpass filters (smooth magnitude spectrum) by multiplying the magnitude frequency response with 20 triangular bandpass filters and then add up each

filter coefficients. Conversion between frequency value in Hertz (f) and in Mel is given by:

$$\text{Mel}(f) = 2595 \log_{10}(1 + \frac{f}{700}) \qquad 4\text{-}3$$

The next step is computing the logarithm of filter bank energies to normalize a channel between human's perception of the loudness and the sound intensity. This step takes log Mel filter bank coefficients.

The final step, DCT used to get cepstral coefficients i.e. average power (energy) in the spectrum. DCT gathers most of the information of the signal to its lower order coefficients, resulting insignificant reduction in computational cost. The result of the conversion is called MFCC. This calculation is given by:

$$c_i = \sum_{k=0}^{K-1} (\log S_k) \,.\, \cos(\frac{i\pi}{K}(k - \frac{1}{2})) \qquad 4\text{-}4$$

Where c_i is the i-th MFCC, S_k is the output of k-th filter bank channel that is the weighted sum of power spectrum bins on that channel and K is the number of coefficients (number of Mel-filter banks). It is usually between 20 to 40. We set the value of K to 20.

4.2.2.2 Spectral Contrast

Spectral contrast features called Octave based spectral contrast; that represent the difference between harmonic content in the sound (peaks) and non-harmonic content i.e. noise (valleys) measured in each frequency sub band by Octave-scale filters [27]. Due to this spectral contrast features designed to provide better discrimination of audio signal than MFCCs [28]. When computing the spectral envelope (MFCCs) spectra in each sub band frequency are averaged. As a result, average spectral characteristics are obtained. But, to differentiate one audio signal to the other, each sub band frequency characteristics required [28].

The extracted feature vectors in spectral contrast passed a procedure that have existed in MFCC; that is pre-processing includes pre-emphasis, frame blocking, windowing and STFT(FFT), and its own additional procedures i.e. Octave-scale filters, selection of peaks, valley and their difference, logarithm of obtained difference value and Karhunen-Loeve

Transform (K-L). Pre-processing includes pre-emphasis, frame blocking, windowing and FFT processes are similar with MFCCs as discussed in Section 4.2.1

The next step is Octave-scale filter, after audio signal is converted to frequency domain spectrum by FFT, Octave scale filter divides each frame to six Octave based sub bands. These sub bands used to select spectral peaks, spectral valleys and to compute their difference that represent spectral contrast. Furthermore, spectral peaks represent the harmonic content of the signal that causes to have high spectral contrast and spectral valleys represent non-harmonic content of the signal that represent noises which allows to have lower spectral contrast. So spectral peaks and valleys inform the relative spectral harmonic and non-harmonic distribution of the signal in each Octave-based sub band frequency.

Then, the strength of spectral peaks and spectral valleys are estimated by the average value in the small neighbourhood around maximum and minimum value respectively, instead of the exact maximum and minimum value themselves to ensure the stability of features. The neighbourhood factor **α** is introduced to describe the small neighbourhood. The value of **α** is tested from 0.02 to 0.2. But varying the value of **α** in the range doesn't have influence in the performance. So, the value of **α** is set 0.02. Detailed expressions are as follows: Suppose the FFT vector of k-th sub-band is $\{X_{k,1}, X_{k,2}, \dots X_{k,N}\}$. This vectors sorted in descending order the new vector represented as $\{ \dot{X}_{k,1}, \dot{X}_{k,2}, \dots \dot{X}_{k,N} \}$ where $\dot{X}_{k,1} > \dot{X}_{k,2} > \dots > \dot{X}_{k,N}$. Now the strength of spectral peaks and spectral valleys computed as:

$$\text{Peak}_k = \log\{\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} \dot{X}_{k,i}\} \qquad 4\text{-}5$$

$$\text{Valley}_k = \log\{\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} \dot{X}_{k,N-i+1}\} \qquad 4\text{-}6$$

Their difference is:

$$SC_k = \text{Peak}_k - \text{Valley}_k \qquad 4\text{-}7$$

Where N is the total number in k-th sub-band, kϵ [1,6].

The raw spectral contrast features represented in 12-dimensions in the form of $\{SC_k, \text{Valley}_k\}$ (kϵ [1,6]). Spectral valleys contained in the feature important to keep more spectral information.

Finally, K-L applied on spectral contrast feature vectors to reduce redundant information in the similar approach as MFCCs Discrete Cosine Transform (DCT) process [28]. Then the feature vectors mapped to orthogonal space by estimating the covariance matrix of each class, and the covariance matrix becomes diagonal in the new feature vector. The matrix that generate the orthogonal base vector is as:

$$S_w = \sum_{i=1}^{5} P_i \Sigma_i \qquad 4\text{-}8$$

Where S_w is the generated matrix; P_i and Σ_i are the prior probabilities and the covariance matrix of i-th class respectively. The value of P_i is set to 0.2 and Σ_i estimated from the training set of i-th environmental alarm sound type. The orthogonal base vectors are the eigenvectors of the generated matrix S_w. At the end, the transformation has done:

$$\dot{x} = Ux \qquad 4\text{-}9$$

$$U = [u_1, u_2, \dots u_j \dots u_D]^T \qquad 4\text{-}10$$

Where x is the raw spectral contrast feature, $\dot{x}$ is the spectral contrast feature vector after K-L transform, u_j is the j-th orthogonal base vector and D is the dimension of feature space. The flow diagram of the spectral contrast feature extraction processes as shown below.

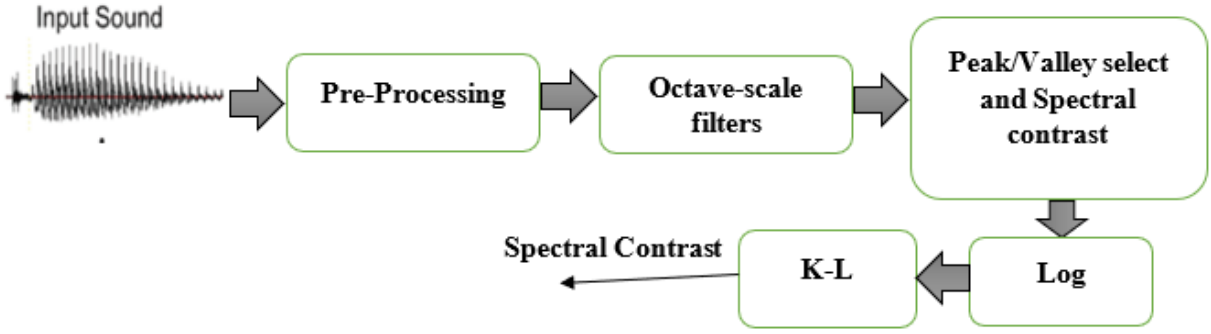

Figure 4-6 Block diagram for Spectral Contrast feature extraction

4.2.2.3 Mel Spectrogram

Mel spectrogram features represent time-frequency feature distribution of the EAS, rather than frame-based features that extract slice of frequency information at a given time. A spectrogram is visual representation of sound. It displays the magnitude of frequency component changes of the signal over time. It visualizes a sound in the form of two-dimensional image (frequency vs time), suitable for sound recognition [32]. In Mel-spectrogram, the frequency scale changed to Mel-scale by over-lapping triangular filters, as the human ear perception of sound, which is being able to discriminate at lower frequency and less discriminate at higher frequency (i.e. comparison of non-linear pitch). The computation of Mel spectrogram features takes place after pre-processing (pre-emphasis, frame blocking, windowing) the input signal. Pre-processing of the signal is similar to the pre-processing of the signal in extracting MFCCs. Detail explanation of pre-processing of the input signal is mentioned in Section 4.2.2.1.

Then we compute the decibel scale spectrogram image of the windowed frame using methods explained in [33]:

$$S(f, t) = \sum_{m=-\infty}^{\infty}(Y_t(m))e^{-j\frac{2\pi}{N}fm} \qquad 4\text{-}11$$

Where $Y_t(m)$ is the windowed m[th] frame at time t, which is $Y_t(m) = X(m)w(t-m)$ as defined in Section 4.2.1.3 . w(t) is the Hamming window; (f, t) is the representation of the pixel coordinates of the spectrogram (i.e.) time-frequency indexes with N being the size of FFT and f=0, …, N-1 is the frequency sub-band.

The Mel spectrogram is represented by a matrix $S(f, t)$, where f is the frequency and t denote the time. Then, the Mel spectrogram feature extraction from the spectrogram image algorithm explores the matrix using the following strategy:

First, t_n and f_n such that $|S(f_n, t_n)| \geq |S(f, t)| \forall$ (f, t), placing n[th] segment in t_n . The amplitude of this point is calculated as:

$$Y_n(0) = 20\log_{10}(|S(f_n, t_n)|) \qquad 4\text{-}12$$

Second, if $Y_n(0) < Y_0(0) - \beta dB$, the segmentation process is stopped, as the signal amplitude is inferior to the stopping criteria β. For EAS, β has been set to 25 dB. Third, from t_n seek the highest peak of $|S(f,t)|$ for t>t_n and t<t_n , until $Y_n(t)$<$Y_n(0) - \beta dB$ for both sides and the starting and ending times of nth segment are denoted as, $t_n - t_s$ and $t_n + t_e$.

Fourth, save the amplitude trajectories as the nth segment. Fifth, delete the nth segment from the matrix $S(f, t_n - t_s, \dots, t_n + t_e) = 0$ and set n=n+1. Sixth, repeat from step one until the end of the spectrogram. Figure 4-7 illustrates the processes of Mel feature extraction.

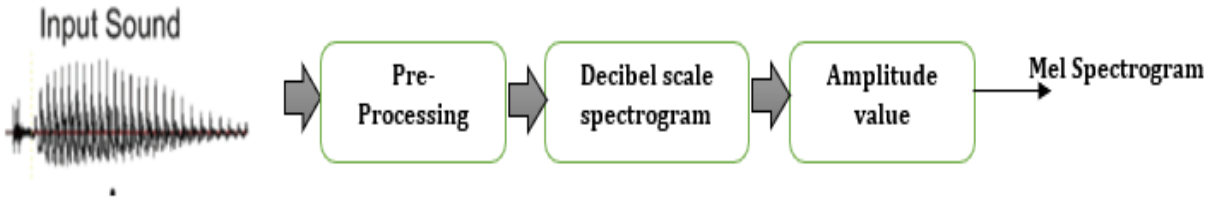

Figure 4-7: Block diagram for Mel spectrogram feature extraction

4.2.2.4 Chroma Gram

Chroma-based features which is also called harmonic pitch class profiles that represent short-time energy distribution among the each pitch classes; suitable for sound processing [29, 30, 31]. A chroma vector can be represented as a 12-dimensional vector x = (x (1), x (2), ..., x (12))^T, where x (1) corresponds to chroma C, x(2) to chroma C#, and so on [31]. The computation of Chroma gram feature vector steps is based on the method explained as in [29]:

First, a chroma spectrum X(n) defined, that measure signal strength with respect to the given value chroma: c. according to Shepard's helix of pitch perception, chroma c is the fractional part of base2 logarithm of the frequency showed as follows:

$$c = \log_2 f - \lfloor \log_2 f \rfloor \qquad 4\text{-}13$$

Where $\lfloor . \rfloor$ function that rounded the greatest integer value. Chroma spectrum is analogues to the standard Fourier power spectrum [29]. If we conduct frame segmentation before it, we can create time-frequency distribution X(t,f), like this we can also define time-chroma

distribution of X (t, c). This distribution is called Chroma gram which is remapping of traditional time-frequency distribution through aggregation function F, generated by:

$$X(t, c) = F(X(t, f)) \qquad 4\text{-}14$$

where:

$$f = 2^{c+h} \qquad 4\text{-}15$$

where the value of c obtained from Equation 4-13, h denotes tone height as Shepard's helix of pitch perception stated and aggregation function, we used to be summation. The elements the chroma feature vector for the t^{th} frame v(t,k) can be calculated using the equation:

$$v(t, k) = \sum_{n \epsilon S_k} \frac{X_t(n)}{N_k} \text{ , } k\epsilon\{0,1, \ldots, 11\} \qquad 4\text{-}16$$

Where $X_t(n)$ is the logarithmic magnitude of the Discrete Fourier Transform (DFT) for the t^{th} frame, S_k is the subset of the discrete frequency space for each pitch class, $N_k m$ defines the number of elements in S_k.

