University of Twente

Master Thesis

An Environmental Audio–Based Context Recognition System Using Smartphones

Author:
Gebremedhin T. Abreha

Supervisor:
Dr. Nirvana Meratnia

Committee:
Prof. Paul Havinga
Ir. Bert Molenkamp

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science in Embedded Systems

Pervasive Systems Chair
Faculty of Electrical Engineering, Mathematics and Computer Science

August 2014

University of Twente

Faculty of Electrical Engineering, Mathematics and Computer Science

Master of Science in Embedded Systems

An Environmental Audio–Based Context Recognition System Using Smartphones

by Gebremedhin T. Abreha

Abstract

Environmental sound/audio is a rich source of information that can be used to infer a person's context in daily life. Almost every activity produces some sound patterns, e.g., speaking, walking, washing, or typing on computer. Most locations have usually a specific sound pattern too, e.g., restaurants, offices or streets. This thesis addresses the design and development of an application for real-time detection and recognition of user activities using audio signals on mobile phones. The audio recognition application increases the capability, intelligence and feature of the mobile phones and, thus, increases the convenience of the users. For example, a smartphone can automatically go into a silent mode while entering a meeting or provide information customized to the location of the user. However, mobile phones have limited power and capabilities in terms of CPU, memory and energy supply. As a result, it is important that the design of audio recognition application meets the limited resources of the mobile phones. In this thesis we compare performance of different audio classifiers (k-NN, SVM and GMM) and audio feature extraction techniques based on their recognition accuracy and computational speed in order to select the optimal ones. We evaluate the performance of the audio event recognition techniques on a set of 6 daily life sound classes (coffee machine brewing, water tape (hand washing), walking, elevator, door opening/closing, and silence). Test results show that the k-NN classifier (when used with mel-frequency cepstral coefficients (MFCCs), spectral entropy (SE) and spectral centroid (SC) audio features) outperforms other audio classifiers in terms of recognition accuracy and execution time. The audio features are selected based on simulation results and proved to be optimal features. An online audio event recognition application is then implemented as

an Android app (on mobile phones) using the k-NN classifier and the selected optimal audio features. The application continuously classifies audio events (user activities) by analyzing environmental sounds sampled from smartphone's microphone. It provides a user with real-time display of the recognized context (activity). The impact of other parameters such as analysis window and overlapping size on the performance of audio recognition is also analyzed. The test result shows that varying the parameters does not have significant impact on the performance of the audio recognition technique. Moreover, we also compared online audio recognition results of the same classifier set (i.e., k-NN) with that of the off-line classification results.

Acknowledgements

First of all, I would like to thank Almighty God, who has blessed and guided me so that I am able to accomplish this thesis.

In this very special occasion, I would like to express my deepest gratitude and appreciation to my Supervisor, Dr. Nirvana Meratnia, who gave her valuable time, guidance, advice, criticism and corrections to the thesis from the beginning till the end. She was always available for my questions and she was positive and gave generously her time and vast knowledge. I also want to thank all of the lecturers and professors of the Faculty who have thought and guided me during the years of my study at the University. In addition, I would like to thank the University of Twente Scholarship for providing me with financial help and funding, without which it would not have been possible to successfully finish my study.

Contents

List of Figures

List of Tables

Dedicated to my Parents.

Chapter 1

Introduction

As modern science and information technology advances, device sizes are becoming smaller and more operations are now feasible on smaller devices. For instance, mobile devices, such as smart-phone, not only do they work as a telephone, but also their role now have expanded to taking pictures, texting/receiving messages, playing music/videos, keeping appointments, etc. Nevertheless, people still want to access or obtain more intelligent and intuitive knowledge anytime and anywhere using their mobile devices. The rapid increase in speed and capacity of smart mobiles or embedded devices equipped with sensors and powerful processors (CPUs) is expected to allow the inclusion of more applications that can increase the capability, intelligence and feature of mobile devices.

One of the key anticipated future capabilities of smart devices is Context Awareness (CA). CA enables mobile devices to sense and recognize user's contextual information such as user activities, surrounding environment, and provide context relevant information for user's current needs. Many sources such as microphone, camera, gyroscope, accelerometer, luminance, Global Positioning System (GPS), and etc., are available for sensing and capturing various types of contextual information. In audio based context awareness systems, environmental sounds are used to obtain contextual information such as the type of environment (location context), activities (what a user is doing) and what activities/evets are going on in a specific location [1–6].

Audio based CA applications provide mobile device (phone) with the ability to automatically know the context of a given environment and use its knowledge to respond to the mobile user in the most appropraite way. In other words, the CA system enables

a cellphone to change automatically the notification or operation mode based on the knowledge of the user's surrounding. For example, a mobile phone can dynamically switch from a ringing mode to a vibration or silence mode when a user enters into a meeting room or holds a presentation, and in contrast, it may ring louder when the user is in a noisy place, e.g. a street. Similarly, if a user receives a call while she or he is in a meeting, the mobile phone can automatically send a message to the caller saying that she or he is in a meeting. Audio based CA systems has been also used in robot navigation [7, 8], audio based surveillance systems [9], audio based forensics [10], hearing aid [11], home-monitoring environment for assisting elderly people living alone in their own home [12, 13] or for a smart home [14].

Auditory signals are chosen for a number of reasons. Firstly, among the human senses, hearing is second only to vision in recognizing social and conceptual settings; this is due partly to the richness in information of audio signals. Secondly, cheap but practical microphones can be embedded in almost all types of places or mobile devices, including PDAs and mobile phones. Thirdly, auditory-based context recognition consumes significantly fewer computing resources than camera-based context recognition. In addition, unlike visual sources of information such as camera and video, audio information cannot be obscured by solid objects and it is multidirectional, i.e., it can be received from any direction. Additionally, audio data is less sensitive to the location and orientation of the phone as compared with other common sensors such as cameras and accelerometers.

Humans can easily segregate and recognize one sound source from an acoustic mixture, such as certain voice from a busy (noisy) background including other people talking and music. The study of sound analysis, which aims to separate and recognize mixture of sound sources present in an auditory scene, is broadly known in the literature as Computational Auditory Scene Analysis (CASA) [15]. CASA aims to enable computers hear and understand audio content much as humans do. Due to its broad nature, the study of CASA is usually dealt with by dividing into three main research topic areas [15]: 1) Context awareness (recognition of audio context) - dealing with recognition of context such as location or activity happening in a given environment. (answers "where" e.g. restaurant, inside a car) based on the audio information/events, 2) Sound event detection and recognition – dealing with categorization of individual sound events present in the auditory scene (answer "who and what", e.g., recognition of sound sources), 3) General

audio classification – dealing with classification and recognition of the contents of audio signals, e.g., for audio content retrieval, indexing, and audio based searching.

This thesis deals with sound event detection and recognition also referred as environmental sound/audio recognition (ESR). The detected sound events can then be used for the purpose of context recognition. For example, the sound event of keyboard typing helps to know that the user is in his/her office, which is, in this case, location context.

1.1 General Challenges of Environmental Audio Classification and Recognition

In this section, main challenges that are faced in ESR and classification are pointed out. Unlike speech or music signals, environmental acoustic signals are difficult to model due to its high unpredictable nature. Speech or music can be categorized to structured sounds due to their formantic or harmonic structure characteristic whereas environmental sounds, on the other side, are typically unstructured, which have a broad noise-like flat spectrum and diverse variety of signal composition and are difficult to build models. Analysis of real-world audio that consists of a rich mix of naturally occurring sounds such as the environmental sound is complex. As a result, classification and processing of environmental sound is generally more cumbersome compared with that of speech or music. The following are the general challenges that are faced during the design and implementation of ESR technique:

- **Overlap in time and/or frequency content** - Different sound events can happen at the same time which makes recognition of the type of sound event difficult. This leads to two challenging tasks: detection of individual sound events within the audio scene (segmentation) and classification. A system involved in the first task has as a goal to cluster mixed sound events from different sources into their corresponding source type, or try to segment the audio into pieces that represent a single occurrence of a specific event class by estimating the start and end time of each event and if necessary separating it from other overlapping events. The aim of the second task is to characterize and identify the type of sound event (e.g., label the environment in which the audio was recorded). Thus, the overlapping sound

events that constitute a natural auditory scene (environmental sound) create an acoustic mixture signal that is more difficult to handle.

- **Dynamic nature of environment** – Apart from containing a wide variety of sound classes, environmental sound has a dynamic nature, i.e., new sound types (classes) can appear and existing sound classes can disappear randomly at any time. Similarly, mobile devices can move from one environment to another environment and may encounter new types of sound. Therefore, the ESR technique has to deal with and adapt to the dynamic nature of an environmental sound.

- **Selection of feature set** - Audio features have a significant impact on the recognition accuracy. Thus, the definition and extraction of the right type of feature sets is a very important step in ESR. However, it is challenging step too. What feature types we define and how we use them depends on the type of application. For example, audio features used for audio classification in indoors might not perform well when used for the classification of types of sounds in outdoor areas.

1.2 Smartphone Specific Challenges

In addition to the above mentioned general challenges, there are also special challenges that have to be dealt with in order to implement audio based CA technique on mobile devices/smartphones. While smartphones continue to provide more computation, memory, storage, sensing, and communication bandwidth, the phone is still a resource-limited device if complex signal processing and inference are required. Signal processing and machine learning algorithms can stress the resources of the phones in different ways: some require the CPU to process large volumes of sensor data (e.g., interpreting audio data), some need frequent sampling of energy expensive sensors (e.g., GPS), while others require real-time inference. Different applications place different requirements on the execution of these algorithms. For example, for applications that are user initiated the latency of the operation is important. Applications (e.g., healthcare) that require continuous sensing will often require real-time processing and classification of the incoming stream of sensor data. We believe continuous sensing can enable a new class of real-time applications in the future, but these applications may be more resource demanding.

Early deployments of phone sensing systems tended to trade off accuracy for lower resource usage by implementing algorithms that require less computation or a reduced amount of sensor data. Limited power supply and real time requirement are the most common issues that have to be addressed while implementing online (real-time) context awareness system on smartphones.

- **Limited power supply**- Mobile devices have limited power supply and hardware capabilities. Most of the previous researches on audio based CA techniques mainly focus on improving the accuracy of ESR and do not address the complexity of the algorithms used. It implies that a number of algorithms which are often used and proposed in many of the literature for the implementation of ESR may not be suitable to directly implement them on mobile devices. For continuous sensing to be viable there need to be breakthroughs in low-energy algorithms while maintaining the necessary application fidelity. Thus, it is always a challenging problem to find algorithms with less complexity, which consumes less power, without degrading classification accuracy. Hence, performance and energy consumption trade-offs must be sought.

  One strategy for reducing energy consumption is to trade off accuracy for lower resource usage by implementing algorithms that require less computation or a reduced amount of sensor data. Another strategy to reduce resource usage is to leverage cloud infrastructure where different sensor data processing stages are offloaded to back-end servers when possible. Typically, raw data collected by the phone is not sent over the air due to the energy cost of transmission, but rather compressed summaries (i.e., extracted features from the raw sensor data) are sent. The drawback to these approaches is that they are seldom sufficiently energy-efficient to be applied to continuous sensing scenarios. Other techniques rely on adopting a variety of duty cycling techniques that manage the sleep cycle of sensing components on the phone in order to trade off the amount of battery consumed against sensing fidelity and latency. However, this technique is not feasible for applications that require continuous (real-time) sensing with high sampling rate (e.g., 16 kHz) such as in our case.

- **Real-time requirement**- the computational complexity of an audio feature extraction and classification algorithms are a critical factor especially in real-time

applications. While feature extraction on standard PCs is often possible in real-time, applications on mobile devices, such as PDAs and mobile phones, due to limited available resources, pose novel challenges to meet the real-time requirement.

1.3 Thesis Objectives

The main objective of the thesis is to design and develop an application in order to correctly detect and recognize environmental context using audio signals on mobile phones. Humans can easily tell the types of activities (contexts) such as human walking, talking, laughing, coffee machine brewing, printing, door opening/closing, etc. based on the sound produced by each of the activities. This thesis aims to develop methods that enable a computer/machine to do the same.

The realization of the CA technique on mobile devices has to cope with special challenges such as limited processing speed, power (energy) supply constraints and memory of the mobile phones. It is usually possible to obtain highest recognition accuracy using sophisticated and advanced feature extraction and classification techniques. However, such techniques are computationally intensive. In this thesis we need to use algorithms with low complexity without deteriorating the recognition accuracy. Thus, it is the objective of the thesis to optimize the sound recognition technique with respect to the accuracy (recognition rate) versus computational speed trade-off.

The recognition technique takes into account parameters such as device operating parameters (sampling rate and duration), number and types of features and classifier choices. The selection of audio features and classifiers affects both the recognition accuracy and computational speed. It is assumed that the computational speed (execution time) is directly proportional to the energy (power) consumption of the mobile device. We evaluate the impact of these parameters (audio feature and classifier) on the recognition accuracy as well as the computational speed.

1.4 Methodology

Like many other pattern classification tasks, audio classification is made up of three fundamental components: (1) Sensing component - for measuring the sound event or signal;(2) audio processing component - for extracting the characteristic features of the measured sound signal; and (3) classification component - for recognition of the context of the sound event.

In audio based CA applications, the sensing (measurement) is normally done using microphones. The audio signal processing part mainly deals with the extraction of features from the recorded audio signal. The various methods of time-frequency analysis developed for processing audio signals, in many cases originally developed for speech processing, are used. That is feature extraction quantizes the audio signal and transforms it into various characteristic features. This results in n dimensional feature vector often representing each audio frame. A classifier then takes this feature vector and determines what it represents - that is, it determines context of the audio event.

Figure 1.1 shows the general architecture of the audio classification system. In the figure, input represents the raw audio data whereas output represents the activity (context) information.

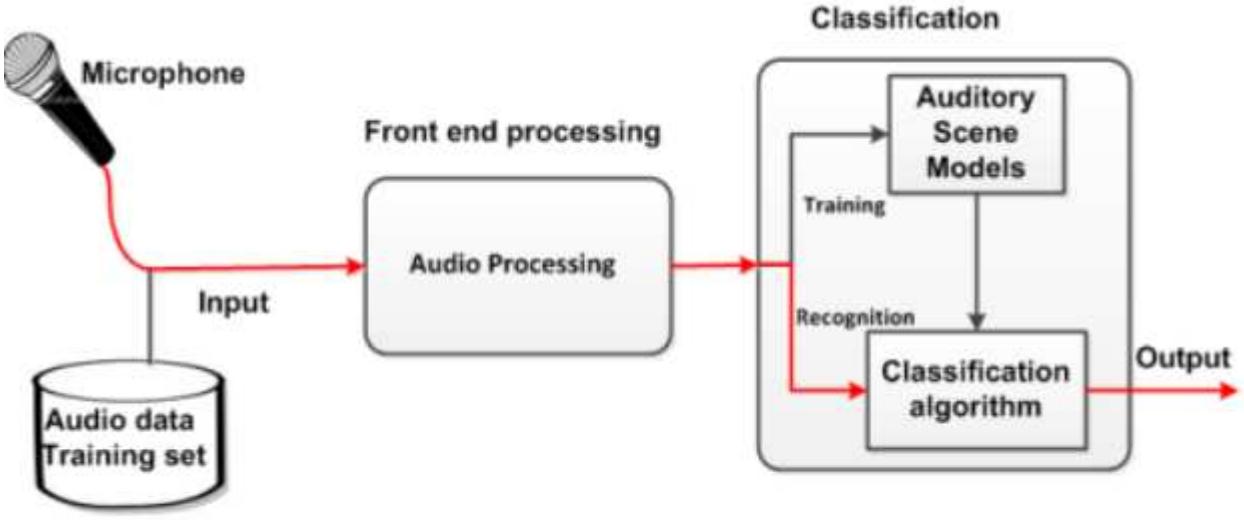


FIGURE 1.1: General architecture of environmental sound recognition technique

The ESR technique has two phases: training phase and recognition phase. During the training phase the system receives its inputs from pre-recorded audio data training sets and generates representative models for each of the audio event/scene. On the other hand, during the recognition phase, the system receives its audio inputs directly from the smartphone's microphone. The recognition phase uses the models generated during the training phase for matching and determining the type of audio received by the microphone. The recognition phase processes the audio data online and in-time without delay in order to deliver continuous and real-time recognition output for the user. Detailed discussion about the process and steps of the ESR technique is provided in chapter 5 (design procedures).

The following are the main procedures that have been followed during the design of the ESR/CA technique.

1. First, thorough literature study of (state-of-the-art) audio feature extraction techniques and classification algorithms is conducted. The main goal of the preliminary literature study is to pre-select the best set of audio feature and classification algorithm combinations that can provide the highest possible recognition accuracy with less computational complexities. This step is performed during the research topic study (literature study)[1]

2. Offline test/simulation 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 environmental sound recognition (ESR) is not yet well matured. It is still at its infant stage which makes it difficult to obtain standard procedures and well organized information to determine the best audio feature extraction techniques and classification algorithms, based solely on the literature study. As a result, it is imperative to make further experimental test and simulations in order to be able to determine the best techniques. All the experimental simulations and comparison are performed first offline using Matlab codes. The simulation results compare the performances of audio features and classification algorithms based on their recognition accuracy and computational speed (complexity). The offline simulation result is discussed in chapter 6 in detail.

[1] title 'Audio based context awareness system using smartphones'

3. Mobile application is developed using the best audio feature extraction and classification techniques chosen based on the Matlab (offline) simulation result, during the Matlab experiment(step 3). The mobile application processes the audio data online and provides real-time classification results. The developed mobile application is discussed in chapter 7.

The rest of the thesis is organized as follows: In chapter 2, we present a background information in order to understand the basics and principles of digital audio signal processing. Chapter 3 and chapter 4, respectively, introduce audio features and classification methods which are used in the thesis. In chapter 5, we discuss the design procedures and steps of the thesis project in detail. The chapter discusses each steps and components of the ESR technique. Then chapter 6 provides the simulation results and analysis of the results. In this chapter, the performance of the different classifiers are first presented and compared in order to choose the best classifier. Then the performance of different audio features is computed and compared in order to reduce feature dimension and to choose the best feature set. Chapter 7 discuss the development of Android application and realization of the ESR technique on smartphone. Finally, in chapter 8, we provide our conclusions and directions for future research.

Chapter 2

Background and Principles Used

This chapter provides a brief background information and basic principles and techniques used in digital audio signal processing and analysis.

2.1 Digital Audio Analysis

The classical method of signal analysis, at spectral level, is based on classical Fourier analysis to the whole signal. However, an exact definition of Fourier transform cannot be directly applied in audio signal analysis because audio signals are time-varying signals (non-stationary) in the real world and, indeed, all their meaning is related to such time variability. Therefore, 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 extractions.