Second, normalization of each feature vector done by taking the arithmetic mean of all log magnitude of DFT bins within a given set S_k and subtracting the scalar mean of that vector's 12 features. The normalized chroma-based features illustrate that short-time energy distribution among the twelve chroma and closely corelate to the harmonic progression of the underlying piece [31]. The 12 sets S_k generated by associate each DFT bin with one of the 12 pitch classes. We calculate a DFT bin's associated frequency, and then calculate the chroma value based on Equation 4-13. The bin is associated with the pitch class with the nearest chroma value. The chroma value of each pitch class reassigned is done let's say pitch class C is centred at chroma value 0 and pitch class B is centred at chroma value of 1; then the other pitch classes centred at k/12.

Finally, the included spectrum is restricted as in [30], we choose the lower bound is 20Hz and upper bound is at 20000Hz. Figure 4-8 shows the process involved in chroma gram feature extraction of the audio signal.

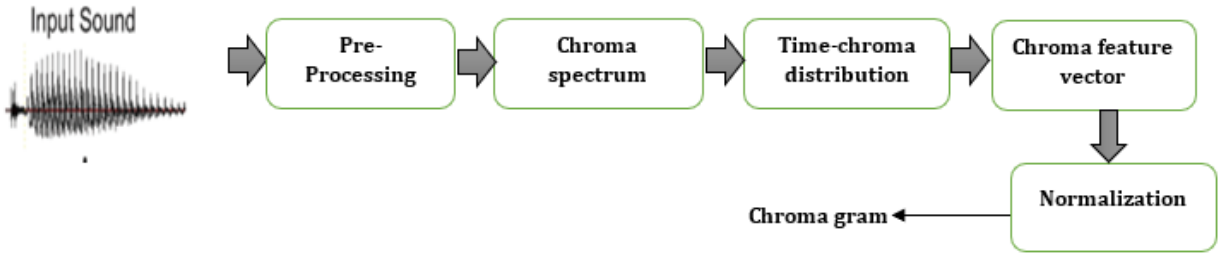

Figure 4-8: Block diagram for Chroma gram feature extraction

4.3 Construction of The Recognition Model

4.3.1 Multi-Layer Perceptron (MLP) Classifier

The multilayer perceptron is the most known and most frequently used type of a neural network [34] that can able to solve linearly non-separable problems. It is a neural network connecting multiple layers in directed graph, which means that the signal path through the nodes (neurons) only goes one way from input to output. In this neural network architecture, all the layers ordered sequentially and each layer receive inputs from the previous one. For an input vector x, an MLP computes the hidden activation vectors h and the output ŷ as:

$$h = \mathcal{F}(W^{xh}x + b^{h}) \qquad 4\text{-}17$$

$$\hat{y} = \mathcal{G}(W^{h\hat{y}} + b^{\hat{y}}) \qquad 4\text{-}18$$

where W** denotes the weight matrices connecting two layers, i.e. W^{xh} are the weights from input to hidden layer and W^{hy} from hidden to output layer, b* are the bias vectors, and $\mathcal{F}$ and $\mathcal{G}$ are activation functions. When an activation function is computed for all the neurons in a layer using vector notation, such as Equation 4-17 and 4-18, it is always computed element-wise. In case of multiple hidden layers, the input to hidden layer h^{l} is the output of the previous hidden layer h^{l-1}.

From an input x a prediction ŷ is computed at the output layer, and compared to the original target y using a *cost function* $E(W, b; x, y)$. The network is trained to minimize $E(W, b; x, y)$ for all the input samples x in the training set, as:

$$E(W, b) = \frac{1}{N}\sum_{n=1}^{N} E(W, b; x, y) \qquad 4\text{-}19$$

Where N is the number of training samples. Since $E(W, b)$ always depends on W and b, we will refer to it as E for simplicity.

Generally, *cross entropy* (CE) is used as a cost function to train NNs for classification task, with the following equation:

$$CE = -\frac{1}{N}\sum_{n=1}^{N} y_n \, log \, \hat{y}_n + (1 - y_n \, log \, 1 - \hat{y}_n) \qquad 4\text{-}20$$

Where $\hat{y}_n$ and y_n are respectively the prediction and the target output of an instance x_n.

Optimizer function we used is the sum of squared errors (SSE) and its variation root mean square error (RMSE):

$$SSE = \sum_{n=1}^{N}(y_n - \hat{y}_n)^2 \qquad RMSE = \sqrt{\frac{\sum_{n=1}^{N}(y_n-\hat{y}_n)^2}{N}} \qquad 4\text{-}21$$

Since NNs are constructed as differentiable operatories, they can be trained to minimize the differentiable cost function using *gradient descent*-based methods. The intuitive idea is to find the gradient on the error surface for a given weight configuration, and iteratively adjust the weights by moving in the direction of the negative slop. The weights W and b are generally initialized with random values and then iteratively adjusted during training. An efficient algorithm we used to compute the gradients for all the weights in the network is backpropagation algorithm [22], an implementation of the chain rule for practical derivatives along the network.

To illustrate the backpropagation algorithm on a MLP we will the following notation: $\mathcal{F}$ is an activation function and $\mathcal{F}'$its derivatives; w_{ji}^l is the weight connecting the i^{th} neuron in layer $l-1$ to the j^{th} neuron in layer l; z_j^l is the weighted input to the j^{th} neuron in layer n, shown as:

$$z_j^l = \sum_i w_{ji}^l h_i^{l-1} + b_j^l \qquad 4\text{-}22$$

h_j^l is the activation of the j^{th} neuron in layer l, which means $h_j^l = \mathcal{F}(z_j^l)$ the boldface version of the aforementioned quantities represents the concatenation of the all respective values in one layer.

Gradient descent can be used to minimize the cost function E, since the network is constructed of differentiable elements. The algorithm requires to compute the derivative of the cost function with respect to each of the weights and bias terms in the network. Once the gradients $\frac{\partial E}{\partial w_{ji}^l}$ and $\frac{\partial E}{\partial b_{ji}^l}$ have been computed, the corresponding weights and biases in the network can be updated by taking a small step towards the negative direction of the gradients, for instance using stochastic gradient descent as:

$$\Delta w_i(\tau+1) = -\eta \frac{\partial E}{\partial w_i} \qquad \textit{4-23}$$

$$w_i(\tau+1) = w_i(\tau) + \partial w_i(\tau+1) \qquad \text{4-24}$$

Where $\Delta w_i(\tau+1)$ is the weight update, η is the learning rate that control the amount of update, and τ is the index of the training iterations. The same update rule applies to the bias terms, with *b* in place of *w*.

Backpropagation is a technique that allows to efficiently compute the gradients for all the parameters of the network, by propagating backwards the errors at the output layer. First, the propagated error δ_j^L is computed for each neuron in the output layer, the L^{th} layer.

$$\delta_j^L = \frac{\partial E}{\partial z_j^L} = \frac{\partial E}{\partial h_j^L}\frac{\partial h_j^L}{\partial z_j^L} = \frac{\partial E}{\partial h_j^L}\mathcal{F}'(z_j^L) \qquad \textit{4-25}$$

Where $\frac{\partial E}{\partial h_j^L}$ depends on the cost function E. We can then compute the backpropagated errors δ_j^l at the l^{th} layer in terms of the propagated error δ_j^{l+1} in the next layer as shown:

$$\delta_j^l = \frac{\partial E}{\partial z_j^l} = \sum_i \frac{\partial E}{\partial z_j^{l+1}}\frac{\partial z_i^{l+1}}{\partial z_j^l} = \sum_i w_{ij}^{l+1}\,\delta_i^{l+1}\mathcal{F}'(z_j^l) \qquad \textit{4-26}$$

Where we used the equivalence

$$\frac{\partial z_j^{l+1}}{\partial z_j^l} = \frac{\partial}{\partial z_j^l}\sum_i w_{ij}^{l+1}\mathcal{F}(z_j^l) + b_i^{l+1} = \sum_i w_{ij}^{l+1}\mathcal{F}'(z_j^l) \qquad \textit{4-27}$$

and by definition

$$\delta_i^{l+1} = \frac{\partial E}{\partial z_i^{l+1}} \qquad \textit{4-28}$$

Finally. We can express the gradients $\frac{\partial E}{\partial w_{ji}^l}$ and $\frac{\partial E}{\partial b_{ji}^l}$ in terms of the error δ_j^l

$$\frac{\partial E}{\partial w_{ji}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ji}^l} = h_i^{l-1}\delta_j^l \qquad \text{4-29}$$

Where we used again the definition of δ_j^l and

$$\frac{\partial z_j^l}{\partial w_{ji}^l} = \frac{\partial}{\partial w_{ji}^l} \sum_i w_{ji}^l h_i^{l-1} + b_j^l = h_i^{l-1} \qquad 4\text{-}30$$

For the gradients with respect to the bias terms, using the same procedure as Equation 4-29 $\frac{\partial E}{\partial w_{ji}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ji}^l} = h_i^{l-1}\delta_j^l$ 4-29 we obtain:

$$\frac{\partial E}{\partial b_j^l} = \delta_j^l \qquad 4\text{-}31$$

Where the only difference is the h_i^{l-1} term that disappears when computing

$$\frac{\partial z_j^l}{\partial b_j^l} = \frac{\partial}{\partial b_j^l} \sum_i w_{ji}^l h_i^{l-1} + b_j^l = 1 \qquad 4\text{-}32$$

The Stochastic gradient descent might lead to slow convergence. There are several techniques such as momentum, RMSprop and others can greatly improve convergence speed. Here we used RMSprop to improve the convergence speed i.e. optimizer algorithm.

RMSprop: RMSprop is an optimizer with a pre-parameter adaptive learning rate. It uses a moving average of the magnitude of recent gradients to normalize the current gradients. The normalization is performed over the root mean squared gradients; this is the way its name is RMSprop. The learning rate η is typically called step rate, and the running average term $r(\tau)$ is added to the weight update equation:

$$r(\tau) = \gamma r(\tau - 1) + (1 - \gamma)\left(\frac{\partial E}{\partial w_i}\right)^2 \qquad 4\text{-}33$$

$$\Delta w_i(\tau + 1) = \frac{-\eta}{\sqrt{r(\tau)}} . \frac{\partial E}{\partial w_i} \qquad 4\text{-}34$$

Where γ is the decay term, typically set to 0.9 that controls the contribution of new gradients to the running average $r(\tau)$.

4.3.2 Convolutional Neural Network (CNN)

CNNs are extended version of neural networks (NNs). They have proven effective at extracting classification features directly from the data with different types of signals such as music, speech, image etc [36]. A CNN's architecture, which is the set of parameters that we need to design a network, which is based on stacking many hidden layers on top of one another that makes CNNs capable of learning hierarchal features. In this way, they can learn higher-level features from a set of previously learned lower-level features.

The architecture of CNN composed of first convolutional layer that operate as feature extractor which consist of a set of convolutional filters with a fixed size. These filters convolve in parallel with all regions of an input data with an overlapping distance called a stride. The output of each convolutional filter in a convolutional layer is a new learned representation of data named feature map. Every entry in a feature map is equivalent to one neuron in a NN's layer. Subsequently, the following hidden convolutional layers similarly extract features from the input feature maps previously learned by a former convolutional layer. The output of these hierarchical feature extractors is fed to a fully-connected neural network that performs a classification task.