To solve these problems, audio signal is first split into a sequence of short segments, called *frames*, in such a way that each one is short enough to be considered pseudo-stationary. This process of dividing audio signal into frames is known as *Framing*. The length of each frame ranges between 10 and 50ms (in such a short time period it is assumed that the audio signal will not able to significantly change). Audio processing (e.g., Fourier transform, feature extraction, etc...) is done frame by frame basis. Usually, we multiply the frames with a smoothening functions such as Hamming window function in order to eliminate sharp corners and discontinuities before we apply Fourier transform operations on the frames. This process is called *Windowing*.

The process of frame by frame analysis is known as short-time signal analysis. In the literature there are variety of short-time analysis techniques such as Short-Time Fourier Transform (STFT), Discrete Wavelet Transform (DWT) and Wigner distribution (WD) [16]. STFT is the most popular short time analysis technique due to its computational simplicity. In this section, we present short-time Fourier transform (STFT). Special attention is reserved on criteria for choosing the analysis parameters, such as window length and type.

2.1.1 Short-Time Fourier Transform

The Short-Time Fourier Transform (STFT) is nothing more than Fourier analysis performed on slices of the time-domain signal. It performs Fast Fourier Transform (FFT) analysis on short windows in time. This is also called "sliding-window" FFT. The results of the FFT represent the contents of the audio signal in terms of time-frequency information. We analyze sound using STFT primarily because:

- It is simpler for time varying (non-stationary) signal processing and analysis.
- Enables us to represent the spectra of signals with spectral profiles that change 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

In STFT the signal to be analyzed or transformed is broken up into a series of chunks called frames, which usually overlap with each other, to reduce artifacts at the boundary. The overlapping is also useful when the sample size of the training data is relatively small. A set of training data produces more instances with a higher percentage of overlapping than the same training data with a lower percentage of overlapping. Then Fourier transform operation is then applied to successive frames. In other words, we can think of STFT as multiplying audio signal $x(n)$ by a short-time window that is centered around the time frame n. The segment of the signal contained in the window is analyzed using the Discrete Fourier Transform (DFT), which implies the evaluation

of the Time-Frequency representation at a set of discrete frequencies. Equation 2.1 provides the mathematical definition of STFT.

$$X_m(k) = \sum_{n=-\infty}^{\infty} x(n).w(m-n).e^{-j2\pi nk/N} \tag{2.1}$$

where

$$x(n) = \text{input signal at time n}$$
$$w(n) = \text{length } m \text{ window function (e.g., Hamming)}$$
$$X_m(k) = \text{DTFT of windowed data (frame) centered about time n.}$$

In practice, we need to compute the STFT on a finite set of N points. In what follows we assume that the window is $m \leq N$ samples long and N size input audio signal, so that we can use the DFT on N points, thus obtaining a sampling of the frequency axis between 0 and 2π in multiples of $2\pi/N$. The k^{th} point in the transform domain (said the k^{th} bin of the DFT) is given by

$$X_m(k) = \sum_{n=0}^{N-1} x(n).w(m-n).e^{-j2\pi nk/N} \tag{2.2}$$

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:

$$Number\,of\,frames = \lfloor\big((N-m)/P\big)\rfloor + 1 \tag{2.3}$$

where $\lfloor\,\rfloor$ is a symbol for rounding down a fraction value to the nearest integer value, known as *flooring*.

2.1.2 Commonly Used Windows

We have already seen that audio analysis methods (such as the STFT) first divide the input audio signal into smaller time segments, or frames. Audio classification algorithms are then applied separately to each frame. The classification result of each frame is then combined to give an activity profile along the entire signal.

In the literature, three different framing techniques have been used for audio analysis: sliding windows, event-defined windows and activity-defined windows [17]. With the sliding window method, the signal is divided into windows of fixed length with no inter-window gaps. A range of window sizes have been used in previous studies from 0.25 s [18] to 6.7 s [19], with some studies including a degree of overlap between adjacent windows [19, 20]. The sliding window approach does not require pre-processing of the sensor signal and is therefore ideally suited to real-time applications. Due to its implementational simplicity, most audio analysis and classification studies have employed this approach. Thus, we use sliding window in our implementation for dividing or splitting the input audio signal into smaller time segments, or frames. Then the frame is multiplied (filtered) with window functions such as Hanning or Hamming functions in order to eliminate boundary discontinuities.

The two commonly used windows are rectangular window and Hamming window:

- **The rectangular window**- Rectangular window is the simplest analysis window. In fact, the framing process using a rectangular sliding window results in already rectangularly windowed signal. Therefore, further windowing process is not required in the case of windowing audio signal using rectangular window. The rectangular window is mathematically defined as

$$w_R(n) = \begin{cases} 1 & n = 1, ..., m-1 \\ 0 & elsewhere \end{cases} \tag{2.4}$$

  where m is the window size (in terms of number of samples)

  Figure 2.1 provides the shape of the rectangular window.

- **Hamming window** - Widowing using Hamming window is performed by simply multiplying the framed signals with the Hamming window. Usually, the framed signal and the Hamming window used has equal size. The Hamming window is mathematically defined as

$$w_H(n) = \big(0.54 - 0.46cos\big(\frac{2\pi(n-1)}{m-1}\big)\big),\ 1 \leq n \leq m \tag{2.5}$$

  Figure 2.2 shows the shape of the Hamming window.

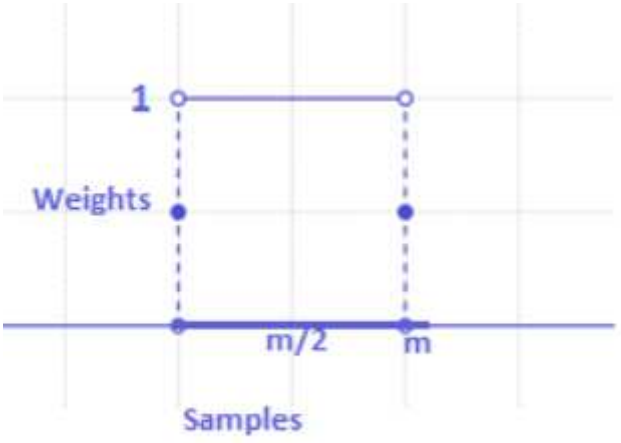


FIGURE 2.1: Rectangular window function

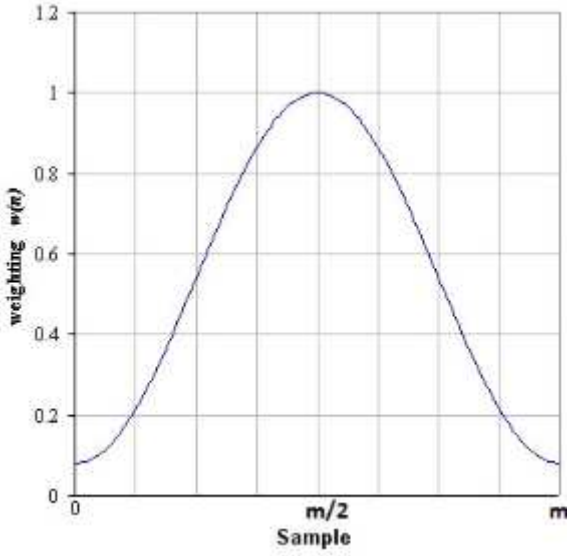


FIGURE 2.2: Hamming window function

2.1.3 Selection of windowing parameters

There are three main windowing parameters that can affect the result of the STFT: window type (shape), window size and ovelapping size (hop size). Next, we examine the effect of each of the parameters on the STFT.

- **Window type**- The rectangular window (i.e., no windowing) can cause problems when we do Fourier analysis; it abruptly cuts of the signal at its boundaries thus potentially inducing erroneous estimations of frequency components. A good

window function has a narrow main lobe and low side lobe levels in their transfer functions, which shrinks the values of the signal toward zero at the window boundaries, avoiding discontinuities. As a result, Hamming window is preferably used for windowing purposes over rectangular window.

- **Window size**- We have discussed different motivations for splitting audio into segments for processing. However, we did not consider how big those segments, frames, or analysis windows should be. There are two main important factors that has to be considered for determining the window size; i.e, *signal stationarity* and *time-frequency resolution*.

  Signal stationarity- Fast Fourier transform (FFT) operation assumes that the frequency components of the signal are unchanging (i.e, stationary– in fact, pseudo-stationary) across the analysis window of interest. Any deviation from this assumption would result in inaccurate determination of the frequency components. This point reveals that the importance of ensuring that the analysis window leading to FFT is sized so that the signal is stationary across the period of analysis. In practice, many audio signals do not tend to remain stationary for so long, and thus smaller analysis window are necessary to capture the rapidly changing details. Many literature assume audio signal to be stationary (pseudo-stationary) over a period of about 20 – 50 ms.

  Time-frequency resolution- Moving back to FFT, the output frequency vector, from an N-sample FFT of audio signal sampled at F_s Hz, contains $\frac{N}{2}+1$ positive frequency bins. Each bin collects the energy from a small range of frequencies in the original signal. The bin width is related to both the sampling rate and to the number of signals being analysed, $\frac{F_s}{N}$. Put another way, this bin width is equal to the reciprocal of the time span encompassed by the analysis window. It, therefore, makes sense that in order to achieve a higher frequency resolution, we need to collect a longer duration of samples. However, for rapidly changing signals, collecting more of them means we might end up missing some time domain features as we have discussed above.

  So the window length (size) is chosen according to the trade-off between higher frequency/spectral resolution (more samples) and time/temporal resolution (less samples) governed by the uncertainty principle. Smaller window width results in better time resolution and poor frequency resolution and vice versa. The STFT

analysis is based on the assumption that, within one frame, the signal is stationary. The shorter the window, the more true the assumption is. However, short windows result in low spectral/frequency resolution. Thus, the choice of analysis window size depends on the requirement of the problem. (Discrete) Wavelet Transform (DWT) and Wigner distribution (WD) [16] are used as alternatives to STFT in order to satisfy the demand of both high frequency and time resolutions. However, these methods are more computationally intensive compared to STFT. The main limitation of STFT is that it has a fixed time-frequency resolution due to the fixed window size used.

- **Window overlapping size**

  Overlapping ensures that audio features occurring at a discontinuity are at least considered whole in the subsequent overlapped frame. The degree of overlap (usually expressed as percentage) describes the amount of previous frame that is repeated in the following frame. Overlap of 25% and 50% are common. Similar to the window size, the determination of the overlap size depends very much on the purposes of the analysis or application. In general, more overlap will give more analysis points and therefore smoother results across time which can possibly lead to better recognition accuracy, but the computational expense is proportionately greater.

Chapter 3

Audio Features

Audio features can be broadly classified based on their semantic interpretation as perceptual and physical features. Perceptual features approximate properties that are perceived by human listeners such as pitch, loudness, rhythm, and timbre. In contrast, physical features describe audio signals in terms of mathematical, statistical, and physical properties. Based on the domain of representation, physical features are further divided as temporal features and spectral features. In this chapter, we introduce various physical features that are used during the implementation of this project. These audio features have been selected as the most appropriate features for ESR applications based on the literature survey that has been performed during the research topic study [1]. However, before we present the audio features, it is important to, first, look into the criteria we used in order to select the audio features.

3.1 Requirements for Audio Features Selection

In the literature, there are a number of audio feature extraction techniques. The recognition accuracy and performance of the ESR is highly affected by the type of audio feature extraction techniques that are used. As a result, a wise selection of audio features and classifiers is important in order to obtain a good (acceptable) performance and recognition accuracy. The type of audio feature one selects depends mainly on the type and purpose of the application one wants to use. The assumption is that the audio based CA

[1] title 'Audio based context awareness system using smartphones'

technique (ESR) will be implemented on a resource constrained devices such as smart mobile phones. The requirements for implementation of such applications include low computational complexity and power consumption. However, these requirements often affect the recognition accuracy of the ESR as well. Therefore, the selection of the audio features should be done with the main goal of developing a CA technique which has low computational complexity, energy consumption, memory requirement, and yet provides acceptable recognition accuracy. The following are some of the parameters that have been used, whenever possible, to select the audio features.

- **Small feature size**- Large feature size leads to high computational cost and curse of dimensionality . Thus, it is important to reduce the number of features, for example, by avoiding redundancies in the feature space. On the other hand, using smaller feature size may result in reduced classification accuracy. Thus, it is important to select an audio feature with optimum feature size that can provide an acceptable level of accuracy and performance as well as reduced computational cost.

- **Low computational complexity**- The computational complexity of an audio feature refers to the amount of computation time required to produce the audio feature. Audio features that require lower computation time are preferred to audio features that require higher computation time.

- **High inter-class variability**-Achieving increased discrimination among different classes of audio patterns is crucial for increased recognition accuracy. Inter-class variability refers to discrimination power of audio feature across different classes. Therefore, good audio features should show high inter-class variability.

- **High intra-class similarity**- Decreased discrimination among similar classes or sound events belonging to same class is crucial for increased recognition accuracy. Consequently, audio features extracted from sound events or environmental sounds belonging to a similar class should have similar behavior or should not show significant deviation among each other.

- **Low sensitivity**- An indicator for the robustness of a feature is the sensitivity to minor changes in the underlying signal. Usually, low sensitivity is desired in order to remain robust against noise and other sources of irritation.

3.2 Audio Physical Features

Unlike the perceptual features which can only perceived by human being, physical features refers to physical quantities that can be measured or computed using mathematical formulations. In some literatures, physical features are further divide into three groups as temporal, spectral and cepstral features. However, most of the literatures categorize physical audio features into temporal and spectral features. In the later case, there is no distiniction between the cepstral domain features and the spectral domain features. They are considered as the same domain with common group name, spectral domain features. We adopt the later grouping method for simplicity purposes (temporal and spectral features).

3.2.1 Temporal Features

The temporal domain is the native representation domain for audio signals. All temporal features are extracted directly from the raw audio signal, without any preceding transformation. Consequently, the computational complexity of temporal features tends to be low compared with that of the spectral features.

Temporal features of audio signal includes:

- **Zero crossing rate (ZCR)**- ZCR is the most common type of zero crossing based audio features [21]. It is defined as the number of time-domain zero crossings within a processing frame. It indicates the frequency of signal amplitude sign change. ZCR allow for a rough estimation of dominant frequency and spectral centroid [22]. We used the following equation to compute the average zero-crossing rate.

$$ZCR = \frac{1}{2N}\Big(\sum_{n=1}^{N} |sgn\big(x(n)\big) - sgn\big(x(n-1)\big)|\Big) \tag{3.1}$$

  where x is the time-domain signal, sgn is the signum function, and N is the size of processing frame. The signum function implementation can be defined as

$$sgn\big(x\big) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{3.2}$$

One of the most attractive properties of the ZCR is that it is very fast to calculate. As being a time-domain feature, there is no need to calculate the spectra. Furthermore, a system which uses only the ZCR-based features would not even need digital-to-analog conversion, but only the information whenever the sign of the signal changes. However, ZCR can be sensitive to noise. Though using a threshold value (level) near to zero can significantly reduce the sensitivity to noise, determining appropriate threshold level is not easy.

- **Short-time energy (STE)** – The short-time energy [23] is one of energy based audio features. Li [24] and Zhang [25] used it to classify audio signals. It is easy to calculate and provides a convenient representation of the amplitude variation over time. It indicates the loudness of an audio signal. STE is a reliable indicator for silence detection. It is defined to be the sum of a squared time domain sequences of audio data, as shown in equation 3.3.

$$STE = \frac{1}{N}\sum_{n=1}^{N}\big(x(n)\big)^2 \tag{3.3}$$

where $x(n)$ is the value of the sample (in time domain) and N is the total number of samples in the processing window (frame size). The STE of audio signal may be affected by the gain value of the recording devices. Usually we normalize the value of STE to reduce the effect.

ZCR and STE are widely used in speech and music recognition applications [26]. Speech, for example, has a high variance in ZCR and STE values, while in music these values are normally much more constant. ZCR and STE have been also used in ESR applications [27] due to their simplicity and low computational complexity.

- **Temporal centroid (TC)** -TC is the time average over the envelope of a signal in seconds [28]. It is the point in time where most of the energy of the signal is located in average.

$$TC = \frac{\sum_{n=1}^{N} n.|x(n)|^2}{\sum_{n=1}^{N} |x(n)|^2} \tag{3.4}$$

Note that the computation of temporal centroid is equivalent to that of spectral centroid (see subsection 3.2.2) in the frequency domain.

- **Energy entropy (EE)**- The short-term entropy of energy can be interpreted as a measure of abrupt changes in the energy level of an audio signal. In order to compute it, we first divide each short-term frame in K sub-frames of fixed duration. Then for each sub-frame, j, we compute its energy as in Eq. (3.3). and divide it by the total energy, $E_{shortFrame_i}$, of the short-term frame. The following equations presents the procedure to compute the energy entropy of a frame (short-term frame).

$$e_j = \frac{E_{subFrame_j}}{E_{shortFrame_i}} \tag{3.5}$$

where

$$E_{shortFrame_i} = \sum_{k=1}^{K} E_{subFrame_k}. \tag{3.6}$$

At a final step, the entropy, $H(i)$, of the sequence e_j is computed according to the equation:

$$H(i) = -\sum_{j=1}^{K} e_j.log_2(e_j). \tag{3.7}$$

- **Autocorrelation (AC)**-The autocorrelation domain represents the correlation of a signal with a time-shifted version of the same signal for different time lags [21]. It reveals repeating patterns and their periodicities in a signal and can be employed, for example, for the estimation of the fundamental frequency of a signal. This allows distinguishing between sounds that have harmonic spectrum and non-harmonic spectrum, e.g., between musical sounds and noise. Autocorrelation of a signal is calculated as follows:

$$AC = f_{xx}[\tau] = x[\tau] * x[-\tau] = \sum_{n=0}^{N-1} x\big(n\big).x\big(n+\tau\big) \tag{3.8}$$

where τ is the lag (discrete delay index), $f_{xx}[\tau]$ is the corresponding autocorrelation value, N is the length of the frame n the sample index, and when $\tau = 0$, $f_{xx}[\tau]$ becomes the signal's power. Similar to the way RMS is computed, autocorrelation also steps through windowed portions of a signal where each windowed frame's samples are multiplied with each other and then summed according to the above equation. This is repeated where one frame is kept constant while the other $x\big(n+\tau\big)$ is updated by shifting the input $x\big(n\big)$ via τ.

- **Root mean square (RMS)**- As STE, the RMS value is a measurment of energy in a signal. The RMS value is however defined to be the square root of the avaerage of a squared signal, as seen in equation 3.9.

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\big(x(n)\big)^2} \tag{3.9}$$

3.2.2 Audio Spectral Features

The group of frequency domain features is the largest group of audio features. The frequency domain reveals the spectral distribution of a signal. For each frequency (or frequency band/bin) the domain provides the corresponding magnitude and phase. Since phase variation has little effect on the sound we hear, features that evaluate the phase information are usually ignored. Consequently, we focus on features that capture basic properties of the spectral properties of audio signal: subband energy ratio, spectral flux, spectral centroid, spectral entropy, spectral roll-off, and Mel-frequency cepstral coefficients (MFCCs).

Popular transformations from time to frequency domain are Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), and Discrete Wavelet Transform (DWT). Another widely-used way to transform a signal from temporal to frequency domain is the application of banks of band-pass filters with e.g. Mel and Bark-scaled filters to the time domain signal. However, discrete Fourier transform is widely used for its simpler computational complexities. Next we introduce spectral audio features that are used in our environmental audio based context recognition application.

- **Spectral centroid(SC)**- Spectral centroid [21] represents the "balancing point", or the midpoint of the spectral power distribution of a signal. It is related to the brightness of a sound. The higher the centroid, the brighter (high frequency) the sound is. A spectral centroid provides a noise-robust estimate of how the dominant frequency of a signal changes over time. As such, spectral centroids are an increasingly popular tool in several signal processing applications, such as speech processing. Spectral centroid is obtained by evaluating the "center of gravity" using the Fourier transform's frequency and magnitude information. The individual centroid of a spectral frame is defined as the average frequency weighted

by amplitudes, divided by the sum of the amplitudes. The following equation shows how to compute the spectral centroid, SC_i, of the i^{th} audio frame.

$$SC_i = \frac{\sum_{k=0}^{K-1} k.|X_i(k)|^2}{\sum_{k=0}^{K-1} |X_i(k)|^2} \tag{3.10}$$

Here, $X_i(k)$ is the amplitude corresponding to bin k (in DFT spectrum of the signal) of the i^{th} audio frame and K is the size of the frame. The result of the spectral centroid is a bin index within the range $0 < SC < K - 1$. It can be converted either to Hz (using equation 3.11) or to a parameter range between zero and one by dividing it by the frame size, K. The frequency of bin index k can be computed from the block (frame) length K and sample rate f_s by:

$$f(k) = (\frac{f_s}{K})k \tag{3.11}$$

Low results indicate significant low frequency components and insignificant high frequency components (low brightness) and vice versa.

- **Spectral spread (SS)**- The spectral spread is the second cental moment of the spectrum. It is a measure that signifies if the power spectrum is concentrated around the centroid or spread out over the spectrum. In order to compute it, one has to take the deviation of the spectrum from the spectral centroid, according to the following equation:

$$SC_i = \sqrt{\frac{\sum_{k=0}^{K-1}(k - SC_i)^2.|X_i(k)|^2}{\sum_{k=0}^{K-1} |X_i(k)|^2}} \tag{3.12}$$

- **Spectral rolloff point (SRP)** - The spectral rolloff point is the N% percentile of the power spectral distribution, where N is usually 85% or 95% [29]. The spectral rolloff point is the frequency below which N% of the magnitude distribution is concentrated. It increases with the bandwidth of a signal. Spectral rolloff is extensively used in music information retrieval [30] and speech/music segmentation. The spectral rolloff point is calculated as follows:

$$SRP = f(N) \ where \ f(N) = (\frac{f_s}{K})N \tag{3.13}$$

where N is the largest bin that fulfills equation 3.14.

$$\sum_{k=0}^{N} |X(k)|^2 \leq TH. \sum_{k=0}^{K-1} |X(k)|^2 \tag{3.14}$$

where *X(k)* are the magnitude components, k frequency index and *f(K)* (the frequency) spectral roll-off point with $\big(100 * TH\big)\%$ of the energy. TH is a threshold between 0 and 1. A commonly used value for the threshold is 0.85 and 0.95 [29, 31]. This measure is useful in distinguishing voiced speech from unvoiced: unvoiced speech has a high proportion of energy contained in the high-frequency range of the spectrum, whereas most of the energy for voiced speech and music is contained in lower bands [32].

- **Spectral flux (SF)**- The SF is a 2-norm of the frame-to-frame spectral amplitude difference vector. It defines the amount of frame-to-frame fluctuation in time. i.e., it measures the change in the shape of the power spectrum. It is computed via the energy difference between consecutive frames as follows:

$$SF_f = \sum_{k=0}^{K-1} ||X_f\big(k\big)| - |X_{f-1}\big(k\big)|| \tag{3.15}$$

where f is the index of the frame and K is the frame length. Spectral flux is an efficient feature for speech/music discrimination, since in speech the frame-to frame spectra fluctuate more than in music, particularly in unvoiced speech [33].

- **Spectral entropy (SE)**- Spectral entropy [34] is computed in a similar manner to the entropy of energy, although, this time, the computation takes place in the frequency domain. More specifically, we first divide the spectrum of the short-term frame into L sub-bands (bins). The energy E_f of the f^th sub-band, $f = 0, ..., L-1$, is then normalized by the total spectral energy, thst is, $n_f = \frac{E_f}{\sum_{f=0}^{L-1} E_f}, f = 0, ..., L-1$. The entropy of the normalized spectral energy n_f is finally computed according to the equation:

$$H = -\sum_{f=0}^{L-1} n_f.log_2(n_f) \tag{3.16}$$

- **Mel-frequency cepstral coefficients (MFCCs)**- MFCC originate from automatic speech recognition but evolved into one of the standard techniques in most

domains of audio recognition applications such environmental sound classifications [27, 51, 52]. They represent timbral information (spectral envelop) of a signal. Computation of MFCC includes conversion of the Fourier coefficients to Mel-scale. After conversion, the obtained vectors are logarthmized, and decorrelated by discrete cosine transform (DCT) in order to remove redundant information. Figure 3.1 shows the process of MFCC feature extraction.

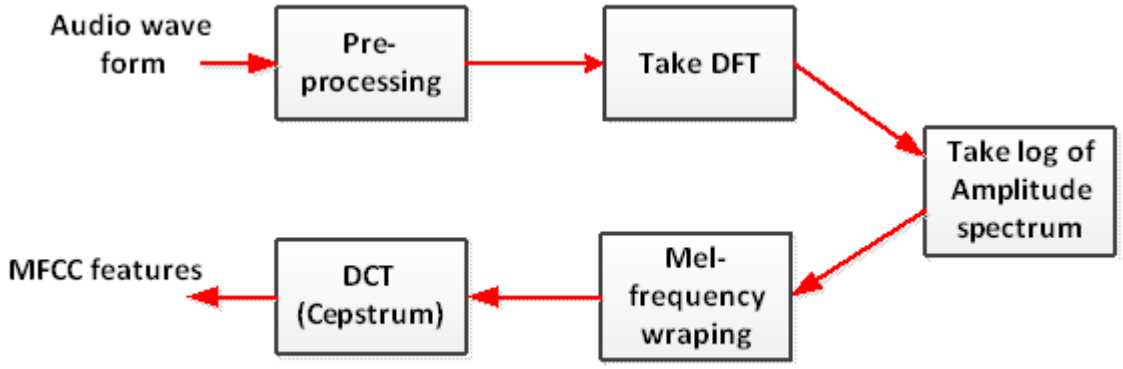


FIGURE 3.1: MFCC extraction process

In figure 3.1, the first step, preprocessing, consists of pre-emphasizing, frame blocking and windowing of the signal. The aim of this step is to model small (typically, 20ms) sections of the signal (frame) that are statistically stationary. The window function, typically a Hamming window, removes edge effects. The next step takes the Discrete Fourier transform (DFT) of each frame. We retain only the logarithm of the amplitude spectrum. We discard phase information because perceptual studies have shown that the amplitude of the spectrum is much more important than the phase. We take the logarithm of the amplitude because the perceived loudness of a signal has been found to be approximately logarithmic. After a discrete Fourier transform, the power spectrum is transformed to Mel-frequency scale. This step smooths the spectrum and emphasizes perceptually meaningful frequencies. Mel- frequency scale is based on mapping between actual frequency and perceived pitch by human auditory system. The mapping is approximately linear below 1 KHz and logarithmic above. This is done by using a filter bank consisting of triangular filters, spaced uniformly on the Mel-scale. An approximate

conversion between a frequency value in Hertz (f) and in mel is given by:

$$mel(f) = 2595\, log_{10}(1 + \frac{f}{700}) \tag{3.17}$$

Finally, the cepstral coefficients are calculated from the mel-spectrum by taking the discrete cosine transform (DCT) of the logarithm of the mel-spectrum. This calculation is given in by:

$$c_i = \sum_{k=0}^{K-1} (logS_k).cos(\frac{i\pi}{K}(k - \frac{1}{2})) \tag{3.18}$$

where c_i is the i^{th} MFCC, S_k is the output of k^{th} filter bank channel (i.e. the weighted sum of the power spectrum bins on that channel) and K is the number of coefficients (number of Mel-filter banks). The value of K used is mostly between 20 to 40. In this project we used the value of K to be equal to 23.

The components of MFCCs are the first few DCT coefficients that describe the coarse spectral shape. The first DCT coefficient represents the average power (energy) in the spectrum. The second coefficient approximates the broad shape of the spectrum and is related to the spectral centroid. The higher order coefficients represent finer spectral details (e.g., pitch). In practice, the first 8-13 MFCC coefficients are used to represent the shape of the spectrum. The higher order coefficients are ignored since they provide more redundant information. However, some applications require more higher-order coefficients to capture pitch and tone information.

Chapter 4

Audio Classifiers

Based on their learning behavior, classifiers can be divided into two groups: classifiers that use supervised learning (supervised classification) and unsupervised learning (unsupervised classification). In supervised classification, we provide examples of the correct classification (a feature vector along with its correct class) to teach the classifier. Based on these examples, which are commonly termed as training samples, the classifier then learns how to assign an unseen feature vector to a correct class. Examples of supervised classifications include Hidden Markov Model (HMM)[35], Gaussian Mixture Models (GMM)[36], K- Nearest Neighbor (k-NN)[35], Support Vector Machine (SVM)[37], Artificial Neural Networks (ANN), Bayesian Network (BN)[35], and Dynamic Time Wrapping (DTW)[38]. In unsupervised classification or clustering, there is neither explicit teacher nor training samples. The classification of the feature vectors must be based on similarity between them based on which they are divided into natural groupings. Whether any two feature vectors are similar depends on the application. Obviously, unsupervised classification is a more difficult problem than supervised classification and supervised classification is the preferable option if it is possible. In some cases, however, it is necessary to use unsupervised learning. For example, this is the case if the feature vector describing an object can be expected to change with time. Examples of unsupervised classifications include k-means clustering, Self-Organizing Maps (SOM), and Linear vector Quantization (LVQ).

Classifiers can also be grouped based on reasoning process as probabilistic and deterministic classifiers. Deterministic reasoning classifiers classify sensed data into distinct

states and produce a distinct output that cannot be uncertain or disputable. Probabilistic reasoning, on the other hand, considers sensed data to be uncertain input and thus outputs multiple contextual states with associated degrees of truthfulness or probabilities. Decision of the class type to which the feature belongs is made based on the highest probability.

4.1 Requirements for Audio Classifier Selection

The two main criteria that can be used to select classification techniques are computational complexity and recognition accuracy. Moreover, robustness to noise can be used as a criteria in some applications; especially, in a noise prone application.

- **Computational complexity**- The computational complexity of a classifier can be measured by the amount of computational time it requires to produce the classification result. Computational complexity of an algorithm can also provide insight about its power consumption. It is preferred to use classifiers with low computational complexity, which can provide classification result faster and as a result consume less power.
- **Recognition accuracy**- The recognition accuracy of a classifier can be affected by a number of factors. Selection of audio feature is the most important factor that affects the recognition accuracy. In addition, selection of a good type of classifier improves the recognition accuracy.
- **Robustness to noise**- Any good classifier should be able to ignore any feature variations caused by disturbances such as noise, bandwidth or the amplitude scaling of an audio signal.

Similar to the case of audio feature selection, the selection process of audio classifiers usually requires a trade-off between computational complexity and recognition accuracy. In the literature, there are many different audio classifiers. However, there is no sufficient previous work that compares the performance of the audio classifiers based on the above requirements. Thus, we select some classifiers based on their popularity and then compare their performance so that we can determine the best one. *k-NN*, SVM and

GMM are chosen due to their common use in a number of ESR applications/problems for discussion (in this chapter) and further performance comparison (in chapter 5).

4.2 Popular classifiers

We start description of selected classifiers with the famous *k-nearest neighbor classifier (k-NN classifier)* and proceed with the *gaussian mixture model (GMM)* and the more sophisticated *support vector machines (SVMs)*. Obviously, this is just a very small subset of the classifiers that have been proposed and studied in the literature but serve well our purpose to focus on selected methods which are both popular and representative of the wealth of techniques that are available. Lengthy theoretical descriptions of the classifiers have been avoided and instead an attempt to highlight the key ideas behind the algorithms being studied is made. These set of classifiers have been selected for experimental and simulation test in our implementation due to their popular use in the literature. We look into their applicability for mobile devices (smartphones) in chapter 6 taking into account the limited resources (like energy, cpu, memory) of the mobile device. Chapter 6 provides the performance comparison of the classifiers based on their classification accuracy and computational speed.

4.2.1 The k-Nearest Neighbor Classifier (k-NN)

Despite its simplicity, the k-NN classifier is well tailored for both binary and multi-class problems. Its outstanding characteristics is that it does not require a training stage in the strict sense. The training samples are rather used directly by the classifier during the classification stage. The key idea behind this classifier is that, if we are given a set of patterns (unknown feature vector), X, we first detect its k-nearest neighbors in the training set and count how many of those belong to each class. In the end the feature vector is assigned to the class which has the highest number of neighbors. Therefore, for the k-NN algorithm to operate the following ingredients are required:

1. A data set of labeled samples, that is a training set of feature vectors and respective class labels.

2. An integer $k \geq 1$.

3. A distance (dissimilarity) measure.

Let us now go through the k-NN algorithm in more detail. In the first step, the algorithm computes the distance, $d(X, V_i)$, between X and each vector $V_i, i = 1, ..., M$, of the training set, where M is the total number of training samples. The most common choice of distance measure is the *Euclidian sistance*, which is computed as:

$$d(X, V_i) = \sqrt{\sum_{j=1}^{D} (X(j) - V_i(j))^2} \tag{4.1}$$

where D is the dimentionality of the feature space. Another popular choice of distance measure is known as the Mahalanobis distance [39]. After $d(X, V_i)$ has been computed for each V_i, the resulting distance values are sorted in ascending order. As a result, the k first values correspond to the k closest neighbors of the unknown feature vector. Now, let k_i be the number of the training vectors among the k neighbors of X that belong to the i^{th} class, $i = 1, ..., N_c$. The unknown vector is then classified to the class which corresponds to the maximum k_i. In the example in Figure 4.1 , we have three classes and the goal is to find a class label for the unknown example X . The Euclidean distance is used and k=5. Of the 5 closest neighbors, 4 belong to *W1* and 1 belongs to *W3* , so X is assigned to *W1*, the predominant class.

It is important to note that the *k-NN* classifier can operate directely in a *multi-class* environment. This is because its algorithmic steps are not restricted by the number of classes that are involved.

The performance of the *k-NN* has been extensively studied in the literature. An interesting theoretical finding is that if $k \to \infty$, $M \to \infty$, and $\frac{k}{M} \to 0$, then the classification error approaches the Bayesian error [39]. In other words, if both k and M approach infinity and k is infinitely smaller than M, then the *k-NN* classifier tends to behave like the optimal (Bayesian) classifier eith respect to the classification error.

The larger the dataset, the more satisfactory of the performance of the *k-NN* algorithm. Concerning parameter k (the number of neighbors), it can be stated that the value of k is tuned after experimentation with the dataset at hand . In general, small values are prefered by also taking into account the size of the dataset (so that $\frac{k}{M}$ remains as small as possible).

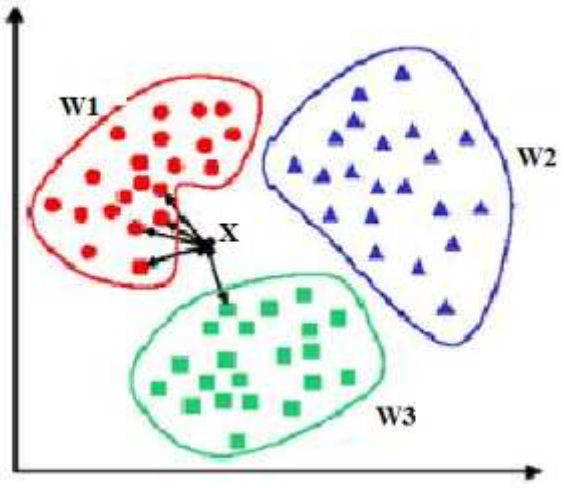


FIGURE 4.1: k-NN Classification Example

Another important issue is the computational complexity of the *k-NN* classifier which can be prohibitively high when the volume of the dataset is really large (mainly due to the number of *Euclidian distances* that need to be computed). Over the years, several remedies to this computational issue have been proposed in the literature, e.g. [40, 41].

4.2.2 Gaussian Mixture Model (GMM)

The Gaussian Mixture Model (GMM) [42] is used in classifying different audio classes. It is an example of a parametric classifier. It is an intuitive approach when the model consists of several Gaussian components, which can be seen to model acoustic features. In classification, each class is represented by a GMM and refers to its model. Once the GMM is trained, it can be used to predict which class a new sample probably belongs to [43].

The probability distribution of feature vectors is modeled by parametric or non-parametric methods. Models which assume the shape of probability density function are termed parametric. In non-parametric modeling, minimal or no assumptions are made regarding the probability distribution of feature vectors. The potential of Gaussian mixture models to represent an underlying set of acoustic classes by individual Gaussian components, in which the spectral shape of the acoustic class is parameterized by the mean

vector and the covariance matrix, is significant, especially if the dataset (feature data points) is large.

Moreover, these models have the ability to form a smooth approximation to the arbitrarily-shaped observation densities in the absence of other information [44] . With Gaussian mixture models, each sound is modeled as a mixture of several Gaussian clusters in the feature space. The basis for using GMM is that the distribution of feature vectors extracted from a class can be modeled by a mixture of Gaussian densities as shown in Fig. 4.2. For example, the figure shows that the dataset is modeled as a linear combinations of three different Gaussian distributions.

Mathematically, the GMM is given as a weighted sum of K component Gaussian densities, as in the following equation:

$$p(x|\lambda) = \sum_{i=0}^{K} w_i g(x|\mu_i, \xi_i), \tag{4.2}$$

where x is a D-dimensional feature vector, $w_i, i = 0, ..., K$, are the mixture weights, and $g(x|\mu_i, \xi_i), i = 0, ..., K$, are the component Gaussian densities. Each component density is a D-variate Gaussian function of the form,

$$g(x|\mu_i, \xi_i) = \frac{1}{(2\pi)^{D/2}|\xi_i|^{1/2}} exp\{ -\frac{1}{2}(x - \mu_i)\xi^{-1}(x - \mu_i)\} \tag{4.3}$$

with mean vector μ_i and covariance matrix ξ_i. The mixture weights satisfy the constraint that $\sum_{i=1}^{K} w_i = 1$.