The convolution within CNN architecture analytical expression is given as:

$$h_j^{(n)} = \sum_{=1}^{K} h_k^{(n-1)} * w_{kj}^{(n)} + b_j^{(n)} \qquad 4\text{-}35$$

Where * denotes a 2-D convolution operation. $h_j^{(n)}$ is the j^{th} feature map output in the n^{th} hidden layer, $h_k^{(n-1)}$ is the k^{th} channel in the j^{th} filter in n^{th} layer and $\mathrm{b}_\mathrm{j}^{(\mathrm{n})}$ is the corresponding bias term. In the CNN learning filters coefficients and fully connected weights initially set random values., then learned and adjusted through an algorithm known as backpropagation. To compute and solve linearly non-separable function as such in our study recognition of EAS, the convolutional and fully-connected layers of CNNs followed by activation function. This enables each artificial neuron to act as a human neuron that suppresses the small values of a layer's output and activate the large ones. Moreover, the set of hierarchical convolutional layers yields a large volume of feature maps, which cause to the CNN computationally very expensive. To handle this problem, the convolutional

layer followed by other type of layer naming pooling which helps to reduce the dimensionality of feature map. In addition to this it reduces the computational cost associated with training the model and tackle the chance of over-fitting the most representative features problem throughout the CNN. There are different types of pooling operations such as max, average and stochastic pooling. In our work max and average pooling operation applied.

The coefficients of the convolutional filters w_{ij}'s is automatically learned using an iterative algorithm which alternates between feedforward and backpropagation passes of the data. This iterative process aims to minimize the average loss between the actual labels and the network output, i.e.

$$E = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c} y_i^{*(k)}\, log\, y_i^{(k)} \qquad 4\text{-}36$$

Where $y_i^{*(k)}$ and $y_i^{(k)}$ are respectively the true label and the network output of the i^{th} feature vector at the k^{th} class with m training EAS and c neurons in the output layer. To solve the underlying optimization problem numerous solvers could be used. In this work we used the stochastic gradient descent (SGD) to train the CNN model.

The iterative update rule for the filter's coefficients $w_{ij}^{(n)}$ in CNN during the back-propagation pass is given bellow:

$$\nabla w_{ij}^{(n+1)} = m.\nabla w_{ij}^{(n)} - d.\varepsilon.w_{ij}^{(n)} - \varepsilon.\frac{\partial E}{\partial w_{ij}^{(n)}} \qquad 4\text{-}37$$

$$w_{ij}^{(n+1)} = w_{ij}^{(n)} + \nabla w_{ij}^{(n+1)} \qquad 4\text{-}38$$

Where $w_{ij}^{(n)}$represents the i^{th}channel in the previous feature maps denoted by $h_i^{(n-1)}$, $\nabla w_{ij}^{(n)}$ denotes the gradient of $w_{ij}^{(n)}$and ε is the learning rate. The letters m and d are respectively the momentum and the decay. The bias term $b_j^{(n)}$ in Equation 4-35 is updated using the same equation presented Equation 4-36 and Equation 4-37.

4.4 Android-Based EAS Recognition Model

The EASR takes place in the Android based smartphone with the help of Frozen Graph generated from the offline model training and testing (see Chapter 5, subsection 5.4). The Frozen Graph consists of the trained feature and the corresponding class label, and finally weights and bias variables value adjusted during offline training and testing process. The Android-based audio recognizer used this frozen graph to match the live recorded audio feature and corresponding class label (for classification)

The Android-based EASR model uses the Frozen Graph to match and detect new (unknown) audio events recorded in Android-based from the microphone of the smartphone. The audio recognition involves recording unknown audio signal, pre-process, extracting audio features for the unknown audio signals, matching the extracted audio features, with the audio features involve in the Frozen Graph and finally displaying the recognition result (corresponding class label) to the user (see the detail in Chapter 6).

CHAPTER FIVE

5. Model Offline training and testing and Comparison

5.1 Introduction

This chapter provides the analysis and python offline training and testing result of the performance of top two generic sound classifier and various audio feature extractor (mentioned in Chapter 4). The first section mentions about the datasets used for offline training and testing of the classifier algorithms. In the subsequent section, the performance of each of the two prevalent classifiers, MLP and CNN, is evaluated and compared (in terms of recognition accuracy and computational speed), based upon which the best classification algorithm selected.

5.2 Offline Training and Testing Setup

We prepared our own dataset by collecting some of data samples from the web and by recording EAS for experimental analysis of this study.

5.2.1 Datasets

The sound recording was conducted in the city of Bahir Dar during normal working hour with some possibility of background noises. Unlike in other previous works which uses high quality standalone microphones, instead using the smartphone's (TECNO Camon CX) internal microphone for recording the sounds with sampling rate of 44.1kHz, mono channel, 16-bit resolution. Because, after offline training and testing of EASR model, the outperformed classifier algorithm is tested on Android based smartphone's microphone captured EAS (i.e. the model finally deployed on this Android smartphones). So that, the data samples should be recorded using smartphone's microphone rather than high quality standalone audio recorder. There is a total of 1,732 data samples (recordings) in the dataset and each of them are 2 to 5 seconds long. The dataset contains five types EAS: Car horn, Dog bark, Engine idling, Gunshot and Fire alarm.

All the data sample files in the dataset corresponding to each of the 5 sound classes are collected as uncompressed wave files, each record from 2 to 5 second duration. From the dataset some of data samples collected from different datasets (from ESC-10 [43] (40 clips

of dog barking),and UrbanSound8K [44] (150 clips of car horn, 350 clips of engine idling, 200 clips of dog barking and 150 clips of gunshot)) totally 890 audio clips download from internet. And the remaining 842 data samples (200 clips of car horn, 62 clips of dog bark, 36 clips of engine idling, 332 clips of fire alarm and 212 clips of gunshot) recorded in the city of Bahir Dar. Each audio clip lasts from 2s to 5s, the analysis done by 25% overlap of windowing. The recorded sound re-prepared to have same format as the audio data that collected from the web by loading and edit the format using *Format Factory* [12] application. This re-preparation enables the recorded audio compatible with the collected data from the web to reduce bias when developing a compressed dataset for the offline training and testing of this work. The dataset of EAS for this work is provided in Table 5-1.

Table 5-1: EAS dataset

EAS class	No of Samples (Segments)	Description
Car horn	350	Sound produced by the car to warn others of the car approach or presence, or to call attention to some hazards.
Dog barking	302	Sound produced by a dog at any noise or object that catches their attention or startles them, or when they are frightened, lonely, surprised or irritated, etc.
Engine idling	386	Sound produced by vehicle to show running a vehicle's engine when the vehicle is hot in motion. When drivers are stopped at a red light, waiting while parked outside a business or residence.
Gunshot	362	Sound that is a single discharge of a gun, typically a man-portable firearm, producing a visible flash.
Fire alarm	332	Sound produced by electronic sounder which uses visual and audio signalization to warm people about a possible fire, smoke, or carbon mono oxide occurrence in the area of coverage.
Total	1,732	

5.2.2 Dataset Partitioning

The correctness of the classification result determined during the validation stage, which is based on effective dataset partitioning. The choice of the training and testing dataset is the most significant factor in the validation stage. Our dataset consists of different combination of feature vectors and corresponding class labels. Hence, the main issue to be addressed here is how our dataset should be partitioned to training and testing set. Once this decision is made, the next stage is to measure the correctness of the design: train the classifier by applying training set and then test the classifier by the unknown test set to know how much accurate the classifier is using one of the performance measures.

For the validation of results and to decide how to split the dataset, several cross-validation techniques have been proposed in the literature [45] such as hold-out (HO), repeated sub-sampling (RHO), leave-one-out and others. In our offline training and testing, in the case of MLP classifier we compare the dataset partitioning of 80/20 and 70/30. Hence, high performance result obtained using hold-out validation with partitioning EAS dataset into 80% dataset for training and 20% of the dataset for testing, achieve high result than 70/30 dataset partitioning. For CNN classifier, in addition to training and testing the dataset partitioned to validation purpose. Due, to this the dataset partitioned to 70% for training, 10% for validation and 20 for testing.

Each of the classifiers recognition performance result presented in confusion matrix. A confusion matrix is a popular evaluation method used to summarize for classification algorithms [46]. We create a confusion matrix CM of size $N \times N$, where N is the number of classes in the dataset (EAS dataset), and rows and columns of the confusion matrix describes the ground truth (true label) and output (predicted label of the classifier) respectively. More details of the confusion matrix and recognition performance of the classifiers discussed in the next section.

Time built-in Python module is used to measure the computational complexity (running speed) of the classifiers. For offline training and testing and testing the performance of the algorithms, we used a machine (laptop) with Intel Core i3 CPU and processor speed of 2.40 GHz.

5.3 Performance Evaluation Results

5.3.1 Performance of MLP Classifier

Configuration

MLP parameters are selected among many numbers of controlled experiments in this thesis. Among these experiments, the MLP network setup with the highest overall accuracy is constructed by using the following parameters:

Table 5-2:Network Parameter configuration for MLP with the highest accuracy

Parameter	Value
Features	MFCC, Mel, Chroma
Number of input neurons	153
Number of hidden layers	1
Number of neurons in hidden layer	576
Number of output neurons	5
Activation function for hidden layer neurons	ReLu
Activation function for output layer neurons	SoftMax
Learning rate	0.01
Initial weight and bias range	(-0.001,0.001)

Offline training and testing of MLP classifier are done by using the configuration in Table 5-2 as base point. The effect of each parameter on the performance of the MLP recognition capability is spanned while keeping all the others fixed. Thus, the aim of observing the effect of individual parameters is to select the optimal parameter values.

5.3.1.1Effects of MLP Parameters

Effect of Features In this thesis, both spectral and spectro-temporal features are used to represent the unstructured EAS in the case of MLP. Initially the combination of all features (MFCC, Chroma, Mel and SC) is used for training and testing MLP recognition model.

Table 5-3: MLP Confusion Matrix (for recognizing EAS dataset) using MFCC, Chroma, Mel and SC

		Predicted Labels				
		Carhorn	DogB	EnginI	FireA	Gunshot
True Labels	Carhorn	54	1	1	3	0
	DogB	0	57	0	11	1
	EnginI	0	0	78	0	0
	FireA	0	0	0	66	0
	Gunshot	3	3	0	1	68

Table 5-4: MLP performance (for EAS dataset) using MFCC, Chroma, Mel and SC Features per class

	Precision	**Recall**	**F: score**	**Accuracy**
Carhorn	**94**	**92**	**93**	**87**
DogB	**93**	**83**	**88**	**78**
EnginI	**98**	**100**	**99**	**99**
FireA	**81**	**100**	**89**	**81**
Gunshot	**98**	**91**	**94**	**89**
Avg.	**93**	**93**	**93**	**87**

Overall accuracy: 87, Average Precision: 93, Average Recall: 93, Average F-score: 93

As shown in Table 5-4, MLP scored different performance for different class label, due to that each class in dataset has different datasamples. In addtion, the feature exatraction techniques representation capablity in this case is lower as well as the quality of datasamples itself affects the recognition performance of MLP.

The feature vectors consist a combination of spectral features (MFCC, Chroma and SC) as well as spectro-temporal feature (Mel) so that its dimension is very high value. Thus, it

causes to the recognition model to require more computational power and time as well as reduces recognition accuracy.

Therefore, to select effective, robust and optimal representative features, the recognition accuracy of each audio feature type is conducted as shown in Figure 5-1. To come up with a good recognition accuracy and to reduce its computational time, we select features that score high recognition accuracy. Hence, MFCC, Mel and Croma have got 89.7%, 86.1% and 71.5% recognition accuracy respectively are selected. MFCC has 39 coefficients but except the first 13 coefficients, others represent redundant information that leads to miss classification and increases computational power requirement for the classifier. Using 13 coefficients of MFCC gives recognition accuracy 89.7%. Mel using 128 coefficients gives recognition result 86.1% and Croma using 12 coefficients gives result 71.5%.