The complete Gaussian mixture model is parameterized by the mean vectors, covariance matrices and mixture weights from all components densities. These parameters are collectively represented by the notation,

$$\lambda = \{w_i, \mu_i, \xi_i\} \; i = 1, ..., K. \tag{4.4}$$

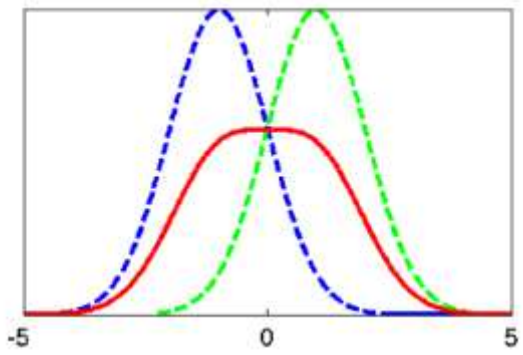


FIGURE 4.2: Gaussian Mixture Models

The goal of the training is to calculate the parameters of GMM from previously estimated model using training feature vectors. Various methods can be applied for estimation but the most popular and robust method is maximum likelihood (ML) estimation [45]. Aim of the ML is to find the best model parameters of the GMM that refine the previously calculated GMM likelihood

The motivation for using Gaussian densities as the representation of audio features is the potential of GMMs to represent an underlying set of acoustic classes by individual Gaussian components in which the spectral shape of the acoustic class is parameterized by the mean vector and the covariance matrix. Also, GMMs have the ability to form a smooth approximation to the arbitrarily shaped observation densities in the absence of other information. With GMMs, each sound is modeled as a mixture of several Gaussian clusters in the feature space.

GMMs model the distribution of feature vectors. For each class, assume the existence of a probability density function expressible as a mixture of a number of multidimensional Gaussian distributions. The iterative Expectation Maximization (EM) algorithm is usually used to estimate the parameters for each Gaussian component and the mixture weights.

A variety of approaches to the problem of mixture decomposition have been proposed, many of which focus on maximum likelihood methods such as Expectation Maximization (EM) or Maximum A Posterior Estimation (MAP). Generally these methods consider separately the question of parameter estimation and system identification, that is to say a distinction is made between the determination of the number and functional form of components within a mixture and the estimation of the corresponding parameter values.

4.2.3 Support Vector Machine (SVM)

Support Vector Machines (SVM) are classifiers that have been successfully employed in numerous machine learning fields. It is very effective method for general purpose pattern recognition.

Given a set of points which belong to either of two classes, a SVM finds a hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of either class from the hyper plane. SVMs perform pattern recognition between two classes by finding a decision surface that has maximum distance to the closest points in the training set which are termed *support vectors*, see figure 4.3.

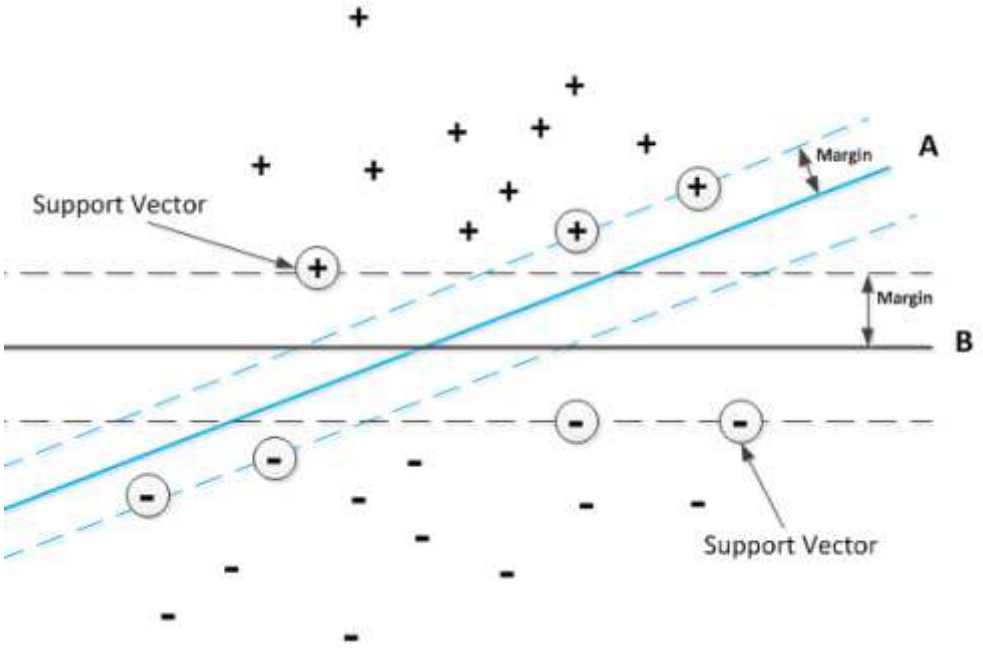


FIGURE 4.3: SVM Classification

Principle of SVM is, where there are many possible linear classifiers that can separate the data, there is only one that maximizes the difference between them. SVMs are particular classifiers that are based on the margin-maximization principle [46]. It performs an implicit mapping of data into a higher (maybe infinite) dimensional feature space and then finds a linear separating hyper plane with the maximal margin to separate data in this higher dimensional space [47]. This property makes SVM to be a powerful machine learning technique for data classification.

SVM maps the input patterns into a higher dimensional feature space through some nonlinear mapping chosen at priori. A linear decision surface is then constructed in this high dimensional feature space, as shown in Figure 4.4. Thus, SVM is a linear classifier in the parameter space, but it becomes a non-linear classifier as a result of the non-linear mapping of the space of the input patterns into the high dimensional feature space.

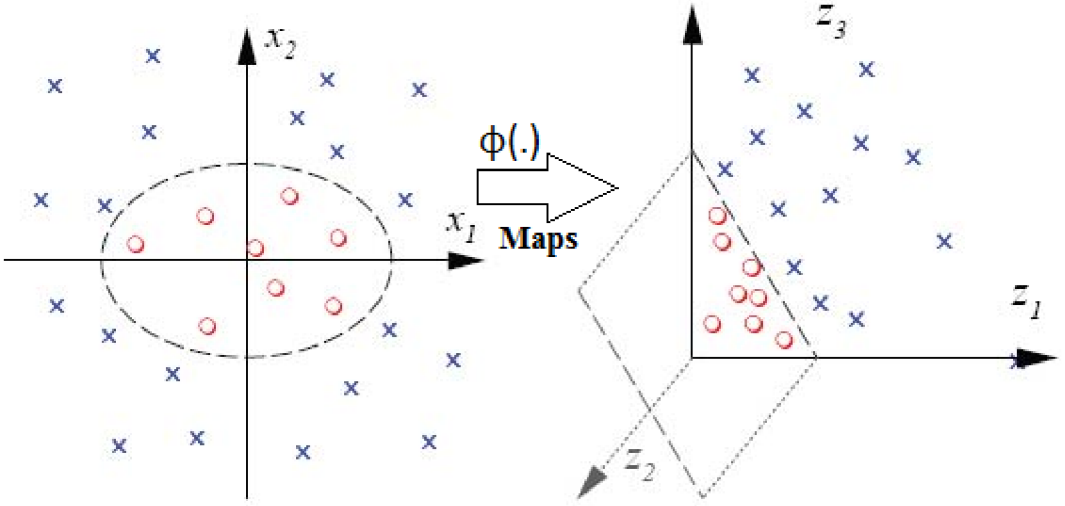


FIGURE 4.4: SVM Mapping to Higher Dimensional Space

For linearly separable data, SVM finds a separating hyper plane which separates the data with the largest margin. For linearly inseparable data, it maps the data in the input space into a high dimension space $x \in R^I \rightarrow \varphi(x) \in R^H$ with kernel function $\varphi(x)$, to find the separating hyper plane. An example for SVM kernel function $\varphi(x)$ which maps 2- Dimensional input space to higher 3-Dimensional feature space is shown in Fig. 4.4.

SVM was originally developed for two class classification problems. The N class classification problem can be solved using N SVMs. Each SVM separates a single class from all the remaining classes [48]. In other words, the multi-class problem is treated as treated as a combination of binary classification tasks, a technique also called as *binarization* [49]. Two popular binarization approaches are frequently encountered in the literature, namely the *One-vs-All* and *One-vs-One* methods.

- **One-vs-All (OVA)** - According to this method (also known as one-vs-rest–OVR), one SVM classifier is employed per class. The goal is to train the individual classifiers to discriminate between the samples of the respective classes (positives examples) and the samples of all the other classes (negative examples). In our work, we used OVA due to its simplicity and proved significant discriminative capablity in various multi-class problems [50]

- **One-vs-One (OVO)**- This is another type of classifier binarization, also know as pairwise or round robin classification [51]. It is based on transforming the initial multi-class task into a number of binary classification problems, where each binary problem involves two classes. The total number of classifier is therefore equal to the number of all possible class pairs. A simple way to combine the individual binary classification decisions to a global decision is via a voting scheme. Specifically, the decision of each pairwise classifier increases by one point the score of the class that won. In the end, the global decision is made based on the class that has accumulated the highest score.

In order to keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function selected to suit the problem. The kernel function may be any of the symmetric functions that satisfy the Mercer's conditions [52]. There are several SVM kernel functions. The most common SVM kernel functions are linear, polynomial, Quadratic, radial basis function (rbf)and multilayer perceptron kernel (mlp). Inquisitive readers can refer to [53] for more details on SVM classifier.

Chapter 5

Audio Classification and Event Detection Design Procedures

We have already briefly highlighted, in section 1.4, that the audio based CA technique is composed of three main components: audio/sound sensing/capturing, audio feature extraction and audio classification and detection. In this chapter, we explore the design procedures of each of these components in a more detail. In practice, the captured audio signal has to go through a pre-processing stage before the feature extraction process could be performed. Similarly, the output of the classifier are post-processed in order to cancel out isolated errors or merge successive segment that have yielded identical class labels. Thus, we will also look into the pre-processing and post-processing methods that have been employed during the design procedures of the ESR system. Figure 5.1 shows the block diagram of the pre-processing and feature extraction stages.

5.1 Audio Capturing

The audio signal is captured using the built-in mobile's microphone (which is, in fact, not high quality compared to high-end stand-alone audio recording devices). The first step in processing audio signal is to convert the analog representation (first air pressure,and then analog electric signals in a microphone) into a digital signal $x(n)$, where n is an index over time. Analysis of the audio spectrum shows that nearly all energy resides in

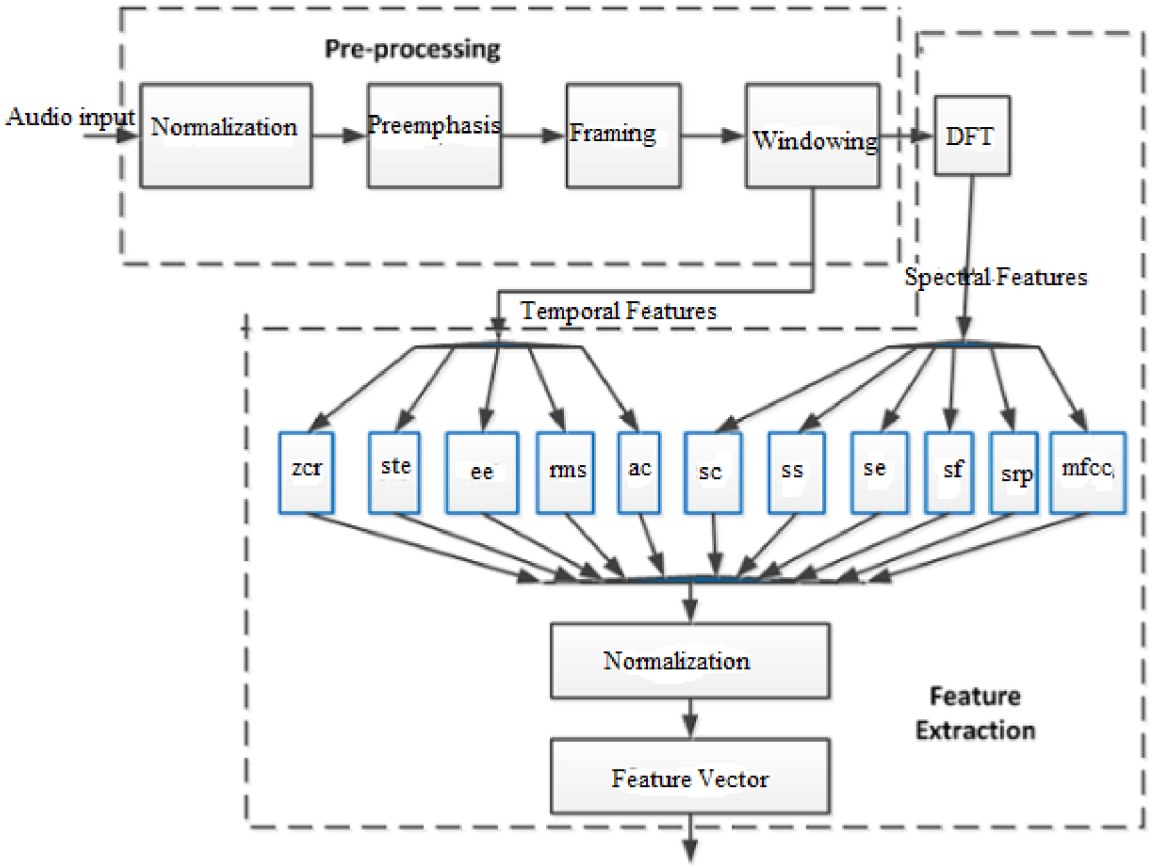


FIGURE 5.1: Pre-Processing and Feature Extraction

the lower frequency band ranges (between DC and 4 kHz, and beyond 10 kHz there is virtually no energy).

The audio signals we use for offline audio classification have the following characteristics:

- The sampling rate employed is equal to 16kHz. High sampling rate may improve recognition accuracy, but leads to more computation.
- Each sample is represented using 16-bits, signed.
- Mono-channel
- Uncompressed PCM (.wav)-raw audio
- The analysis window (mid-term window) size (duration) is set to 2 sec, (i.e., 32000 samples) with 50% overlap. The duration of frame (short-term window) is set to 50 ms with 50% overlapping (i.e., 800 samples per frame). The mid-term and short-term (frame) windows are explained in section 5.3.3.

5.2 Pre-processing

The pre-processing stage is comprised of audio input normalization, pre-emphasis, framing and windowing steps.

5.2.1 Normalization

The extracted pulse code modulated (PCM) values of amplitude is normalized, to avoid amplitude variation during capturing. This normalization step makes the technique to be robust to loudness variations. Equation 5.5 is used for normalizing the raw input audio signal.

5.2.2 Pre-emphasis

Usually sound signal is pre-emphasized before any further processing. In most cases, if we look at the spectrum of audio segment signals, there is more energy at lower frequencies than the higher frequencies. However, the human hearing system is more sensitive above the 1-kHz region of the spectrum [54]. The pre-emphasis filter amplifies this area of the spectrum. Boosting the high frequency energy makes information from these higher frequency components more available to the acoustic modeling algorithm and improves sound detection. The pre-emphasis filter is a first-order high-pass filter. In the time domain, with input *x(n)* and $0.9 \leq \alpha \leq 1.0$, the filter equation is given by:

$$y(n) = x(n) - \alpha.x(n-1) \tag{5.1}$$

We used $\alpha = 0.95$. Figure 5.2 and Figure 5.3 show a sample audio signal before and after pre-emphasis, respectively.

5.2.3 Framing

We have already seen, in chapter 2, that audio signal is a non-stationary signal, meaning that its statistical properties are not constant across time. Thus, we need to extract audio features from a small window of audio signal (frame)for which we can make the (rough) assumption that the signal is stationary. The process of splitting audio signal

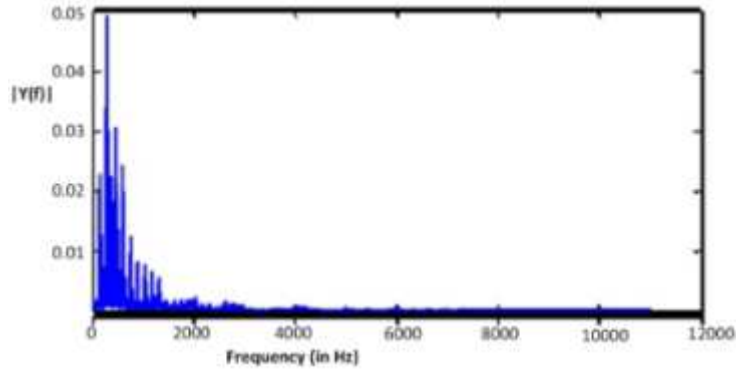


FIGURE 5.2: Signal before Pre-Emphasis

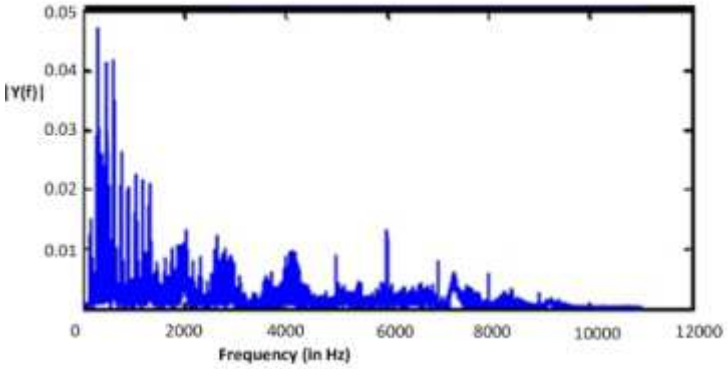


FIGURE 5.3: Signal after Pre-Emphasis

into a sequence frames is called framing. In this thesis, we used frame block size of 50 ms with 50% overlapping i.e., 800 samples per frame.

Figure 5.4 shows framing process using fixed sliding window with overlapping.

5.2.4 Windowing

Frames should be windowed before Fourier transform operations are applied. The windowing step is used to avoid the discontinuities caused by the sharp corners of the frames. In this thesis we used Hamming window function for windowing.