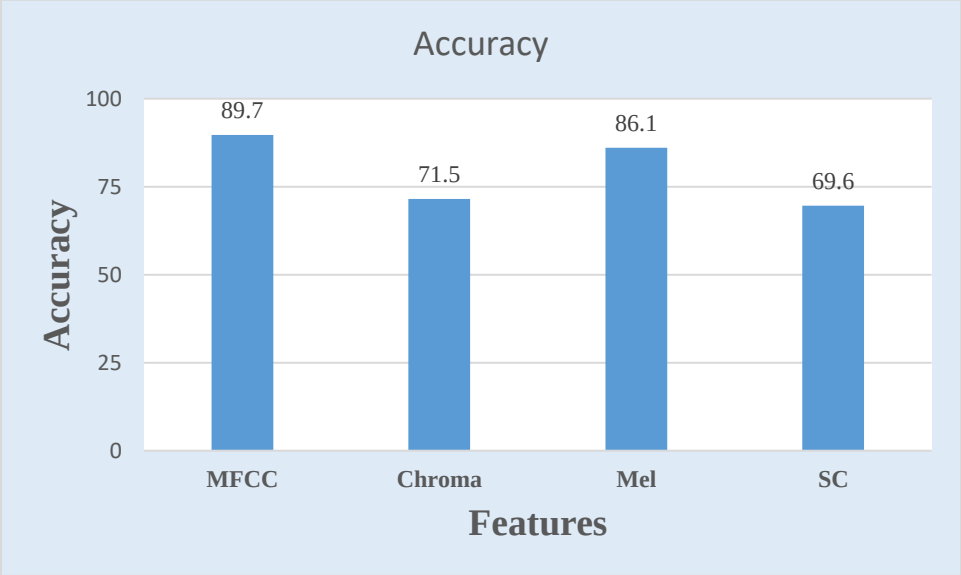

Figure 5-1: MLP Recognition on each Features (MFCC, Chroma, Mel, SC)

Finally, by combining MFCC, Mel, Chroma with 153 coefficients gives a result of 92.1 overall recognition accuracy as shown in Table 5-6 which MLP scored improved recognition accuracy than using a combination of all feature extraction techniques as well using individual extraction techniques .

MFCC and Chroma are spectral features that reduce recognition accuracy when in signals that possessed a flat spectrum. EAS such as gunshots which have a strong temporal domain

signatures, these sounds have a broad flat spectrum which are sometimes does not modelled effectively by MFCCs and Chroma. In this work, Mel proposed in addition to MFCC and Chroma to analyse EAS. Mel offer a way to extract time-frequency domain features as discussed in (Chapter 4 section 4.2.2.3) that can classify sounds in effective manner. The advantage of the time-frequency representation is the ability to bring out the useful structure of each type of sound. They provide an excellent simultaneous spatial and frequency localization of information.

Table 5-5: MLP Confusion Matrix (for recognizing EAS dataset) using MFCC, Mel and Chroma

		Predicted Labels				
	Class	**Carhorn**	**DogB**	**EnginI**	**FireA**	**Gunshot**
True Labels	**Carhorn**	**54**	0	1	1	3
	DogB	0	**68**	0	0	1
	EnginI	0	0	**78**	0	0
	FireA	0	0	0	**66**	0
	Gunshot	1	5	0	0	**69**

Table 5-6:MLP performance (for recognition EAS dataset) using MFCC, Chroma and Mel features per class

	Precision	**Recall**	**F: score**	**Accuracy**
Carhorn	**98.1**	**91.5**	**94.7**	**90**
DogB	**93.1**	**98.5**	**95.7**	**91.8**
EnginI	**98.7**	**100**	**99.3**	**98.7**
FireA	**95.5**	**100**	**99.2**	**98.5**
Gunshot	**94.5**	**92**	**93.2**	**87.3**
Avg.	**96.6**	**96.4**	**96.4**	**93.3**

Overall accuracy: 93.3, Average Precision: 96.6, Average Recall: 96.4, Average F-score: 96.4

Table 5-5 presents the confusion matrix of recognizing EAS dataset results obtained with feature combinations (MFCC, Chroma and Mel). We remark that none of the individual

features are able to attain very high performance. In this case, the use of feature combination is a solution as shown in Table 5-6 in recognition of EAS dataset.

Therefore, for recognition of EAS using MLP, a combination of audio features provides excellent recognition accuracy. Hence, MFCC, Mel and Chroma feature vectors are the optimal feature sets, which enable MLP to perform well with EAS dataset as shown in Figure 5-2.

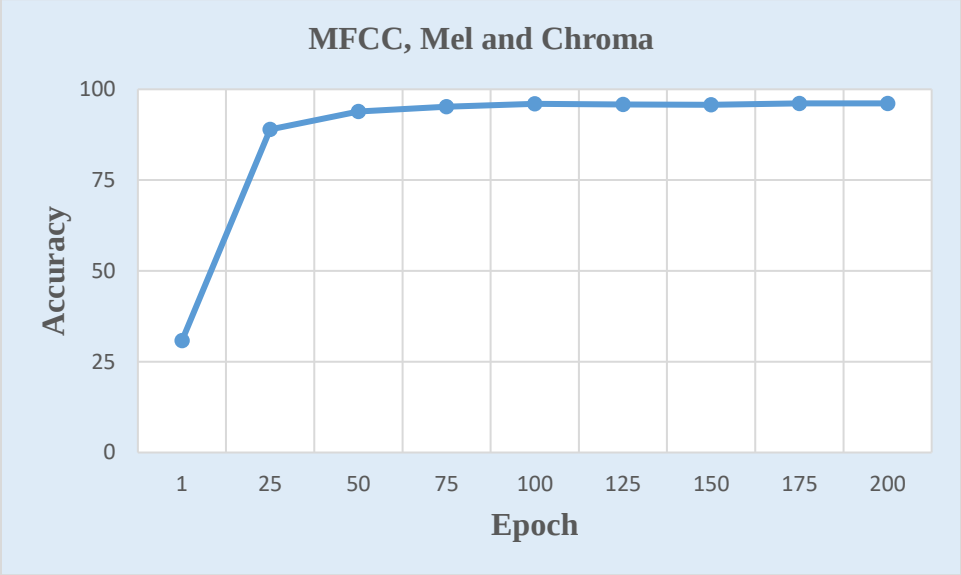

Figure 5-2: MLP performance accuracy vs Epoch using MFCC, Mel and Chroma for recognition of EAS dataset

Table 5-7 shows the overall audio features effect on the recognition of EAS using MLP that provides high recognition accuracy when using a combination of frequency features and time-frequency domain features; and lower recognition performance with using individual audio features.

Table 5-7:MLP recognition performance comparison across different audio features per class

Class	MFCC%	Chroma%	Mel%	SC%	MFCC+Chroma+Mel+SC%	MFCC+Mel+Chroma%
Car horn	83.6	64.2	83.8	44.1	87.1	90
Dog bark	84.8	58.1	81.9	72.9	78.1	91.8
Engine Idling	98.7	96.2	92.3	93.5	98.7	98.7
Fire alarm	100	67	91.3	76.2	81.5	98.5
Gun shot	81.5	72.1	81.5	64	89.5	87.3

As shown Figure 5-3 in recognition performance extremely increases from individual features to combined feature (MFCC, Mel and Chroma). So, there is an improvement on recognition accuracy of MLP from using all feature extraction techniques (87%) as well as individual feature vectors to using MFCC, Mel and Chroma (93.3%).

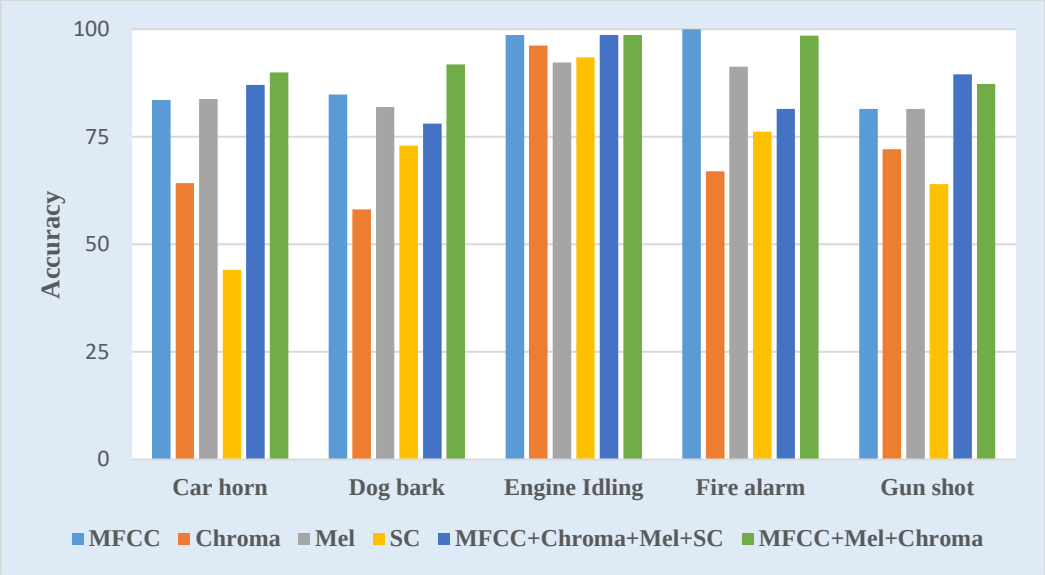

Figure 5-3:MLP recognition performance comparison across different audio features per class

Effect of Number of the Hidden Neurons The effect of number of hidden neurons on the recognition performance of the model investigated in Figure 5-4. In this thesis, one hidden layer is used for MLP, because of backpropagation rule learning mechanism that the computational complexity increases as the number of hidden layer increases [47]. Recent researchers suggested that it is enough to use one hidden layer with optimal number of neurons for MLP neural network due to its neurons connected each other and back

propagation rule. Hence, we are used one hidden layer MLP network with 576 number of neurons are used which is the outperformed value of number of neurons during offline training and testing as a function of number of hidden neurons shown in Figure 5-4.

The number of hidden neurons is highly determining the network discrimination power. Thus, it is essential to have optimal number of hidden neurons to represent the nonlinearity of the system. When the number of hidden neurons increases, the number of weights and biases increases too and huge computational power requirements are needed. On the other hand, increasing the number of hidden neurons, the weights and biases might adapt the training data. This causes to overfitting, especially in case of the number of training data is insufficient. To address such like issues from 64 to 640 number of hidden layer neurons is tried and evaluated, 576 neurons outperforms as is shown in Figure 5-4 for recognizing EAS datasets. Because these number of hidden neurons effectively represent the nonlinearity of MLP classifier and can able to generalized the training data.

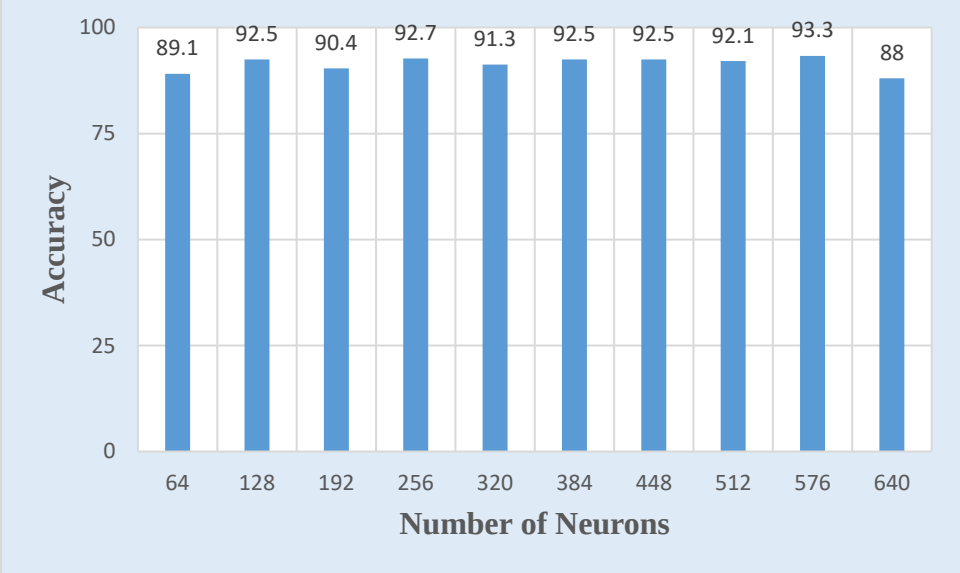

Figure 5-4:The Detection accuracy of MLP network as a function of number of hidden neurons

Effect of Number the Learning Rate The learning rate is a hyper parameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing learning rate is the vital task; as using too low learning rate causes to slow convergence. Whereas, using too high learning rate may result in leaning a sub-optimal set of weights too fast or an unstable training process. As seen in Table 5-8,

once the optimal learning rate is chosen, the network convergence to the best accuracy. The experiment is done from 0.001 to 1.0 learning rate values for EAS datasets, however values in the tables below is taken which scores better recognition accuracy than others that are not shown in the tables.