The windowing of audio frame takes place by multiplying the value of the frame signal at time n, $S_{frame}[n]$, with the value of the window at time n, $S_w[n]$:

$$Y[n] = S_{frame}[n] \times S_w[n] \tag{5.2}$$

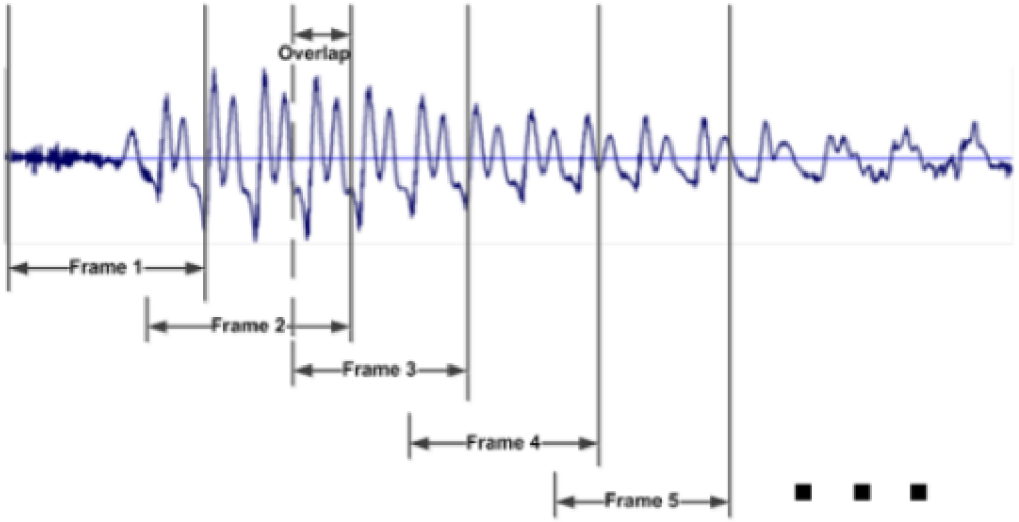


FIGURE 5.4: Framing/splitting process with fixed size sliding window

Figure 5.5 shows the windowing process of framed audio signal with hamming widow function.

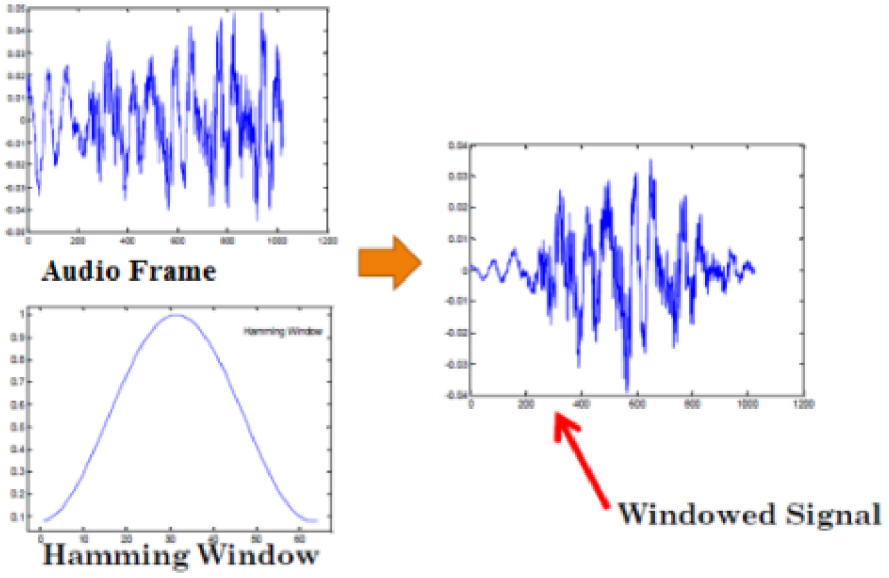


FIGURE 5.5: Windowing process using Hamming window function

In practice, the audio signal splitting (framing) and windowing processes can be performed in one step by multiplying the input audio signal with a sliding window, which can be rectangular window or Hamming window, avoiding unnecessary redundant computations.

5.3 Feature Extraction

Feature extraction is an important audio analysis stage. It is used to extract a set of features that can provide a representative characteristics of the frame. The goal is to form feature vector for each frame that can serve as an input to the classifier. Feature extraction can also be viewed as data rate reduction procedure because we want our analysis algorithm to be based on relatively small number of features, i.e., feature vectors. The audio features that have been used in our implementation are introduced in chapter 3.

Two types of features are extracted in the feature extraction stage: Temporal features and spectral features. As shown in Figure 5.1, the temporal features are computed after the windowing process while extraction of spectral features requires further step, Discrete Fourier transform (DFT) operation. We used Fast Fourier Transform or in short FFT algorithm for computing the DFT. The extracted temporal features include zero-crossing rate (ZCR), short-time energy (STE), energy entropy (EE), root-mean square (RMS) and auto-correlation (AC) and the spectral features are spectral rolloff point (SRP), spectral Spread (SS), spectral flux (SF), spectral entropy (SE), spectral centroid (SC) and mel-frequencty cepstrum coefficients (MFCCs).

5.3.1 Feature Normalization

Features with large values may have a larger influence in the cost function than features with small values (e.g. Euclidean distance). The feature normalization applied so that all feature types can contribute equally and avoid bias during the classification. Especially, the normalization step is important if the classifier uses Euclidean distance metric which is true in the case of the k-NN. Therefore, it is important to normalize the audio features before feeding to the classifiers.

Equation 5.5 has been used to normalize a features. For N available data points of the d^{th} feature ($d = 1, 2, ..., D$);

$$\textbf{Mean:} \quad m_d = \frac{1}{N}\sum_{n=1}^{N} x_{nd} \tag{5.3}$$

$$\textbf{Variance:} \quad \sigma_d = \frac{1}{N-1}\sum_{n=1}^{N}(x_{nd} - m_d)^2 \tag{5.4}$$

$$\textbf{Normalized Feature:} \quad x_{nd} = \frac{x_{nd} - m_d}{\sigma_d} \tag{5.5}$$

5.3.2 Composition of Feature Vectors

A total of 24 feature coefficients are computed for each frame: ZCR(1), STE(1), EE(1), RMS(1), AC(2), SRP(1), SS(1), SF(1), SE(1), SC(1) and MFCCs(13) (Note that the integer numbers inside the brackets indicate the number of coefficients for each feature type). That is, each frame is processed separately which results in one feature vector per frame. We call such features intra-frame features because they operate on independent frames.

We compute a single (average) value per frame for ZCR, RMS, STE, TC, SRP, SE,SC and SS where as the computation of AC produces two values per frame: *fundamental period* and *maximum auto-correlation value*. We used 23 Mel-scaled triangular filters for computing MFCCs. The first 13 MFCCs are selected because the higher order MFCCs carry less discriminative information and , therefore, are often ignored. Moreover, the very first MFCC coefficient is the sum of all the log-energies computed at the previous step - so it is an overall measure of signal loudness and is not very informative as to the actual spectral content of the signal. Thus, it is often discarded for audio recognition applications where the system has to be robust to loudness variations.

5.3.3 Short-Term and Mid-Term Processing

Short-term Feature extraction

As explained above, The audio signal is first divided into short-term frames and audio features are extracted from each of the frames. This process of extracting set of features from each frame can be called *short-term feature extraction*. i.e., the short-term feature

extraction generates a sequence, **F**, of feature vectors per audio frame. The dimensionality of the feature vector depends on the nature of the adopted features. In our case, the dimension of the (short-term) feature vector is 24. The extracted sequence of feature vectors can then be used for subsequent processing/analysis of the audio data, e.g., in mid-term feature extraction as we will see next.

Mid-Term Feature Extraction

Another common technique is the processing of the feature sequences on a mid-term basis. According to this type of processing, the sequence of feature vectors which have been extracted during the short-term processing are combined and used for computing mid-term features using statistics such as mean, standard deviation (std), maximum, minimum, etc. In other words, the function that computes mid-term features takes as input short-term feature sequences, computes the feature statistics (in our case, mean and standard deviation) and then returns a feature vector that contains the resulting feature statistics. For example, considering the 24 feature sequences that have been already computed based on a short-term basis and the two mid-term statistics that have been calculated per frame (i.e., mean value and standard deviation), the output of the mid-term extraction process is a 48-dimensional mid-term feature vector. The structure of the resulting mid-term feature vector is the following: elements 1 and 25 correspond to the mean and standard deviation of the first short-term feature sequence, elements 2 and 26 correspond to the mean and standard deviation of the second audio short-term sequence, and so on.

During mid-term processing, we assumed that the mid-term segment exhibits homogeneous behavior with respect to audio type and , therefore, it makes sense to proceed with the extraction of statistics on the mid-term basis. In practice, the duration of mid-term windows [1] typically lies in the range 1-10 s, depending on the application domain [39]. For this thesis, we used mid-term window (analysis window) duration of 2 secs with 50% overlapping.

[1] In the literature, mid term windows/segments are sometimes refered to as “texture” or “analysis” windows

5.4 Audio Classification and Detection

Initially, we implemented a simulation for three of the classification algorithms (k-NN, GMM, SVM) using MatLab and compared their performance in terms of recognition accuracy and computational speed. The purpose of this initial simulation is to compare the performance of different audio features and classifiers in order to select the best method. The MatLab simulation and comparison results are provided in chapter 6. Mobile app is then developed using feature extraction techniques and classifier algorithm with the best performance algorithms and parameters/techniques. The implementation of the mobile app is provided in chapter 7 . Figure 5.6 provides the classifier's implementation block diagram.

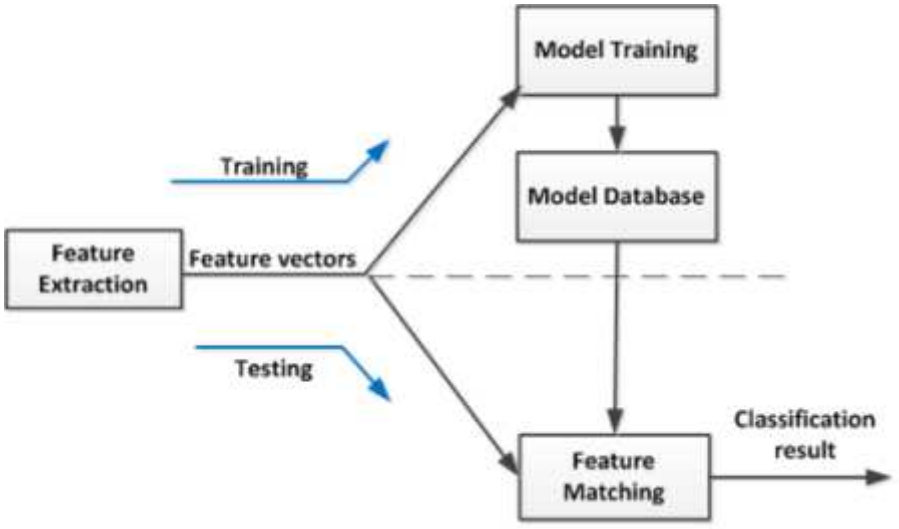


FIGURE 5.6: Classifier Implementation

During the training phase, feature vectors that have been extracted from the training data set are used for training the classifier and generating class models. The generated class models are stored in a memory and they are retrieved during the testing (classification) phase for matching with test feature vector. During the testing phase, feature vectors that have been extracted from the test data set are matched with class models and their audio type (class type) is determined.

5.5 Post-processing

As it was explained before, the audio classification process first splits the input audio stream into a fixed-sized segments and classify each segment separately to a predefined set of audio classes. The post-processing step is used merge successive segments of the same audio type and remove some classification errors. In the literature, there are two techniques of merging the individual classification results.

- *Naive merging*- the key idea is that if successive segments share the the same class label, then they can be merged together to form a single segment. Figure 5.7 demonstrates how the post-processing using naive merging is performed. The example given in the figure uses three class labels (i.e., three sound types): W, D and P which stand for sounds of "Walking", "Door opening/closing" and "Printer printing", respectively. Each subsequence whose element exhibit the same class label merge together forming a single class label.
- *Probability smoothing techniques*- if the adopted classification technique has a soft output, i.e., estimate of the classification probability instead of hard classification, then it is possible to apply more sophisticated smoothing techniques on the sequence that has been generated during the classification stage. Viterbi algorithm [55] is the most common probability smoothing technique. However, this technique can not be straightforwardly used with classification methods that produce hard classification (deterministic) outputs such as *k-NN* and SVM.

5.6 Audio Features Dimension Reduction

Before we conclude this chapter, it is important to introduce one important step that has been employed to select the optimal audio feature sets (types) from the original audio feature sets. The feature dimensionaliy reduction was used to further improve the classification accuracy and reduce the computational time of the *k-NN* classifier as further discussed in section 6.5.

In many real-world applications, numerous features are used in an attempt to ensure accurate classification. If all those features are used to build up classifiers, then they

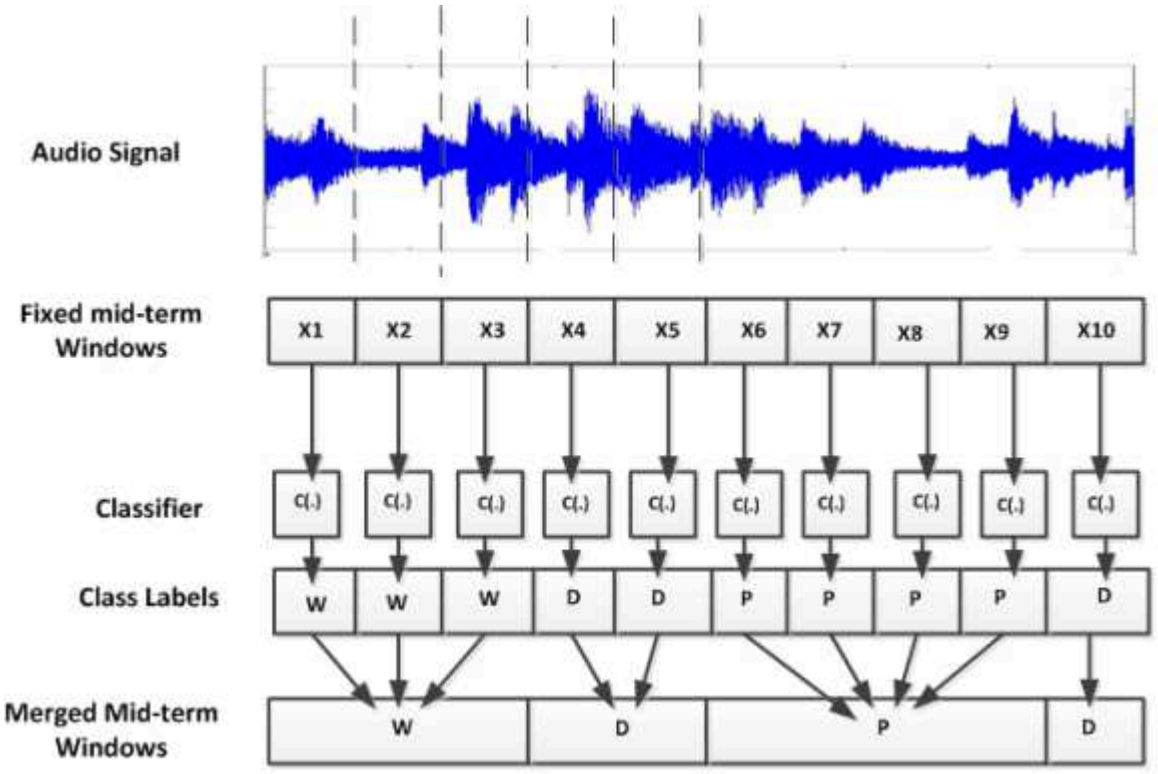


FIGURE 5.7: Post-Processing- Merging similar class labels

operate in high dimensions, and the learning process becomes computationally and analytically complicated, resulting often in the drastic rise of classification error. There are two main reasons why feature selection and dimension reductions might be an important step to be undertaken before a certain machine learning technique. The first issue is related to the so-called "curse of dimensionality" and the necessity for dimensionality reduction. The second issue is related to the potentially poor representation of the problem in terms of some irrelevant or indirectly relevant features and the corresponding necessity to improve the representation [56].

Hence, there is a need to reduce the dimensionality of the feature space before classification. According to the adopted strategy dimensionality reduction techniques are divided into *feature selection* and *feature transformation* (also called feature discovery). The key difference between feature selection and feature transformation is that during the first process a subset of original features only is selected while the second approach is based on the generation of completely new features [57]. Examples of feature transformation includes principal component analysis (PCA) and Linear discriminant analysis

(LDA). Compared to feature selection, feature transformation are more computationally complex. Therefore, many researchers used the feature selection techniques.

For example, Parkka, et al. [58] used simple visual and statistical analysis to assess the distribution of a given feature for different activities. Features which changed markedly between activities and showed little overlap were selected for subsequent analysis. In their study of six daily activities, Maurer, et al. [59] used correlation-based feature selection. With this approach optimal features are defined as those which exhibit high within-class but low between-class correlations. Another method for feature selection is a forward–backward search in which features are sequentially added and removed from a larger set. Optimal features are identified depending on the resulting classification accuracies for each feature subset. This approach was used by Pirttikangas et al. [60] to identify the best sensors/features for the classification of 17 different activities.

In our experiment, we used the *sequential forward search* feature selection method to obtain optimal feature sets due to its simplicity.

5.6.1 Sequential Forward Search (SFS)

The SFS feature selection method is particularly suitable for finding a few "golden" features. The following steps are used to find the optimal number and types of audio features:

1. Compute criterion value (the classification accuracy, in our case) for all individual feature types (i.e., ZCR, STE,RMS, TC,...). Then select the audio feature that produced the best value (the feature that provided the highest recognition accuracy)'

2. Form all possible two-dimensional feature vectors that contain the winner from the previous step. Calculate the criterion (recognition accuracy) for each feature vector and select the best one (with the highest accuracy).

3. Continue adding features one at time, taking always the one that results in the largest value of the criterion (i.e., highest recognition accuracy).

4. Stop when the desired feature vector dimension M is reached or other stopping criteria is reached (e.g., if the recognition accuracy is not increasing anymore or starts to decrease)

However, SFS is a suboptimal search procedure, since nobody can guarantee that the optimal r–1 dimensional vector has to originate from the optimal r dimensional one. In addition, the search can be computationally expensive if the feature dimension is very large.

Chapter 6

Off-line Audio Classification and Event Detection

Unlike speech and music recognition, the research on environmental sound recognition (ESR) is not yet well matured. It is still at its infant stage which makes it difficult to obtain standard procedures and well organized information to determine the best audio feature extraction techniques and classification algorithms, based solely on the literature study. As a result, it is imperative to make further experimental test and simulations in order to be able to determine the best techniques. Off-line test/simulation is performed using MatLab in order to compare and evaluate the performance of different audio feature extraction and audio classification techniques and then select the best one. The performance comparison of the techniques is performed based on their recognition accuracy and computational complexity (speed).