Table 5-8: MLP recognition accuracy as a function of learning rate

Learning Rate	**0.001**	**0.005**	**0.01**	**0.03**	**0.1**	**0.5**	**1.0**
Accuracy	92.6	92.6	93.3	91.2	92.2	92.6	93.1

In addition to MLP parameters, the data samples quality and number of data samples affects the recognition accuracy of MLP classifier. As shown in each MLP parameter effect investigation MLP performance differs from one class to the other class.

5.3.2 Performance of CNN Classifier

Configuration

CNN parameters are selected in the same fashion as MLP parameters, among many numbers of controlled experiments. Among these experiments, the CNN network setup with the highest overall accuracy is constructed by using the following parameters:

Table 5-9: Network Parameter configuration for CNN with the highest accuracy

Parameter	Value
Features	Log-mel-spectrogram
Number of convolutional layers	2
Number of neurons in 1st convolutional layer	96
Number of neurons in 2nd convolutional layer	32
Pooling layers	Max Pooling and Global Average Pooling
Activation function for convolutional layers	ReLu
Dropout rate	0.5
Activation function for output layer neurons	SoftMax
Learning rate	0.01
Initial weight and bias range	(-0.001,0.001)

In the results section for CNN, experiments in EAS datasets recognition is done by using the configuration in Table 5-9 as base point. The effect of each parameter on the performance of the CNN recognition accuracy is spanned while keeping all the others fixed. Thus, the aim of observing the effect of individual parameters is to select the optimal parameter values.

5.3.2.1Effects of CNN Parameters

Effect of Features In the case of CNN, we do not evaluate the effect of features, we just leave feature extraction for CNN's convolutional (kernels that perform subsequent nonlinear filtering operations along small context windows of the input) layer. we expect that a CNN can able to autonomously learn new features based on relevant patterns occurring in the training data. Hence, the feature representation we choose for our system is the log-mel spectrogram, which, in addition to being a convenient image like (i.e., a two-dimensional matrix) input for a CNN, is also a well-established and perceptually motivated representation of the audio spectra. Hence, the convolutional layer generates feature maps from the windowed audio frame. Thus, CNN score a recognition accuracy of 93.8 for recognizing EAS dataset as shown in Table 5-11.

Table 5-10: CNN Confusion Matrix as a function of feature extraction technique for recognizing EAS dataset

		Predicted Labels				
	Class	**Carhorn**	**DogB**	**EnginI**	**FireA**	**Gunshot**
True Labels	**Carhorn**	**70**	0	1	0	0
	DogB	0	**59**	1	1	0
	EnginI	0	3	**72**	0	1
	FireA	0	0	0	**60**	0
	Gunshot	0	4	0	0	**69**

Table 5-11: CNN performance for recognizing EAS dataset

	Precision	**Recall**	**F: score**	**Accuracy**
Carhorn	**100**	**98.6**	**99.3**	**98.6**
DogB	**89.4**	**96.7**	**92.9**	**86.8**
EnginI	**97.3**	**94.7**	**96**	**92.3**
FireA	**98.4**	**100**	**99.2**	**98.4**
Gunshot	**98.5**	**94.5**	**96.5**	**93.2**
Avg.	**96.7**	**96.9**	**96.8**	**93.8**

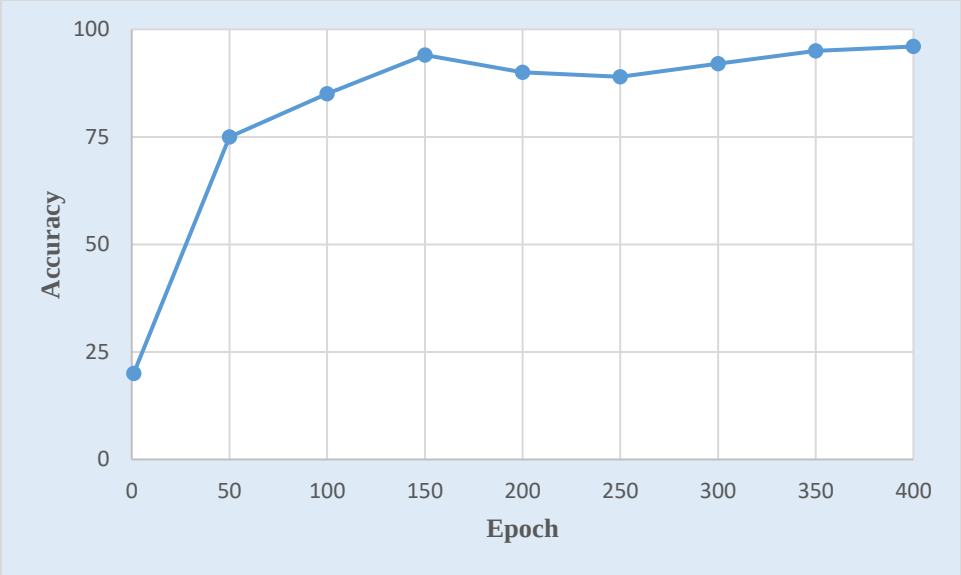

Figure 5-5: CNN performance accuracy vs epoch using spectrogram for recognition of EAS dataset

Effect of Number of Convolutional layers Convolutional layers are characterized by neurons (kernels) that performs subsequent non-linear filtering operation along small context window of the input. Theses convolutional layers determines both the computational time and recognition accuracy of the classifier. Increasing the number of convolutional layers increases the complexity of the classifier which leads slow performance of the system and resource consumption. It also causes for overfitting. On the other hand, using least number of convolutional layers affects the recognition accuracy of the classifier. Thus, using optimal number of convolutional layers have a good significance in the recognition capability of the classifier. In this thesis 2 convolutional layers are used. The number of neurons (filters) in the first convolutional layer is 96 as shown in Table 5-12

Table 5-12: CNN recognition performance as a function of number of neurons in the first convolutional layer

Number of filters	32	64	96	128
Accuracy (%)	92.3	92.3	93.8	90.1

The number of neurons in the second convolutional layer is 32 used in order to grant higher level representation as shown in Table 5-13.

Table 5-13: CNN recognition performance as a function of number of neurons in the second convolutional layer

Number of filters	32	64	96	128
Accuracy (%)	93.8	92.8	92.3	90.4

Effect of Dropout rate Dropout layer is a crucial layer in CNN classifier model that reduce the likelihood of overfitting randomly by setting the weights of the neurons in the network to zero (0). This makes the network robust to smaller variation of the data which increases the accuracy of the network to unseen data. Small percent of the neuron's outputs dropout reasons for overfitting. On the other case higher percent of the neurons output dropout causes to reduction of the network performance since there may be valuable representative features. Therefore, the dropout rate of CNN network as shown in Table 5-14, is 0.5, thus 50% of the neuron receives zero weight when the time of training.

Table 5-14: CNN performance as a function of dropout rate

Dropout rate	0.3	0.4	0.5	0.6
Accuracy (%)	89.9	91.8	93.8	91.3

Effect of Learning Rate Learning rate regulates the size of steps taken in adjusting each weight and bias. Too low learning rate causes to slow convergence as well as network computational time increase. Too high learning rate is a reason for overshooting a global minimum [17]. Optimum learning rate allows to the network to convergence to a high accuracy model as shown in Table 5-15, thus, the learning is set 0.01.

Table 5-15: CNN performance as a function of learning rate

Learning rate	0.005	0.01	0.02	0.03
Accuracy (%)	92.7	93.8	93.1	92.5

5.3.3 Comparison of Classifier's Performance

In this section, we compare two classifier algorithms based on their recognition accuracy and computational speed (running time). Table 5-16 summarizes the overall performance of the classifiers. Based on the table, MLP is not acceptable to be used in its recognition accuracy (93.3%). However, the required computation time is so small compared to CNN computation time that it is appropriate in devices with limited resource devices (CPU, battery and memory). Even though, it is not preferred in recognition of EAS due to its lower performance. The CNN classifier provides high classification performance (93.8%, greater than that of MLP) with high execution time. Therefore, the CNN classifier is the best algorithm that can be used for recognizing unstructured EAS, can be implemented as classification model in portable devices such as smartphones.

Table 5-16: Summery of Classifier's comparison

Classifier	**Dataset**	**Overall Recognition Accuracy**	**Execution Time (**in secs**)**
MLP	EAS dataset	93.3	49.2 seconds
CNN	EAS dataset	93.8	534.82 seconds

So far, we compute and compare the recognition accuracy of the classifiers without looking the effect of other configuration parameters such as audio features, class types, learning rate, activation function, number of hidden layer and number of neurons in hidden layer. In fact, the classification (recognition) accuracy does not only depend on the type of classifier used. There are a number of parameters that affect the recognition accuracy and the computation time as well. The most powerful parameters that determines the recognition performance and computation time are type of audio feature vectors, class types, number of hidden layers and number of neurons in the hidden unit, learning rate, activation function in the hidden units as well as in the output nodes. In section 5.3.1.1 and 5.3.2.1 the effect of these parameters is investigated in the recognition accuracy of MLP and CNN respectively. Though, MLP is not an appropriate classifier for implementing on the smartphone due to its lower recognition accuracy.

5.4 Freezing the Trained Feature and Exporting

This section describes activities performed to save optimized trained feature set and variables (weights and bias) used when training process, in such a way that it can be used by our Android EASR model.

To do this TensorFlow is used [48] that allows placeholders for assigning memory to feed data to the graph; and variables for storing values that adapts after every step of training to improve accuracy. Variables of the model are weights and bias, which initially assign random values. During training their values can be altered as the model tries to fit the correct recognition. In our model, two placeholders used, to feed: optimized trained feature matrix and the corresponding class label matrix.

As discussed earlier in Chapter 4 (subsection 4.3.2), we have used two convolutional layer and two pooling layer in our CNN classifier. Thus, we have five sets of values of variables; each set connecting one layer to another. Each set contains a weight and bias variables, that illustrates the total number of variables we require to describe the model is ten. Then freeze all these variables at the end of the training process to convert variables to constants.

Finally, to save a graph we use a TensorFlow module (i.e., NodeDef, Checkpoint and GraphDef) that aid us to achieve this task.

- **NodeDef**: to define one single operation (task) in the model
- **Checkpoint**: to store variables value during every iteration in the time of training process. As the variables change regularly, these checkpoints are stored periodically to files.
- **GraphDef**: to store all the list of NodeDefs, which used to define the entire executable graph. During executing the graph, if it requires the variables value at any node, their values are taken from the checkpoint files.

As a result, for complete deployment of our trained model in Android mobile device, both GraphDef and Checkpoint files are required. These data are very high volume and have similar data structure. In this case, we need a language-neutral, platform-neutral and extensible way of serializing structured data for use in communications protocols and data storage. Hence, we use Protocol buffer [48] to serialize these structured data. Protocol

buffers (pb) are used for storing and transferring all types of structured data. In this format, first it is necessary to declare the schema, afterwards the data is serialized in the format based on the schema.

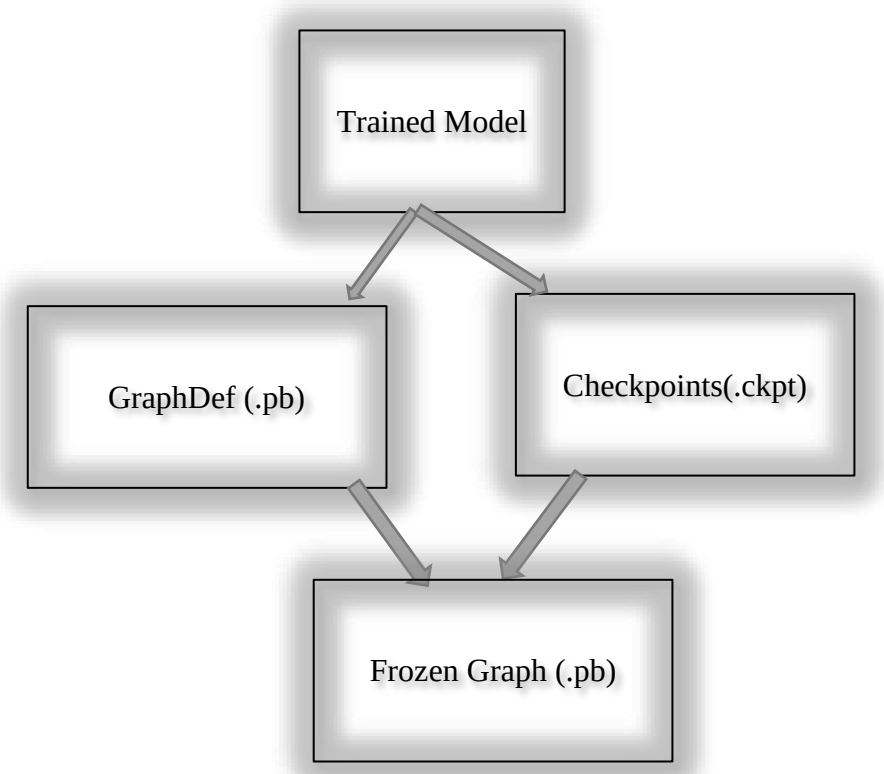

Figure 5-6: Converting trained data to binary GraphDef file that can be loaded into Android mobile

As shown in Figure 5-6, the ‘freeze’ class of TensorFlow takes the (.ckpt) file and (.pb) graph as the input. It changes the variables in the checkpoint to file constants and output a new GraphDef created using these constants.

CHAPTER SIX

6. Android-Based EAS Recognition Result and Analysis

6.1 Introduction

Off-line sound processing can be used for applications whenever on-line audio recognition is not required. For instance, if we are interested in following the daily routine of EAS, the audio recorder can collect the data during a day; and it can then be uploaded to a server at the end of the day to be processed off-line for classification purpose. Though, someone is interested in knowing the current EAS happened, on-line recognition system becomes important. Android-based EASR, is unlike off-line audio recognition, it creates an awareness about the current environment status, what prompts occurred our surrounding especially for those HI individuals. This chapter describes Android-based EASR model performance result.

Since CNN is provided the best performance in the model offline training and testing discussed in Chapter 5, it is selected as classifier algorithm in the development of mobile based EASR model. Besides, to develop the Android-based EASR model, we used the following parameters (i.e. parameter values that outperformed in the model offline training and testing)

- Frozen Graph that consist of trained feature and training variables generated from the offline training, as explained in Chapter 5(Sub section 5.4)
- Analysis window of 50ms.
- Sampling rate of 44.1kHz.
- Log-mel-spectrogram that is a bi-dimensional matrix features for the input to CNN
- Two convolutional layers (96 neurons for the first layer and 32 neurons for the second layer)
- Rectified linear unit (ReLu) activation function in the convolutional layer and SoftMax activation function at the output layer
- TECHNO Camon CX (with 1.5GHz 8 Core-Count system on chip processor) has been used for developing and testing our system.

6.2 Android-Based EAS Recognition Model

Android-based EASR model is developed based on the result of offline model training and testing. The result of offline model (CNN) training is the frozen graph which consists of the trained feature matrix, corresponding class label and checkpoints. Using Android API, we develop EASR model and using frozen graph for crosscheck of features of unknown and already trained features to performs EAS recognition.

Thus, the phone records the stream of audio signal incessantly at the sampling rate of 44.1kHz) and put it into buffer. For processing (feature extraction and classification), the recorded audio data read at a fixed time interval (50ms). The signal is segmented into frames of duration 50ms. This is important to compute audio features in each frame and then average the segment feature to find each audio stream found in the buffer (duration 50ms) feature value. In feature extraction process, we used log-mel-spectrogram which used CNN classifier to perform well (see Chapter 5 subsection 5.3.2.1). Figure 6-1, illustrates the user interface (UI) of Android-based EASR model. The main application user interface contains three buttons; Start, Stop and Exit. When Start button clicked, two very important threads run simultaneously. The former thread th_1, initiates the smartphone to record unknown audio stream (each record with 2seconds duration) and store to the buffer. The later thread th_2, compute features from the stored audio stream in the buffer and then match the extracted feature from the Frozen Graph features matrix; and finally recognize the corresponding class label. Stop button is used to stop recording and recognition process, in time when the user wants to pause the application. Exit button is used to close the application, when the user wants to close the application.

As is shown in the Figure 6-1, the second figure illustrates that the Android model is running and listening the environment, and the third figure notifies the user that Carhorn alarm sound is detected.

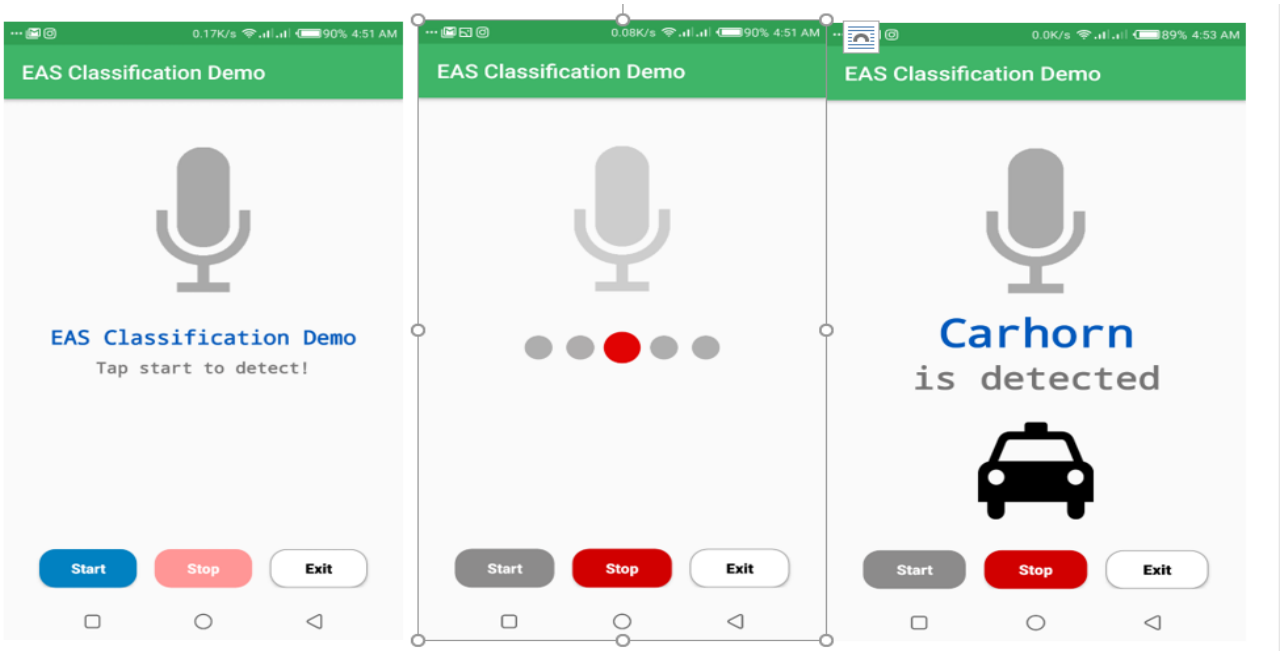

Figure 6-1: The UI of Android-based EASR model

6.3 Performance Result of Android-Based EAS Recognition

The recognition model of Android-based EASR in this work, evaluated based on the experiment result to recognize the EAS. The experiment is performed using recorded EAS, which picked up the smartphone's microphone. The experiment tests the performance of CNN classifier, with log-mel-spectrogram feature, which generated from each frame segment sequence of a 2sec audio signal.

Since, in the model offline training and testing we used 70% of the dataset for training, 10% for validation and 20% for testing as discussed in Chapter 5 (subsection 5.2.2). Therefore, the performance of Android-based EAS recognition measured using on the recognition of 20% of the EAS dataset (347 audio clips) with 10 trials using (TECHNO Camon CX) smartphone. The process performed by playing pre-recorded audio clips, while at the same time the smartphone's EAS recognition application is running. Interaction log files were recorded to analyses the accuracy of Android-based EAS recognition technique including: car horn, dog barking, fire alarm, engine idling and gunshot. The log files have been captured for each EAS recognition event, for each 10 trials.

The Android-based EAS recognition accuracy is evaluated on the basis of performance measures (i.e. Accuracy, Precision, Recall and F-score). Table 6-1, shows the average number of EAS detected per class which was tested on each trial of recognizing EAS attempts. The columns indicate the number of correctly and incorrectly recognized audio classes that are relevant and not relevant. The rows indicate the number of relevant audio events that are retrieved for each class. For example, in Carhorn recognition total 698 audio events were detected. Among this 613 were correctly classified while 85 events recognized incorrectly (36 as FireA, 24 as DogB, 20 as EnginI and 5 as Gunshot). In other words, 36 events were classified as FireA, 24 as DogB, 20 as EnginI and 5 as Gunshot when expected to be classified as Carhorn. When we came to FireA, a total of 659 audio events recognized, from this 611 EAS were recognized correctly and 48 EAS were incorrectly recognized (24 as Carhorn, 15 as DogB and 9 as Gunshot) expected as Carhorn. The detail illustrated in the Table 6-1, as shown in the confusion matrix there is misclassification. This is due to the lack of sophisticated large dataset that allow the CNN classifier generalize EAS effectively during training phase. The Android-based EASR model recognize the event that recorded by the smartphone's microphone. When the smartphone's microphone records the sound, there is noise interreference and also sound overlapping. This is the other reason for misclassification of EAS.

Table 6-1:The average number of recognized EAS logged from 20% EAS dataset

	Carhorn	**FireA**	**DogB**	**EnginI**	**Gunshot**
Carhorn	**613**	24	5	31	13
FireA	36	**611**	21	20	3
DogB	24	15	**536**	27	80
EnginI	20	0	0	**650**	20
Gunshot	5	9	28	12	**598**
Total	**698**	**659**	**590**	**740**	**714**

Hence, we can say that the accuracy of Android-based EAS recognition affected by noise interference and the edge boundaries of the audio clip (the end of one audio class and the starting of another) in addition to annotated dataset scarcity.

Table 6-2, illustrates the average Recall (R), Precision (P), F-score (F) and Accuracy (A) results of recognition model in accordance with 5 EAS: car horn, fire alarm, dog barking, engine idling and gun shot. The average result for all three metrics (R, P and F) were **88.3**%. It is interesting to note that the recognition achieved good accuracy in FireA and EnginI classification **83**%) whereas slightly low in recognition of DogB (**73**%). Generally, the average accuracy rate ranged from **73**%-**83**%. The average accuracy results of Carhorn, FireA, DogB, EnginI and Gunshot recognition were **80**%, **83**%, **73**%, **83**% and **78**% respectively which is positive.