This chapter provides analysis and MatLab simulation results of the performance of various audio classifiers and features. The first section introduces the dataset used in the simulations. In the subsequent sections, the performance of each of the three popular classifiers, k-NN, SVM and GMM, is evaluated and compared (in terms of recognition accuracy and computation time), based upon which the best classification algorithm is selected. The selected classification algorithm is then tested with different combinations of set of audio features and parameters in order to further improve its performance.

6.1 Simulation Setup

Even though there are numerous studies for speech and music recognition, very limited numbers of studies have been done on environmental sound domain till now. Due to this reason, there is no well know standard datasets like TIMIT [61] , which is a standardized utterance dataset for speech recognition performance experiments. Each researcher who studies on environmental domain sound recognition uses a different dataset which have different structures, sound type and quality. As a result, for our experiment we collected and recorded environmental sounds of our own to be used as a dataset.

6.1.1 Datasets

The sound recording was conducted in the University of Twente office corridors during normal working hours (with some possiblity of background noises). Unlike in other previous works which uses high quality stand alone microphones, we used the Smartphone's (Samsung Galaxy S-III) internal microphone for recording the sounds with a sampling rate of 16000 Hz,mono-channel, 16-bit per sample.

In order to investigate the effect of dataset size on classification performance, we used two types of datasets in our simulations. The first dataset contains a small total number of data samples (144 data samples), referred to as Small Dataset, whereas the second dataset contains large total number of data samples (317 data samples), referred to as Big Dataset. Both datasets contain 6 types of audio context (class) or sound activities: walking, elevator, coffee machine, water tape, door opening/clossing and silence. All the sound files (6 sound files corresponding to each of the 6 sound classes) were collected as uncompressed wav file with varying time durations. Using analysis window (mid-term window) of 2 secs with 50% overlap, the number of mid-term segments, which represent the number of data samples, is computed using equation 2.3.

The Small Dataset and Big Dataset are provided in Table 6.1 and Table 6.2, respectively.

Audio Clases	**No. of mid-term segments (samples)**	**Descriptions**
coffeeMachine	28	Sound produced by coffee machine brewing
doorOpen/Close	26	Produced by door opening or closing
elevator	14	Sound of elevator
silence	21	No activities
walking	36	Sound produced when some one is walking
waterTape	19	Sound of water tape (pipe) recorded while a person/user is washing hands
textbfTotal	**144**	

TABLE 6.1: Small Dataset

Audio Clases	**No. of mid-term segments (samples)**	**Descriptions**
coffeeMachine	59	Sound produced by coffee machine brewing
doorOpen/Close	53	Produced by door opening or closing
elevator	40	Sound of elevator
silence	61	No activities
walking	51	Sound produced when some one is walking
waterTape	53	Sound of water tape (pipe) recorded while a person is washing hands
Total	**317**	

TABLE 6.2: Big Dataset

6.2 Performance Evaluation Metrics and Methods of Audio Classifiers

A number of criteria can be used for evaluating and comparing the performance and efficiency of the classifiers. In this thesis, we compare them based on recognition accuracy and computational complexity (or computational speed). Before we present the performance results, we first describe measures that we used to quantify the performance of the classifiers.

6.2.1 Performance Measures

- *Overall accuracy, (Acc)* - Overall accuracy of a classifier is defined as the fraction of samples of the dataset that have been correctly classified. The overall accuracy (Acc) can be computed by dividing the number of correctly classified samples by the total number of samples in the dataset.

$$Acc = \frac{T_{correctClassification}}{T_{dataset}} \tag{6.1}$$

  where $T_{correctClassification}$ is total number of correctly classified datasets and $T_{dataset}$ is total number of datasets. Obviously, the quantity $1 - Acc$ is the overall classification error.

- *Recall, (Re(i))* - Apart from the overall accuracy, which characterizes the classifier as a whole, there also exist two class-specific measures that describe how well the classification algorithm performs on each class. One of these measures is the*class recall (Re(i))*, which is defined as the proportion of data with true class label i that were correctly assigned to class i.

$$Re(i) = \frac{i_{correctClassification}}{i_{dataset}} \tag{6.2}$$

  where $i_{correctClassification}$ is the number of datasets that have been correctly classified to class label i and $i_{dataset}$ is the number of dataset that belong to class label i.

- *Precision, (Pr(i))* - The second class specific performance measure is called *class precision (Pr(i))*, which is defined as the fraction of samples that were correctly classified to class i if we take into account the total number of samples that were classified to that class.

$$Pr(i) = \frac{i_{correctClassification}}{i_{totalClassification}} \tag{6.3}$$

  where $i_{correctClassification}$ is the number of datasets that have been correctly classified to class label i and $i_{totalClassification}$ is the total number of datasets that have been classified to class label i.

- *Harmonic mean of Recall and Precision* ($F_1(i)$) - Finally, a widely used class specific performance measure that combines the values of precision and recall is the

Harmonic mean of Recall and Precision, also known as $F_1 - measure$. It is computed as harmonic mean of the precision and recall values:

$$F_1(i) = \frac{2Re(i)Pr(i)}{Pr(i) + Re(i)} \tag{6.4}$$

6.2.2 Validation Methods

The crucial stage in the life cycle of any classifier is the validation stage, during which the correctness of the classification results should be determined. The choice of the training and testing dataset is the most important factor in the validation stage. Our dataset consists of feature vectors and respective class labels. Therefore, an important question addressed during the validation stage is how our dataset should be partitioned to training and testing datasets. Once this decision is made, it is then possible to train the classifier on the training set and measure its performance on the testing dataset using any of the performance measures.

For the validation of results and to decide how to split the datasets, several cross-validation techniques have been proposed in the literature, including:

- *Hold-out validation (HO)*- Hold-out method partitions the dataset into two non-overlapping subsets: one for training and the other for testing. Commonly, one-third of the samples are used for testing and the rest for training. However, this method is sensitive to the choice of datasets. For example, important (representative) samples can be left out of the training set, leading to a less accurate classifier. On the other hand, if the number of testing samples is significantly reduced in order to increase the size of the training set, then our confidence in the derived performance measures is also likely to decease.

- *Repeated random sub-sampling validation (Repeated-hold-out)(RHO)*- A remedy for the weakness of the standard hold-out method is to repeat the method a number of k times. At each iteration, the dataset is randomly split into two subset (random sub-sampling): training dataset and testing dataset. The classification performance is obtained by averaging the result of each hold-out iterations. Due to the random nature of the splitting process, it is still possible that some samples

may never be participate in the training (or testing) dataset, however, compared to the standard hold-out method the risk of over-fitting is reduced.

- *Leave-one-out (LOO)* - In the leave one out method all the samples, apart from one, are used for training the classifier. The remaining (one) sample is then used in the testing stage. This process is iterated a number of times (until all the dataset are covered) each time by reserving a sample for testing. The classifier performance is computed by averaging the results obtained from each of the iterations. LOO is used usually when there are small number of data samples available.

In our simulations, we tested both the *repeated random sub-sampling validation (RHO)* and *leave-one-out cross validation (LOO)* methods. In the case of RHO validation method, one-third of the total dataset is randomly selected and used for testing and the rest are used for training at every iteration. The maximum number of iterations is set to 20. Therefore, the overall recognition performance is computed by averaging the results obtained from each iterations (20 iterations). However, after repeated simulations, we observed that the result of each repeated (RHO based) simulation is not consistent and slightly varies from one simulation result to another. This slight variation happens due to random selection of the training data and testing data. On the other hand, the simulation result obtained using the LOO validation method is more stable (consistent) for each simulation test (trial) and, thus, more reliable. Consequently, we use the LOO based simulation results for comparison and evaluation purposes.

The recognition accuracy of each of the classifiers is presented in a *Confusion Matrix*. The rows and columns of the *Confusion Matrices* refer to the true (ground truth) and predicted class labels of the dataset, respectively.

Matlab profiler is used in order to measure the computational speed of the classifiers. We used a machine (laptop) with Intel Core i3 CPU and processor speed of 2.27 GHz for testing the performance of the algorithms.

6.3 Performance Evaluation Results

6.3.1 Performance of k-NN Classifier

Recognition Accuracy

The *k-NN* classifier has only one parameter to be adjusted, i.e., k value. In order to asses the effect of dataset size on the classification performance, the k-NN classifier is applied on both datasets (i.e., Small Dataset and Big Dataset). We used LOO for cross-validation and analysis widow of 2 secs (with 50% overlap). In the simulation, the entire feature set, which has been composed in section 5.3.2, has been used as our feature vector.

Figure 6.1 presents the overall classification accuracy for different values of k. The maximum overall accuracy is obtained for value of k equal to 2 in both datasets.

Table 6.3 and Table 6.4, respectively, present the confusion matrices of the *k-NN* classifier on the Small Dataset and Big Dataset using *LOO* validation method. The confusion matrices have been computed using the best value of k (in our case, k=2), for which the overall accuracy is maximum. Each element of the confusion matrices represents the *recall* values for each class.

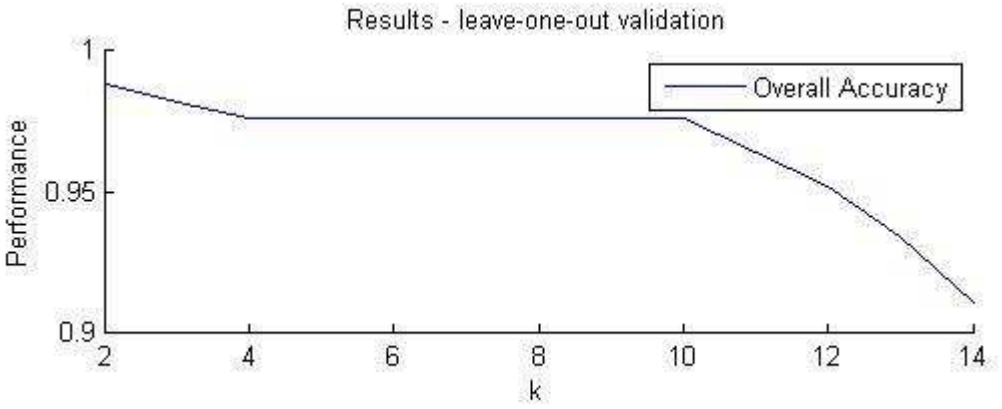


FIGURE 6.1: k-NN Performance vs k values

Confusion Matrix

	Predicted					
True ⇓	coffee	door	elevator	silence	walk	waterTape
coffee	92.9	7.1	0.0	0.0	0.0	0.0
door	0.0	100.0	0.0	0.0	0.0	0.0
elevator	0.0	0.0	100.0	0.0	0.0	0.0
silence	0.0	0.0	0.0	100.0	0.0	0.0
walk	0.0	0.0	0.0	0.0	100.0	0.0
waterTape	0.0	0.0	0.0	0.0	0.0	100.0

Performance Measures (per class)

Precision:	100.0	93.3	100.0	100.0	100.0	100.0
Recall:	92.9	100.0	100.0	100.0	100.0	100.0
F1:	96.3	96.6	100.0	100.0	100.0	100.0

Table 6.3: Confusion Matrix for Small Dataset.
Overall accuracy: 98.8, Average Precision: 98.9, Average Recall: 98.8, Average F1 measure: 98.8

Confusion Matrix

	Predicted					
True ⇓	coffee	door	elevator	silence	walk	waterTape
coffee	96.6	3.4	0.0	0.0	0.0	0.0
door	0.0	94.3	0.0	0.0	5.7	0.0
elevator	0.0	0.0	100.0	0.0	0.0	0.0
silence	0.0	0.0	0.0	100.0	0.0	0.0
walk	0.0	0.0	0.0	0.0	100.0	0.0
waterTape	0.0	0.0	1.9	0.0	1.9	96.2

Performance Measures (per class)

Precision:	100.0	96.5	98.1	100.0	93.0	100.0
Recall:	96.6	94.3	100.0	100.0	100.0	96.2
F1:	98.3	95.4	99.1	100.0	96.4	98.1

Table 6.4: k-NN: Confusion Matrix for Large Dataset
Overall accuracy: 97.9, Average Precision: 97.9, Average Recall: 97.9, Average F1 measure: 97.9

The classification accuracy of *k-NN* on the 6-audio classes is pretty high. The overall recognition accuracy is equal to 98.8% and 97.9% for the Small Dataset and Big Dataset, respectively. The simulation result indicates that the overall accuracy for the Small Dataset is slightly higher than that of the Big Dataset. In the case of Small Dataset, coffee brewing machine scored the least class recall (92.9%). 7.1% of the coffee machine dataset are misclassified as door opening/clossing.

Based on the simulation results we can conclude that increasing the dataset reduces (or do not improve) the recognition accuracy of the k-NN classifier. The k-NN classifier is vulnerable to noise. Using large dataset can introduce unnecessary redundancy (irrelevant features) and noise leading to reduced overall accuracy.

Computational Speed

The *k-NN* classifier is the simplest, yet quite powerfull classification algorithm. In addition, it does not have training stage, which makes it unique to the other classification algorithms.

The total execution time of the *k-NN* classifier is profiled with MatLab profiler. It is found out that the classifier requires, in average, 0.52 seconds to classify the Small Dataset and 1.47 seconds for the Big Dataset.

We conclude that using large dataset reduces the recognition accuracy and increases the computation time. Moreover, The Big Dataset requires more memory for storing the dataset.

6.3.2 Performance of SVM Classifier

The SVM is a sophisticated classification algorithm which has been employed in many applications. Next, we present the recognition accuracy and computational speed of the SVM classifier.

Recognition Accuracy

We used both datasets (i.e., Small Dataset and Big Dataset) for training and testing the SVM classifier, which uses One-Vs-All (OVA) strategy for multi-class classification.

Similar to the evaluation of *k-NN* classifier, LOO (Leave-One-Out) cross validation method has been used to compute the recognition performance with analysis widow of 2 secs and 50% overlap. In the simulation, we used the entire feature set that has been composed in section 5.3.2 as our feature vector.

In the case of *SVM* classifier, there are a number of parameters that need to be first set by the user: *type of kernel* (e.g., linear, polynomial, radial basis function (rbf), quadratic, Multilayer Perceptron kernel (mlp) and quadratic), *kernel properties* (parameters or values)(e.g.,in the case of polynomial kernel, the order of the polynomial must be specified by the user) and *constraint parameter, C*. We investigated different types of kernel functions (linear, polynomial, radial basis function, quadratic and mlp) in this work and found out that the *linear* kernel function to perform the best as shown in figure 6.2. Tables 6.5 and 6.6, respectively, present the confusion matrices of the SVM classification accuracy for the Small Dataset and Big Dataset using LOO cross-validation method. The results presented in the confusion matrices represents the class recall values and are obtained using the *linear* kernel function, which has the best performance score.

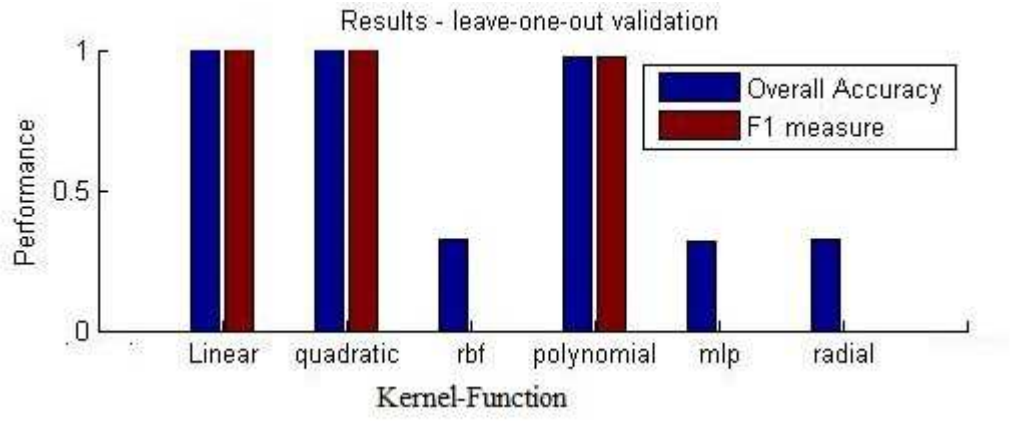


FIGURE 6.2: SVM Performance for different kernel functions

Leave-one-out, best kernel function=='linear'

	Predicted					
True ⇓	coffeeMachine	doorOpen/Close	elevator	silence	walking	waterTape
coffeeMachine	100.0	0.0	0.0	0.0	0.0	0.0
doorOpen/Close	0.0	100.0	0.0	0.0	0.0	0.0
elevator	0.0	0.0	100.0	0.0	0.0	0.0
silence	0.0	0.0	0.0	100.0	0.0	0.0
walking	0.0	0.0	0.0	0.0	100.0	0.0
waterTape	0.0	0.0	0.0	0.0	0.0	100.0

Performance Measures (per class)						
Precision:	100.0	100.0	100.0	100.0	100.0	100.0
Recall:	100.0	100.0	100.0	100.0	100.0	100.0
F1:	100.0	100.0	100.0	100.0	100.0	100.0

TABLE 6.5: SVM performance Confusion Matrix (for Small Dataset). Overall accuracy: 100.0, Average Precision: 100.0, Average Recall: 100.0, Average F1 measure: 100.0

Leave-one-out, best kernel function=="linear"

	Predicted					
True ⇓	coffee	door	elevator	silence	walk	waterTape
coffee	100.0	0.0	0.0	0.0	0.0	0.0
door	0.0	94.3	0.0	0.0	1.9	3.8
elevator	0.0	0.0	100.0	0.0	0.0	0.0
silence	0.0	0.0	0.0	100.0	0.0	0.0
walk	0.0	0.0	0.0	0.0	100.0	0.0
waterTape	0.0	3.8	0.0	0.0	0.0	96.2

Performance Measures (per class)						
Precision:	100.0	96.2	100.0	100.0	98.1	96.2
Recall:	100.0	94.3	100.0	100.0	100.0	96.2
F1:	100.0	95.2	100.0	100.0	99.1	96.2

TABLE 6.6: SVM performance Confusion Matrix (for Big Dataset) Overall accuracy: 98.4, Average Precision: 98.4, Average Recall: 98.4, Average F1 measure: 98.4

The simulation result indicates that the overall recognition accuracy of the SVM classifier is slightly higher (100%) when applied on the Small Dataset than when applied on the Big Dataset (98.4%). This slight reduction is caused due to over-fitting.

Computational Speed

The computational time (speed) of the SVM classifier is measured using the MatLab profiler. The time required to train and then classify the Small Dataset and Big Dataset is, respectively, equal to 16.13 and 89.9 seconds, which is very high compared to that of the *k-NN* classifier. Though the SVM classifier provides the highest recognition accuracy, the required computation time is too expensive to be used in applications with real-time requirements. The high execution time also indicates that SVM classifier consumes more energy which makes it unsuitable to be used in energy constrained devices such as smartphones.

6.3.3 Performance of GMM Classifier

Unlike the k-NN and SVM which are deterministic classifiers, GMM classifier is a statistics (probability) based classifier. The common characteristics of probability based classifiers is that they provide better classification results when used in classification applications that has large size of dataset. However, the down side of these classifiers is that the high computational time required, in addition to the requirement of large dataset. Next, we look into the classification performance of the GMM classifier.

Recognition Accuracy

In the case of Gaussian mixture model (GMM), there is one parameter that need to be set by the user, i.e., the number of GMM components that give the best recognition accuracy has to be determined in advance by the user. In order to find the number of GMM components that give maximum recognition accuracy, we have performed a simulation by varying the number of GMM components. Figure 6.4 presents the recognition accuracy for different number of mixture components (k). Based on our simulation, the best number of mixture components is found to be equal to 1 (i.e., k=1) using the LOO cross validation method.

The GMM classifier is investigated using both types of datasets (the Small Dataset and the Big Dataset). It is first tested using the Small Dataset, presented in Table 6.1. However, GMM is found to perform quite poor with the Small Dataset as shown in Figure 6.3. The confusion matrix for the best number of mixture components, which is 1 in this case, is presented in Table 6.7. We can see all audio events are simply classified as coffee machine (to the first class) and the maximum overall accuracy is quite low which is only equal to 16.7%. In fact, classifying all the test data as belonging to only one class (the first class) shows that the classifier completely failed to model and recognize the test data. The 16.7% recognition accuracy is produced since the 16.7% of the test data originally belong to the first class. Therefore, the recognition accuracy in this case is actually zero.

On the other hand, when we increased the size of the dataset, we observed that the performance of the GMM classifier improving. Using the Big Dataset (of size 317 data samples), we obtained a comparable recognition performance to that of k-NN and SVM classifiers. The obtained maximum overall accuracy using the Big Dataset is equal to 90.0%. The corresponding confusion matrix is presented in Table 6.8, computed using LOO cross-validation methods.

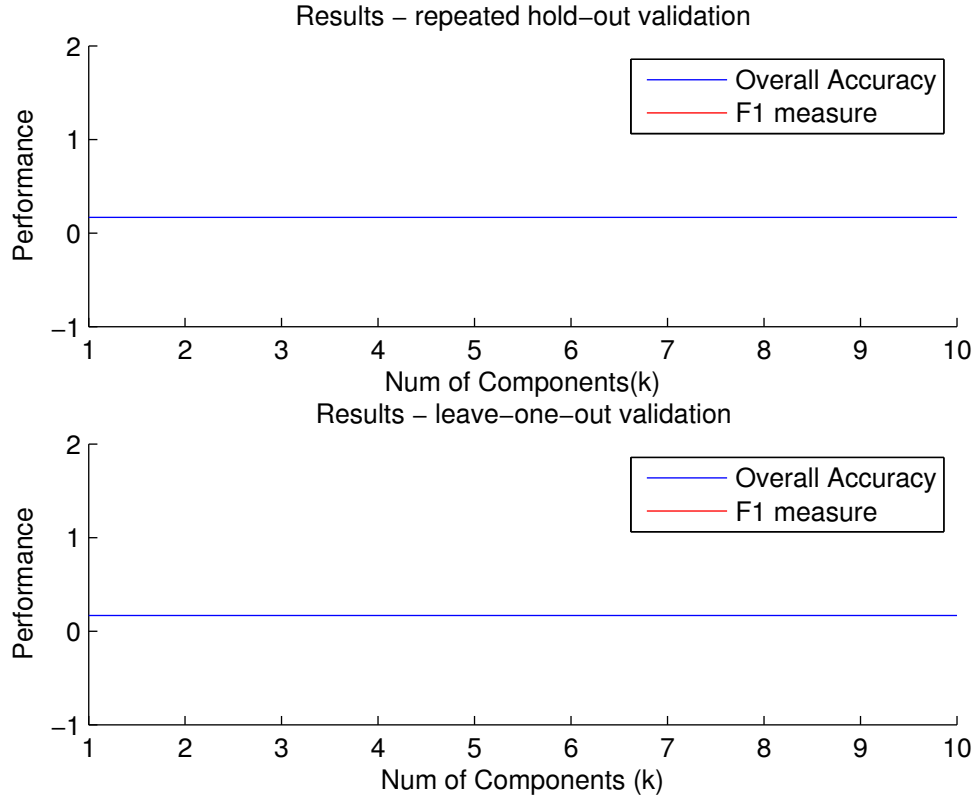


FIGURE 6.3: GMM Performance for different number of gmm components, k (Low GMM performance for small datasets)

Leave-one-out (best value of gmm component, k=1)

	Predicted					
True ⇓	coffee	door	elevator	silence	walk	waterTape
coffee	100.0	0.0	0.0	0.0	0.0	0.0
door	100.0	0.0	0.0	0.0	0.0	0.0
elevator	100.0	0.0	0.0	0.0	0.0	0.0
silence	100.0	0.0	0.0	0.0	0.0	0.0
walk	100.0	0.0	0.0	0.0	0.0	0.0
waterTape	100.0	0.0	0.0	0.0	0.0	0.0

Performance Measures (per class)

Precision:	16.7	NaN	NaN	NaN	NaN	NaN
Recall:	100.0	0.0	0.0	0.0	0.0	0.0
F1:	28.6	NaN	NaN	NaN	NaN	NaN

TABLE 6.7: Low GMM performance confusion matrix (LOO).
Overall accuracy: 16.7, Average Precision: NaN, Average Recall: 16.7, Average F1 measure: NaN

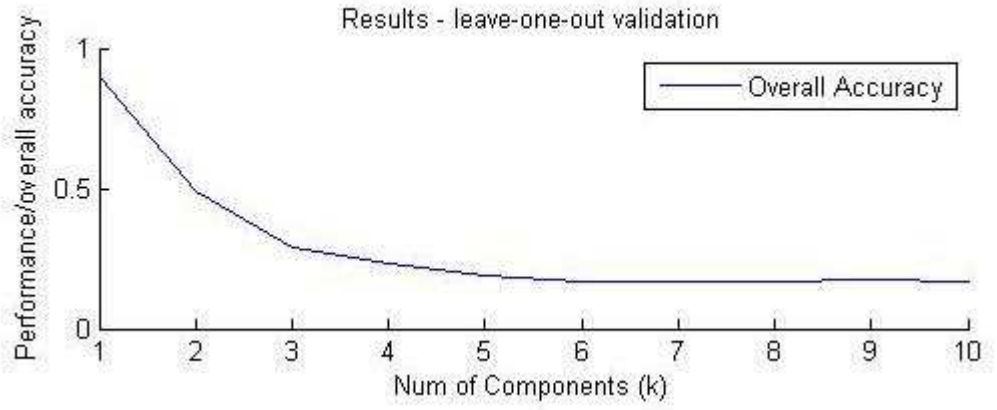


FIGURE 6.4: GMM Performance for different number of gmm components, k (High performance for large dataset)

Confusion Matrix

	Predicted					
True ⇓	coffee	door	elevator	silence	walk	waterTape
coffee	94.9	5.1	0.0	0.0	0.0	0.0
door	0.0	100.0	0.0	0.0	0.0	0.0
elevator	22.5	7.5	70.0	0.0	0.0	0.0
silence	0.0	0.0	0.0	100.0	0.0	0.0
walk	3.9	13.7	0.0	0.0	82.4	0.0
waterTape	7.5	0.0	0.0	0.0	0.0	92.5

Performance Measures (per class)

Precision:	73.6	79.2	100.0	100.0	100.0	100.0
Recall:	94.9	100.0	70.0	100.0	82.4	92.5
F1:	82.9	88.4	82.4	100.0	90.3	96.1

TABLE 6.8: GMM-Confusion Matrix for Big Dataset
Overall accuracy: 90.0, Average Precision: 92.1, Average Recall: 90.0, Average F1 measure: 90.0

Computational Speed

We have already seen that the GMM classifier completely failed when used for classification of small size dataset (i.e., the Small Dataset). Therefore, it is not useful to discuss about the required computation time when the GMM is used with the Small Dataset. However, it makes sense to measure and compare the required computation

time of GMM when the larger size dataset (i.e., Big Dataset) is used (as the recognition accuracy is acceptable). The measured computation time for training the GMM and then classifying the Big Dataset is equal to 12 seconds, which is prohibitably high.

Based on the simulation results, we conclude that the GMM classifier has completely failed to model correctly the Small Dataset. Comparative performance was obtained only when the Big Dataset is used, which in turn result in a very expensive computation time. GMM is, thus, not preferable both because of its poor classification accuracy (especially with small dataset) and prohibitably very high computational time and memory requirement (especially with large dataset).

6.3.4 Comparison of Classifiers' Performance

In this section, we compare the performance of the three classifiers based on their recognition accuracy and computational speed (simplicity).

Table 6.9 summarizes the overall performances of the classifiers. Based on the table, GMM is not totally acceptable to be used in classification problems with small dataset (such as in our case). It can only be used when there is large dataset available. However, the required computation time is so high that it is not suitable to be used in devices with limited computational and memory resources. Though SVM has very high (100%) recognition accuracy, the required execution time is high too. As a result, it is not preferred for real-time and energy contained implementations. The *k-NN* classifier provides high classification performance (98.8%, almost equal to that of the SVM) with less execution time. Therefore, the *k-NN* classifier is the best algorithm that can be used for implementations of classification applications in devices with limited resources (i.e., limited energy and memory) such as *smartphones*. One common property of k-NN and SVM is that both classifiers perform well with the smaller dataset (Small Dataset).

Classifier	**Dataset**	**Overall Recognition Accuracy**	**Execution time** (in secs)
k-NN	Small Dataset	98.8%	0.52
	Big Dataset	97.9%	1.47
SVM	Small Dataset	100%	16.13
	Big Dataset	98.4%	89.9
GMM	Small Dataset	16.7%	–
	Big Dataset	90.0%	12.8

TABLE 6.9: Summary of Classifiers' Comparison: Small Dataset referes to the dataset in table 6.1 and Large Dataset to the dataset in table 6.2

So far we have been computing and comparing the recognition accuracy of the classifiers without looking the effects of other parameters such as audio features, class types, analysis window and overlap sizes. In fact, the classification (recognition) accuracy does not only depend on the type of classifier used. There are a number of other parameters that affect the recognition accuracy and the computation time as well. The major ones include type of audio feature, number and types of classes, number and size of feature vectors, and analysis window and overlap size. In the next sections, we investigate the effect of these parameters on the recognition accuracy of the *k-NN* classifier. The goal is to find the optimal parameters to further improve the recognition accuracy of *k-NN* before finally implementing on smartphone.

6.4 Parameter Selection

We conducted a series of additional simulations to examine the effect of parameters settings on the performance of the k-NN classifier. Only minor changes in performance were detected using different parameter settings, for example, varying the analysis window length and window overlap. The simulation result (confusion matrix) of the classification performance of k-NN using analysis window of 1 secs with 50% (0.5 secs) is provided in Table 6.10.

The confusion matrix in Table 6.10 presents the classification performance of k-NN when used to classify the Small Dataset. The simulation result shows that the overall accuracy of the k-NN is not affected by changing the size of analysis window and overlap. However, the computation time has now increased (to 1.2 secs) by a factor of 2 (twice) compared with that of 2 secs analysis window (with 50% or 1 secs overlap). Note that

reducing the analysis window (from 2secs to 1 secs) results in increased size of dataset which can lead to increased computation time.

leave-one-out (bestK=2)

	Predicted					
True ⇓	coffee	door	elevator	silence	walk	waterTape
coffee	96.4	3.6	0.0	0.0	0.0	0.0
door	0.0	96.2	0.0	1.9	1.9	0.0
elevator	0.0	0.0	100.0	0.0	0.0	0.0
silence	0.0	0.0	0.0	100.0	0.0	0.0
walk	0.0	0.0	0.0	0.0	100.0	0.0
waterTape	0.0	0.0	0.0	0.0	0.0	100.0

Performance Measures (per class)

Precision:	100.0	96.4	100.0	98.1	98.1	100.0
Recall:	96.4	96.2	100.0	100.0	100.0	100.0
F1:	98.2	96.3	100.0	99.0	99.0	100.0

TABLE 6.10: Overall accuracy: 98.8, Average Precision: 98.8, Average Recall: 98.8, Average F1 measure: 98.8

6.5 Feature Selection and Dimensionality reduction

We have seen in section 5.6 that the type of audio features used for the classification has a major impact on the classification performance and why care should be taken when selecting them. A good feature set should show little variation between sample features (subjects) of the same class but should vary considerably between samples belonging to different classes. Furthermore, it is important to minimize any redundancy between features as this can result in unnecessarily increased computational demands and, also, reduced accuracy with some classification methods. Therefore, the goal of feature selection is to select a subset of M features from D originally available feature set so that we reduce the dimensionality of the feature vector ($M < D$), which in turn reduces the computational time and improves the classification accuracy at the same time.

A number of different techniques, of varying complexity, have been used to select appropriate features for classification applications. In our work, we used the *sequential*

forward search (see section 5.6) feature selection method to obtain optimal feature sets. Initially, the recognition accuracy of each audio feature type was computed (using the *k-NN* classifier) in order to choose the best audio feature with highest accuracy. In the initial test, MFCC (mfcc) feature was found to perform the best with overall accuracy of 98.2%. This performance result was obtained using the first 13 mfcc coefficients. However, it is still important to find the optimal number of mfcc coefficients as well. To this end, the overall accuracy is improved (to 98.8%) when the number of first mfcc coefficients are reduced to 8.

Figure 6.5 presents the overall recognition accuarcy of the audio features when used to classify the audio dataset in table 6.1 separately (individually). The result shows that the mfcc feature type (with the first 8 coeffs used) performs the best with an overall accuracy of 98.8% (which is quite high) whereas the energy entropy (ee) performs the least with an overall accuracy of 55.5%. Thus, the mfcc feature is the most important audio feature and we test it again by combining it with the other features. We repeat this until we get the optimal feature combinations following the *sequential forward search* procedures.

Thus, in the second iteration, it is found a feature set with combination of MFCCs and spectral entropy (SE) to produce the highest overall accuracy (99.4%). Finally, after the third iteration, a feature set which is composed of three feature types (MFCC+SE+SC) has resulted in overall accuracy of 100%, which is the highest. Therefore, the optimal feature set is determined and consists of MFCCs (with first 8 coeffs.), spectral entropy (SE) spectral centroid (SC). That is, our feature vector dimension is now reduced from 24 to 10 (8+1+1). This optimal feature set not only provide better overall recognition accuracy, but also provides reduced computational time when used with *k-NN* classifier.

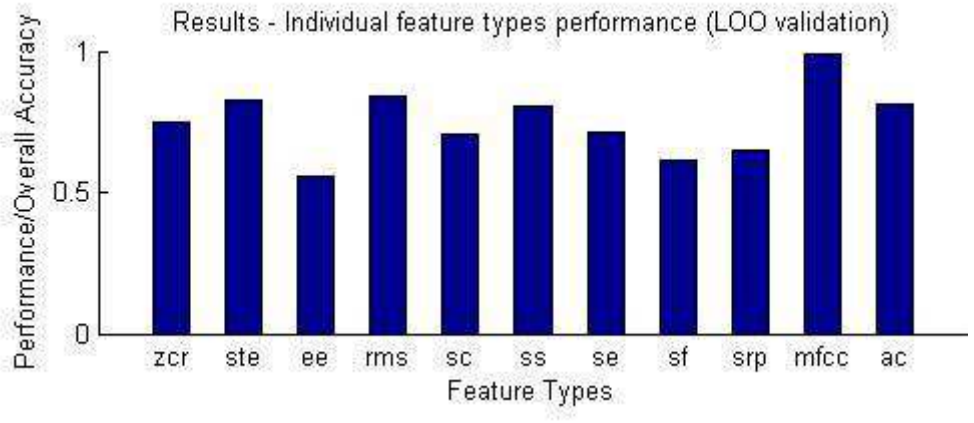


FIGURE 6.5: Individual feature performance: MFCC performs the best, overall accuracy 98.8% (the first 8 mfcc coefficients are used)

6.6 Summary: on *k-NN* Performance

After conducting a series of experiments (such as feature selection), it is finally possible to improve the overall accuracy of *k-NN* classifier to 100% (using reduced set of features) from 98.8% (original overall accuracy (see subsection 6.3.1)).

Generally, the best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques.

The accuracy of the *k-NN* algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. In the case of multi class problem, other techniques can be used such as increasing or decreasing the value of k until the tie is resolved.

Chapter 7

On-line Audio Classification and Recognition

Off-line audio processing can be used for applications where on-line audio recognition is not necessary. For instance, if we are interested in following the daily routine of human activities, 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 purposes. However, if one is interested in knowing the current context (user's current location or activities), on-line audio recognition technique becomes important. Unlike the Off-line recognition, the on-line audio (context) recognition provides a real-time information such as what a user is currently doing or where his current location is. This chapter discuss Android (mobile) application development process which is used for on-line audio recognition.

Since k-NN provided the best performance in the off-line recognition tests described in Chapter 6 , it is selected as audio classifier for the development of online audio recognition application. In addition, to develop the Android application, we used the following:

- The Small Dataset, presented in Table 6.1, – used as a training dataset.
- Analysis window of 2 secs.
- Sampling rate of 16 kHz, 16-bit per sample.
- The Optimal Feature Set, determined in Chapter 6 (Section 6.5), – used as feature vector.

Samsung Galaxy SII (with 1.2 GHz dual-core system on a chip (SoC) processor) has been used for developing and testing our application.

7.1 The On-line Audio Recognition Application

The online audio recognition technique has two components/phases (Fig. 7.1): off-line (training phase) and on-line (testing phase). Both the off-line training and on-line testing phases are performed on the smartphone.

- **Off-line** - The on-line audio recognition technique is first trained offline with previously collected data (i.e., the Small Dataset in our case). Data for off-line training must be labeled, i.e. we need to know which audio type (class) each analysis window belongs to.

  In this step, audio features (optimal audio features) from the training dataset are extracted and then stored on the smartphone's SD card memory for later retrieval during the on-line testing phase.

- **On-line** - In the online phase of the process, the Android app uses the audio features already extracted and stored in the previous step to match and detect new (unknown) audio events recorded in real-time from the microphone of the smartphone. The on-line process involves recording unknown audio signal, extracting audio features for the unknown audio signals, matching the extracted audio features with the audio features of the training dataset and finally displaying the recognition result to the user.