Table 6-2: Average Recall, Precision, F-score and Accuracy of Android-based EASR logged from 20% EAS dataset

Class	P	R	F	A
Carhorn	**0.88**	**0.89**	**0.89**	**0.80**
FireA	**0.93**	**0.88**	**0.91**	**0.83**
DogB	**0.91**	**0.79**	**0.84**	**0.73**
EnginI	**0.88**	**0.94**	**0.91**	**0.83**
Gunshot	**0.84**	**0.92**	**0.88**	**0.78**
Avg.	**0.89**	**0.88**	**0.88**	**0.79**

As indicated in Figure 6-2, **88**%, **93**%, **91**%, **88**% and **84**% recognition attempts were correct in Carhorn, FireA, DogB, EnginI and Gunshot recognition respectively. The average result of correctly and incorrectly recognized EAS were **89**% and **11**% respectively, which means **89**% were correctly recognized while **11**% were recognized incorrectly. As the result shows. FireA scored high result than others, because FireA have fixed pattern (same frequency) across the audio signal, which is easily learnt by the classifier. However, DogB, Gunshot, Carhorn and EnginI have different pattern across different dog, gun, car and engine respectively. So, to achieve good performance on these events, vast amount of labelled data required for training purpose.

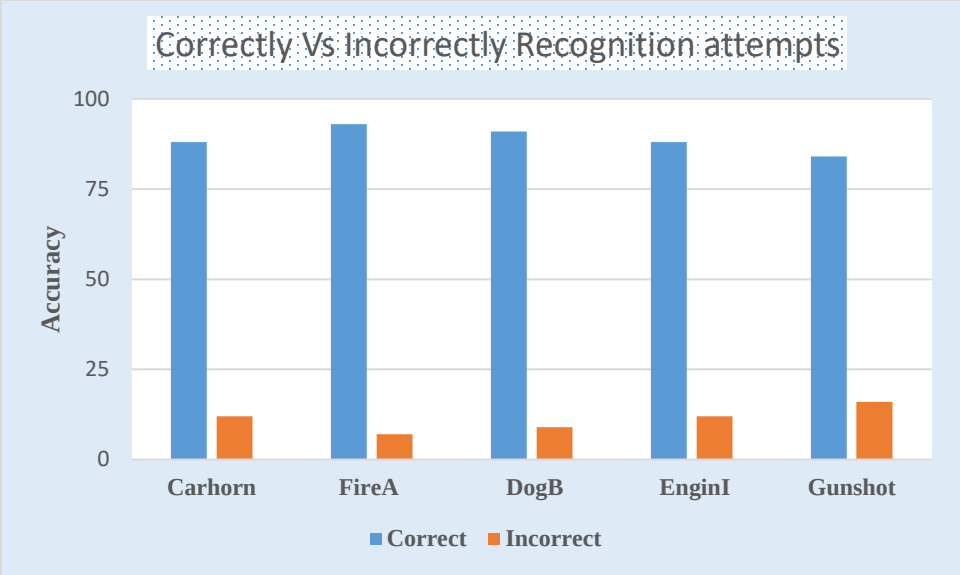

Figure 6-2: Correct Vs incorrect recognition attempts of Android-based EAS recognition model

6.3.1 Comparison of Recognition Performance of Android-based Vs Offline EAS Recognition Model

The purpose of this comparison and contrast is to determine the effect of the platform on the EASR model performance. We use a file of 150sec duration of alarm event that contains a continuous sound comprising of all environmental alarms (i.e., car horn, dog barking, fire alarm, engine idling and gunshot) to compare the performance of Android-based EAS recognition technique with that of offline EAS recognition technique. This continuous audio file is played while at the same time Android-based EAS recognition model is running. The classification result of both offline EAS recognition technique and Android-based EAS recognition technique is shown in Figure 6-3. As is shown in the figure, we compare the result of offline EAS recognition technique and Android-based EAS recognition technique with that of the ground truth (the actual EAS class) that have been labeled manually.

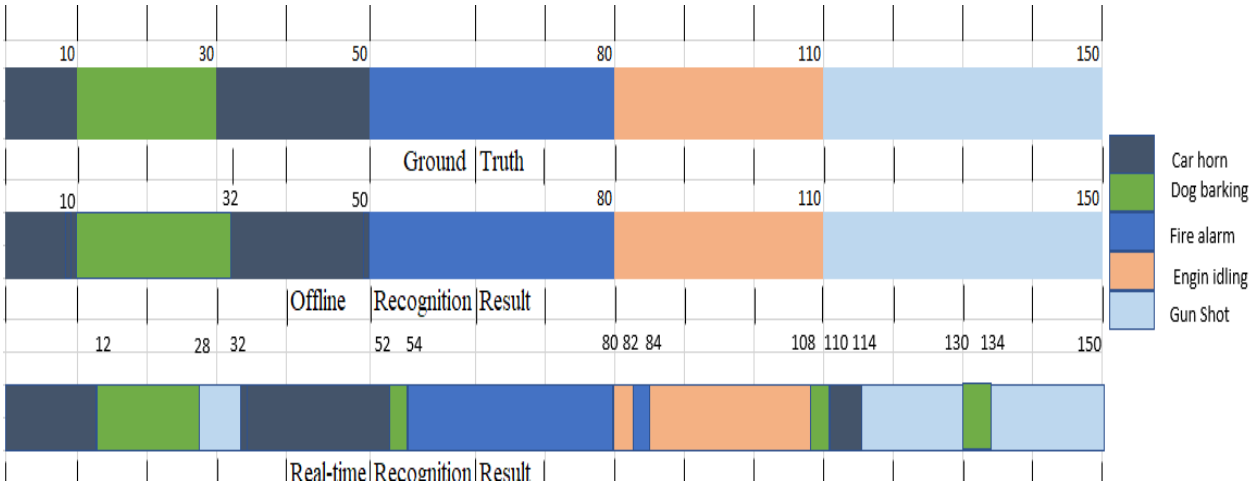

Figure 6-3: Comparison of Android-based Vs Offline EAS recognition technique performance

The numbers in the X-axis denotes the start and end time (in secs) of each individual audio class after merging similar consecutive EAS types. It is important to note that the minimum duration of a single EAS is 2sec (duration of analysis window for feature extraction and classification).

In 150sec alarm event, we have in a total 75 EAS. The Android-based EAS recognition technique incorrectly classified 10 out of 75 alarm events. Based on the comparison, the recognition performance of Android-based EAS recognition technique achieves **86.7**%, which is relatively less than that of the logged recognition performance. As is shown in the Figure 6-3, the result of offline EAS recognition technique (performed using python) is likely to the ground truth excluding for a very short time period. When we come to the result of Android-based EAS recognition has a greater number of incorrectly classified audio segments. The main causes of for lower recognition performance of Android-based EAS recognition technique are:

- The first cause for lower recognition performance of the Android-based EAS recognition technique is noise interference. The Android-based EAS recognition technique is more sensitive to noise interference. Because, during the recognition the smartphone's microphone can record unwanted and unknown nearby audio events. This provides to have noise interference and affects the classification performance.

- The second point is audio event overlapping at edge boundaries. As is shown in the Figure 6-3, the rate of misclassification is higher at the edge boundaries (the end of one event and the start of another event).
- Lastly, offline EAS recognition technique performed with training the classifier by the labeled dataset and then apply the test dataset (unknown) to classify. During recognition, the more overlapping between the training data and testing data, the more recognition result in better accuracy. This is the reason that offline EAS recognition technique to have higher recognition accuracy.

6.3.2 Discussion of the Result

As we have presented in the previous section in detail, the Android-based EAS recognition model constructed using the frozen graph that consists the trained feature, the corresponding class label and checkpoints (variables which comprised weights and biases value for each training phase). To as the best of researcher's knowledge, no research work has done in the area of Android-based EAS recognition using Android smartphones, this study is the first attempt in the area. The feature sets used by this study is a time-frequency domain feature (log-mel-spectrogram). Then as the recognition model CNN is used (selected from offline training and testing with high performance). In order to test the performance of Android-based EASR model, 20% of EAS dataset audio clips used (70% of the dataset was used for training and 10% of the datasets was used for validation in offline model training), and then the model is tested 10 times. During testing interaction log files recorded for each event detection. In general, the overall result showed that the proposed approach with the spectro-temporal feature with CNN classifier have better classification performance.

6.3.3 Running Time

The battery consumption of our model is tested by measuring model running time for 10 repetitions, each starting from fully charged phones. The test is held by using two smartphones: Techno Camon CX and Huawei G630. During each trail, the application is continuously running and displaying the recognition result every second. Additionally, the connection to internet is switched off and the background processes of the phones not stopped. The running time is measured until the phone switched off due to low battery. As shown in Figure 6-4, the running time of Techno Camon CX and Huawei G630 achieved an average of 16.23h and 13.21h respectively. Both phones running time show less usage of battery power. The reason for this lower usage of battery power (high running time) is that the model developed used a protocol buffer which is very simpler, smaller and faster than XML as well as JOSON, which enables data transmission very fast [48].

We do not compare our model battery consumption with that of previous efforts, because we did not find the application that done with that previous studies, to measure running time by our own smartphone. But we used running time obtained by their smartphones scale to compare with our work.

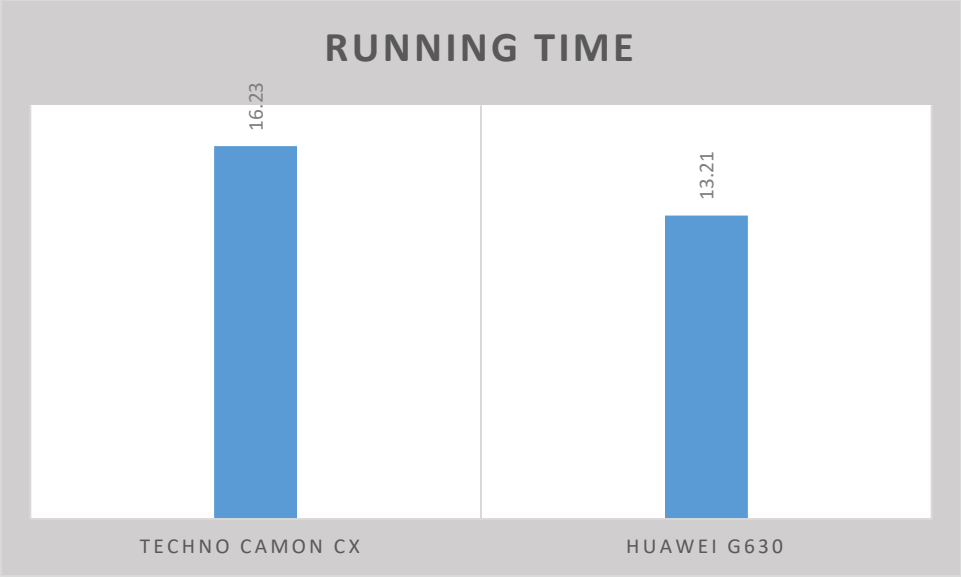

Figure 6-4: Average runtime of the testing devices in Android-based EAS recognition model

As shown in the Figure 6-4, we note that the model developed in this work is compatible with that of the limited resource device. Even though, the aim of the study is not resource optimization, the model considers the limited resource of smartphones. Thus, this work outperforms the state of the art and is compatible with smartphone's resource requirement.

CHAPTER SEVEN

7. Conclusion and Recommendations

7.1 Conclusions

EAS is a crucial way of creating awareness of hazard events that just occurred in the real-life environment. From the alarm sounds we can understand what is going on the environment specifically for events out of sight and attention. In this thesis, we tackled the area of design and development of a model for correctly detecting and recognizing EAS using audio signals on mobile phones for hearing impaired. We compared performance of different ANN (CNN and MLP) audio classifiers and sound features based on their recognition accuracy and computational speed to select the optimal technique for developing the model on Android smartphone. In order to find the classifiers useful range of parameters that significantly speed up learning process and result in better accuracy, many experiments have been conducted. The effect of several parameters and classifiers recognition accuracy have been experimented and explained. We found out that CNN classifier achieves high recognition accuracy.

We also, evaluated the performance of different audio features both time-frequency and frequency domain to choose the optimal feature vector for the audio classifier in the case of MLP. Feature sets comprised of Mel Frequency Cepstral Coefficient (MFCC), Mel spectrogram (Mel) and Spectral contrast (SC) outperforms (when used with MLP classifier), thus chosen as optimal feature set. This show that the combination of time-frequency and frequency domain audio features were robust in recognition of EAS and useful. In the case of CNN, we used log-mel-spectrogram as input for CNN classifier that allow convolutional layers of the network can able to extract useful training features and then perform the recognition task. The experiment is conducted offline using python offline training and testing. The Android-based EAS recognition model, which is implemented as an Android app, uses CNN as audio classifier and the selected optimal audio feature set as audio feature vector. The application continuously classifies alarm events (alarm type) by analyzing EAS sampled from smartphone's microphone. The result of Android-based EAS

recognition technique show that the intended goal of this work moderately (comparatively) achieved.

7.2 Recommendations

The result of Android-based EAS recognition technique performs poorly when there is audio overlapping (more than one audio event occurs simultaneously) and noise interference. This limitation needed to extend noise reduction and audio segmentation techniques at the pre-processing stage to improve the recognition accuracy. And also, multilabel classification technique in case of overlapped events.

In this work, the training is performed offline and the Android-based EAS recognition cannot recognize new EAS as well as unknown environmental sound events. Hence, in the future study it is needed online training technique to adapt the dynamic nature of the environment and changing audio types.

As discussed, when the Android-based EAS recognition application running the smartphone's microphone records audio event in every 2sec, and the application also recognizes every 2 sec regardless whether the occurrence of EAS or not. This process drains the smartphone's battery and consumes CPU time. The future study needed adaptive duty cycling technique that performs detection and recognition by optimizing the devices resources, which is based on switching the smartphone's microphone such adaptation state.

Lastly, I recommend that CNN classifier recognition accuracy more effective in the presence of large training dataset. Thus, the future research work extended by enlarging the dataset as well as the number of EAS classes to improve the recognition and expand the Android-based EASR model.

REFERENCES

[1] S. N.-C. J. K. Selina Chu, "Content Analysis for Acoustic Environment Classification in Mobile Robots," *American Association for Artificial Intelligence,* 2006.

[2] ISO7731:, Ergonomics - Danger signals for public and work areas - Auditory danger signals, International Organization for Standardization, 2013.

[3] C. Selina, "Unstructured Audio Classification for Environment Recognition," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence,* Los Angeles, CA 90089, USA, 2008.

[4] L. V. P. Vasileios Bountourakis, "Machine Learning Algorithms for Environmental Sound Recognition: Towards Soundscape Semantics," *ACM. ISBN 978-1-4503-3896-7,* October 2015.

[5] S. N. a. J. K. Selina Chu, "Environmental Sound Recognition With Time–Frequency Audio Features," *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING,* vol. 17, no. 6, AUGUST 2009.

[6] M. a. H. S. a. E. L. a. M. K. Mekonnen, "Socio-emotional problems experienced by deaf and hard of hearing students in Ethiopia," *Deafness and Education International,* vol. 17, no. 3, pp. 155-162, 2015.

[7] N. H. a. R. E. L. Danielle Bragg, "A Personalizable Mobile Sound Detector App Design for Deaf and Hard-of-Hearing Users," *ASSETS,* October 2016.

[8] K. A. A. P. a. S. M. Angelos Pillos, "A REAL-TIME ENVIRONMENTAL SOUND RECOGNITION SYSTEM FOR THE ANDROID OS," *Detection and Classification of Acoustic Scenes and Events,* September 2016.

[9] Wai-ling Ho-Ching, Jennifer Mankoff, James Landay, "Can you see what I hear? The Design and Evaluation of a Peripheral Sound Display for the Deaf," *Group for User Interface Research, Computer Science Division,* April 2003.

[10] Alhabbash et.al., "Sound Visualization for Deaf Assistance Using Mobile Computing," *JOURNAL OF ENGINEERING RESEARCH AND TECHNOLOGY,* vol. 2, no. 2, JUNE 2015.

[11] G. T. Abreha, "An Environmental Audio–Based Context Recognition System Using Smartphones," University of Twente, MSc Thesis, August 2014.

[12] J. L. S. a. J. D. NIGUIDULA, “FILE CONVERSION AFTERMATH: ANALYSIS OF AUDIO FILE STRUCTURE FORMAT,” in *CMSIT 2017: 4th International Conference on Management Science, Innovation, and Technology*, 2017.

[13] Saurabh B et. al., “Android Operating Systems,” *International Journal of Engineering Technology & Management Research,* vol. 1, no. 1, pp. 147-150, February 2013.

[14] M. Cowling, “Non-Speech Environmental Sound Classification System For Autonomous Survillance,” Griffith University, March 2004.

[15] L. B. A. a. F. P. Alain Dufaux, “AUTOMATIC SOUND DETECTION AND RECOGNITION FOR NOISY ENVIRONMENT,” *Institute of Microtechnology, University of Neuchâtel,* Jun 2013.

[16] S. F. O. A. N. B. S. M. a. G. T. Mirco Rossi, “AmbientSense: A Real-Time Ambient Sound Recognition System for Smartphones,” in *International Workshop on the Impact of Human Mobility in Pervasive Systems and Applications*, San Diego, March 2013.

[17] E. Cakir, “Multilabel Sound Event Classification with Neural Networks,” TAMPERE UNIVERSITY OF TECHNOLOGY, Tampere, September 2014.

[18] M. U. A. A. H. A. S. K. B. a. A. W. Umair Zafar Khan, “Detection of Acoustic Events by using MFCC and Spectro Temporal Gabor Filterbank Features,” *CSPS2016*, November 2016.

[19] S. S. a. Z. Lachiri, “Using Spectro-Temporal Features for Environmental Sounds Recognition,” *American Journal of Circuits, Systems and Signal Processing,* vol. 1, no. 3, pp. 60-68, 2015.

[20] D. F.-S. E. B.-V. G. B. B. a. A. B. Sebastien Tremblay, “Exploiting Environmental Sounds for Activity Recognition in Smart Homes,” in *Artificial Intelligence Applied to Assistive Technologies and Smart Environments*, Canada, 2015.

[21] Sivaprakasam and Dhanalakshmi, “A Robust Environmental Sound Recognition System using Frequency Domain Features,” *International Journal of Computer Applications,* vol. 80, no. 9, October 2013.

[22] Emre Cakır et.al., “Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection,” *CSC IT Center for Science,* February 2017.

[23] S. A.-Z. a. M. AlQahtani, “Audio Environment Recognition using Zero Crossing Features and MPEG-7 Descriptors,” *Journal of Computer Science,* vol. 6, no. 11, pp. 1283-1287, 2010.

[24] D. a. P. S. Patil, “Zero crossing rate and Energy of the Speech Signal of Devanagari Script,” *Journal of VLSI and Signal Processing (IOSR-JVSP),* vol. 4, no. 1, pp. PP 01-05, January 2014.

[25] P. GIAMBATTISTA, “RECURRENT NEURAL NETWORKS FOR POLYPHONIC SOUND EVENT DETECTION,” TAMPERE UNIVERSITY OF TECHNOLOGY, TAMPERE, November 2015.

[26] J. S. a. J. P. Bello, “Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification,” *EEE SIGNAL PROCESSING LETTERS,* NOVEMBER 2016.

[27] J. J. J. Z. a. Y. D. Xiaocheng Luo, “Parallel Algorithm Design for Audio Feature Extraction,” in *5th International Conference on Machinery, Materials and Computing Technology (ICMMCT 2017)*, Hunan, China, 2017.

[28] K. W. a. S. Cox, “FEATURES AND CLASSIFIERS FOR THE AUTOMATIC CLASSIFICATION OF MUSICAL AUDIO SIGNALS,” in *ISMIR*, 2004.

[29] J. Z. J. L. W. W. a. W. Y. Xiaoqing Yu, “An Audio Retrieval Method Based on Chromagram and Distance Metrics,” *ICALIP,* 2010.

[30] M. A. B. a. G. H. Wakefield, “Audio Thumbnailing of Popular Music Using Chroma-Based Representations,” *IEEE TRANSACTIONS ON MULTIMEDIA,* vol. 7, no. 1, FEBRUARY 2005.

[31] P. G. V. K. a. M. M. Nanzhu Jiang, “Analyzing Chroma Feature Types for Automated Chord Recognition,” in *AES 42ND INTERNATIONAL CONFERENCE*, Germany, July 2011.

[32] H. S. a. P. K. Ajmera, “Interweaving Convolutions: An application to Audio Classification,” in *In Proceedings of KDD Deep Learning Day (KDD’18),* New York, NY, USA, 2018.

[33] C. M. T. a. D. S.-R. Juan J. Noda, “Fusion of Linear and Mel Frequency Cepstral Coefficients for Automatic Classification of Reptiles,” *Applied Sciences,* vol. 7, no. 2, p. 178, 2017.

[34] M. S. R. K. a. M. A. Syrine Ben Driss, “A comparison study between MLP and Convolutional Neural Network models for character recognition,” in *SPIE Conference on Real-Time Image and Video Processing*, Anaheim, CA, United States, Apr 2017.

[35] I. M. E.-E. a. S. A. N. Jamal M. Nazzal, “Multilayer Perceptron Neural Network (MLPs) For Analyzing the Properties of Jordan Oil Shale,” *World Applied Sciences Journal,* vol. 5, no. 5, pp. 546-552, January 2008.

[36] K. J. Piczak, “ENVIRONMENTAL SOUND CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS,” in *IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING*, BOSTON, USA, Septeber 2015.

[37] P. B. M. M. J. D. a. M. A. R. Stefan R. Maetschke, “Supervised, semi-supervised and unsupervised inference of gene regulatory networks,” *Briefings in Bioinformatics*, vol. 15, no. 2, pp. 195-211, March 2014,.

[38] Dai Wei et.al., “Acoustic Scene Recognition with Deep Neural Networks (DCASE challenge 2016),” in *Detection and Classification of Acoustic Scenes and Events 2016*, Budapest, Hungary, 2016.

[39] A. F. M. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” *arXiv:1803.08375v2,* 2018.

[40] J. P. K. L. a. Y. H. Hyungui Lim, “RARE SOUND EVENT DETECTION USING 1D CONVOLUTIONAL RECURRENT NEURAL NETWORKS,” in *Detection and Classification of Acoustic Scenes and Events 2017*, Munich, Germany, 16 November 2017.

[41] J. a. S. M. V. a. N. a. S. S. Gibson, “Comparing time-frequency representations for directional derivative features,” in *Fifteenth Annual Conference of the International Speech Communication Association,* 2014.

[42] W. S. M. S. M. Y. S. N. a. M. M. D Anggraeni, “The Implementation of Speech Recognition using Mel-Frequency Cepstrum Coefficients (MFCC) and Support Vector Machine (SVM) method based on Python to Control Robot Arm,” in *The 2nd Annual Applied Science and Engineering Conference*, Indonesia, 2017.

[43] K. J. Piczak, “ESC: Dataset for Environmental Sound Classification,” in *in Proceedings of the ACM International Conference on Multimedia*, Poland, October, 2015.

[44] Justin Salamon et.al., “A Dataset and Taxonomy for Urban Sound Research,” in *in Proceedings of the ACM International Conference on Multimedia*, New York University, November, 2014.

[45] Z. Reitermanov´a, “Data Splitting,” in *WDS'10 Proceedings of Contributed Papers*, Czech Republic, 2010.

[46] A. K. M. a. K. Kochersberger, “Unattended sensor using deep machine learning techniques for rapid response applications,” in *Event: SPIE Defense + Security*, Florida, 2018.

[47] T. V. a. H. H. Oguzhan Gencoglu, “RECOGNITION OF ACOUSTIC EVENTS USING DEEP NEURAL NETWORKS,” in *22nd European Signal Processing Conference (EUSIPCO)*, 33720 Tampere, Finland, 2014.

[48] H. C. T. a. A. Thakur, “Talos App: On-device Machine Learning Using TensorFlow to Detect Android Malware,” in *Fifth International Conference on Internet of Things: Systems, Management and Security (IoTSMS)*, India, December 2018.