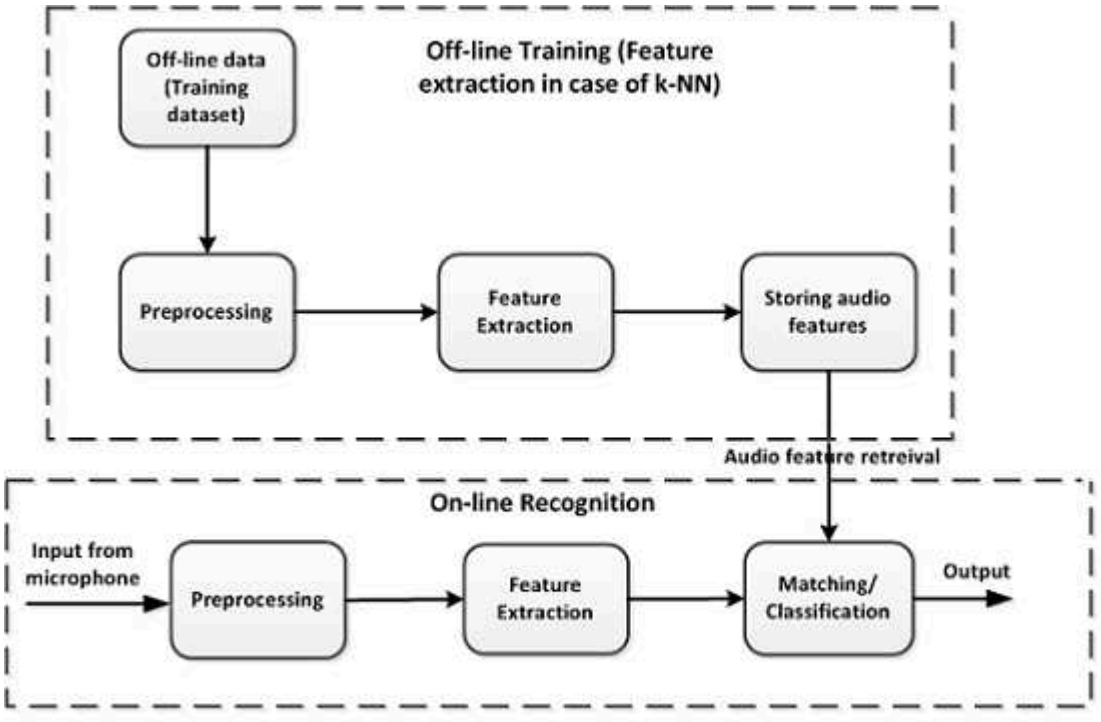


FIGURE 7.1: The classification process of on-line audio recognition application.

The phone records stream of raw audio data continuously (at 16 kHz sampling rate) and stores it into buffer. The contents of the buffer are read at a fixed time interval (2 secs) for processing (feature extraction and classification). The stream is segmented into short-term windows (of duration 0.05 secs with 50% overlap). Audio features are first computed for each short-term window and then averaged over the mid-term window (of duration 2 secs). For this work, we compute the three audio feature types, MFCCs (first 8 coeffs), SE and SC, which have been already tested to be the optimal feature set when used with k-NN classifier.

The graphical user interface (GUI) of the audio recognition application is illustrated in Figure 7.2. The main application user interface contains two buttons: *Start Listening to Audio* and *Start Generating Model* buttons. The *Start Generating Model* button (when clicked) is used to extract audio features from the training dataset and to store them. The *Start Listening to Audio* is used to record unknown audio events and to display the recognition results in real-time to the user. In other words, the *Start Generating Model* is used during the off-line training phase whereas the *Start Listening to Audio* is used

during the on-line audio recognition phase.

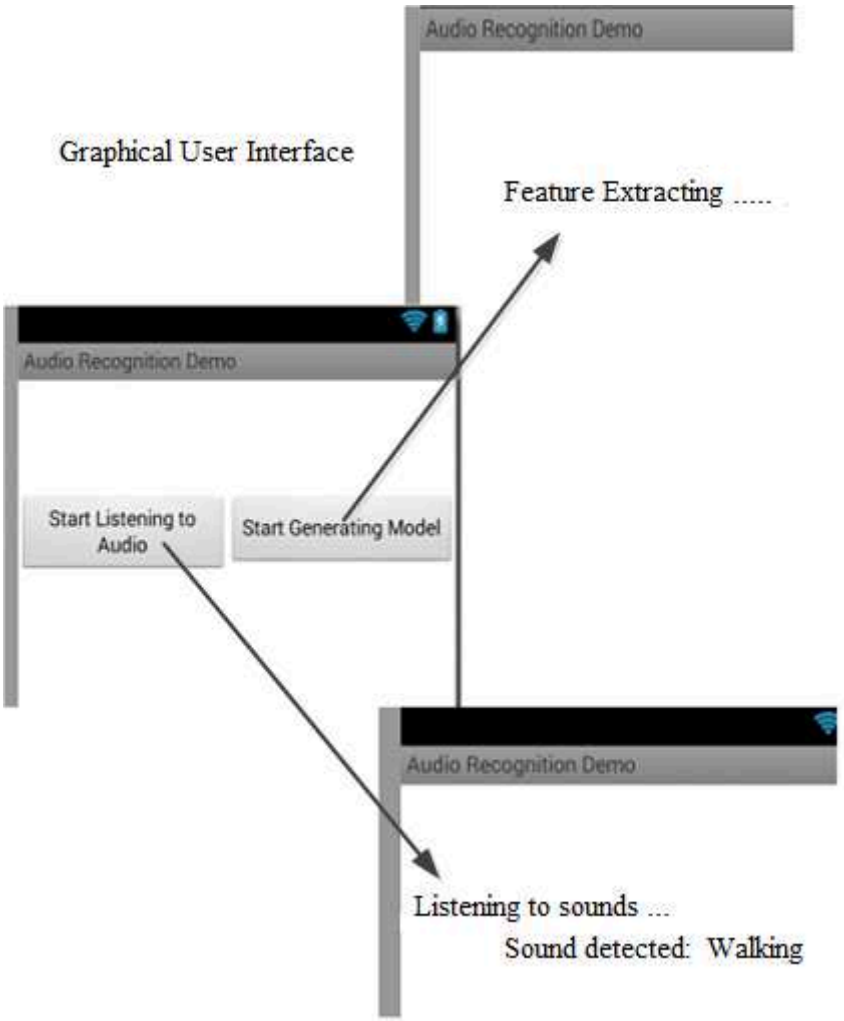


FIGURE 7.2: The GUI of Android App

7.2 Performance of On-line Audio Recognition

7.2.1 Computational speed

In order to measure the computational speed of the application, we used Android SDK tool, called *Traceview* [62]. *Traceview* is a Dalvik profiler which measures how much time the application spends calling methods. The tool provides the execution time

measurement in terms of both CPU time [1] and response (real) time [2], also called wall clock time. It can also show each threads execution and calls in chronological order.

During the off-line training (when the *Start Generating Model* button is pressed), the main activity (application) runs a thread that reads the training dataset and extracts the audio features. Finally, it stores the audio features in a file on the SD card memory. Based on the profiling results, the off-line training (i.e., reading and feature extraction of the training dataset) in average takes about 4 secs (real time or response time).

On the other hand, the recognition time of the on-line audio recognition technique is, in average, equal to 2 secs (response time). We define the recognition time as the time the on-line audio recognition technique needs to process and determine the audio type for one audio segment whose duration is equal to the analysis window (i.e., 2 secs). In short, the recognition time is the time required for activity recognition of one analysis window.

The feature extraction step takes 57.7% of the recognition time of which 55.2% is taken by the MFCCs feature extraction step. One interesting result is that even though the actual classification step uses only 1.5% of the recognition time, it takes 20.1% of the recognition time in order to read and load the audio features of the training data from the SD card during the classification stage. The recording and preprocessing stage account for a small percentage of the recognition time.

The result of the *Traceview* proves that the application can continuously recognize audio events (every 2 secs) in real-time. Fig. 7.3 shows the profiling result of the Traceview for the on-line audio recognition phase. The figure shows the profiling result for a duration of 2 secs.

As shown in the figure, the execution time log is displayed in two panels:

- **Time line panel** - describes when each thread and method starts and stops. In the time line panel, each thread's execution is shown in its own row, with time increasing to the right. The width of the bars corresponds to the CPU time a function/thread takes. Each method is shown in another color (colors are reused

[1] CPU time is the actual time the process uses the CPU (For example, this would not include waiting on input/output operations).

[2] Response time is the total time spent from the start of a process to the end of process. It includes waiting time for other process to execute; for example, time spent doing input/output operations

in a round-robin fashion starting with the methods that have the most inclusive time).

- **Profile panel** - provides a summary of what happens inside a method, i.e., a summary of entire period spent by a method. The profile panel shows both the inclusive and exclusive times as well as the percentage of the total time (in terms of both CPU time and real/response time). Exclusive time is the time spent in the method. Inclusive time is the time spent in the method plus the time spent in any called functions.

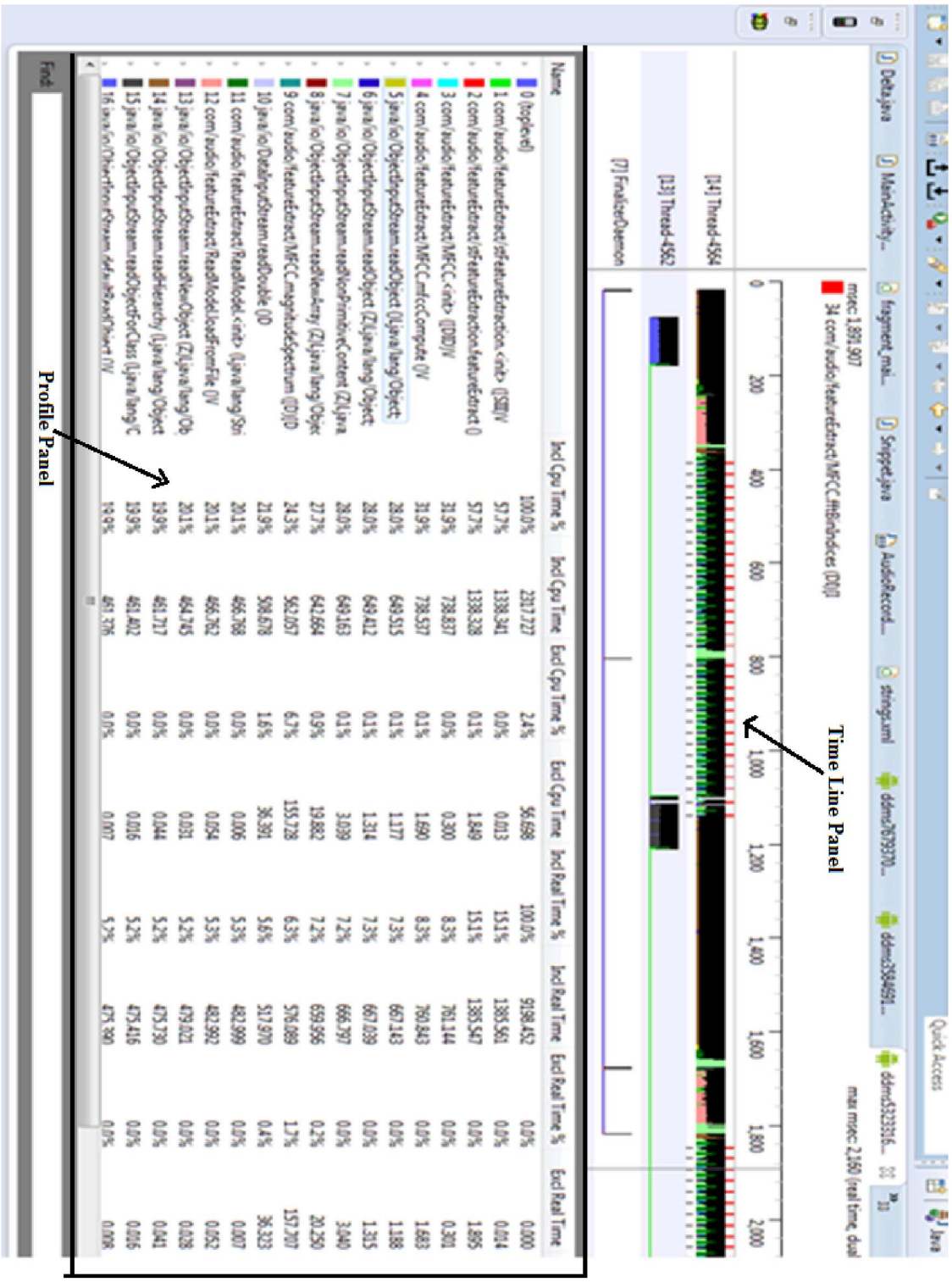


FIGURE 7.3: On-line audio recognition, execution time

7.2.2 Recognition accuracy

In order to test the recognition performance of the on-line audio recognition technique, we use a file of 82 secs duration that contains a continuous audio signal consisting

all audio types (i.e., walking, silence, elevator, door and coffee machine). This pre-recorded continuous audio file is played while at the same time smartphone's on-line audio recognition application is running. The recognition result is displayed in Fig. 7.4. The figure provides comparison of the on-line and off-line test results with that of the ground truth (real audio types) which has been labeled manually. The numbers in the figure indicate the time of (in secs) start and end of audio segment after merging similar consecutive audio types (events). It is assumed that the minimum duration of a single audio event is 2 secs (equal to the duration of the analysis window).

In 82 secs audio signal, we have in total 41 audio recognition events (mid-term frames). The on-line audio recognition technique misclassified 10 out of the 41 audio recognition events (mid-term frames). Thus, the recognition performance of the on-line audio recognition application is 75.6%.

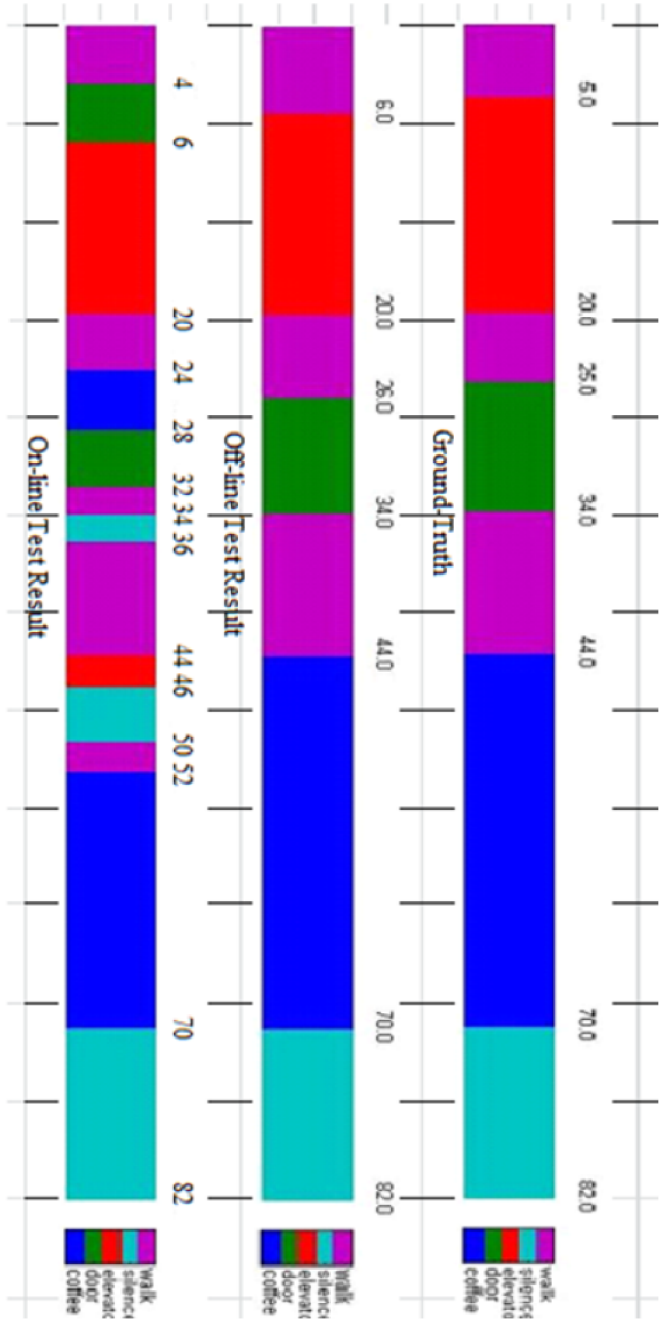


FIGURE 7.4: Continuous Sound Event Recognition

The figure shows that the result of the off-line audio recognition technique (performed using matlab simulation) is very similar to the ground truth except for a very short time period. On the other hand the result of on-line audio recognition technique has more number of misclassified segments. There are a number of reasons for the lower recognition performance of the on-line audio recognition:

- The first reason is sound overlapping at transition boundaries. As shown in Fig. 7.4, the rate of misclassification is higher during the transition from one type

of sound event to the next type. At the transition boundaries, there is higher chance of audio overlapping. This means that the analysis window is more likely to have mixture (combination) of more than one audio event type, which results in misclassification.

- Another reason for the lower recognition performance of the online audio recognition is noise interference. The on-line audio recognition is more susceptible to noise interference during testing (on-line testing). That is, during testing the microphone can pick up unwanted nearby audio events leading to noise interference which in turn impacts the classification performance.

- During the off-line audio recognition, we first collect audio data and apply classification algorithms (offline) on the collected data, using a large part of the collected data for training [3]. It is clear that the larger the amount of overlap between the training data and the testing data, the better recognition results will be achieved. Off-line processing exploits this advantage resulting in better recognition accuracy. However, it is problematic to use overlapping between successive analysis windows in the case of on-line audio recognition technique.

- Moreover, on-line audio recognition uses the naive merging while the off-line uses the probability smoothing post-processing technique. During off-line recognition, it is easier to use more advanced post-processing techniques such as probability smoothing techniques (section 5.5). Unlike the naive merging technique which simply merges similar successive classification results, the probability smoothing technique is so powerful that it can also filter out classification errors.

7.2.3 Memory usage

Random-access memory (RAM) is a valuable resource in any software development environment, but it is even more valuable on a mobile operating system where physical memory is often constrained.

Android's Dalvik virtual machine performs routine garbage collection in order to reclaim memory from unused objects of the app. The default maximum heap size allowed per

[3] We consider supervised learning methods thus the models require labeled training data to learn the model parameters

application, in Samsung Galaxy SII, is 64 MB. An application that requires a heap size larger than the maximum heap size allowed throws *OutOfMemmoryError*. In order to fit everything it needs in RAM, Android tries to share RAM pages across processes/apps. Due to the extensive use of shared memory, determining how much memory an app is using requires care.

There are different techniques for analyzing the memory usage of an Android app. The *Heap View* (in Eclipse) shows some basic statistics about an app's heap memory usage. The measurements show that the on-line audio recognition (tested during on-line testing) a memory heap size of 10 MB (in average) is allocated. The total number of objects (alive) is approximately 51,699 out of which about 20,000 have a size of 32 bytes each. Fig. 7.5 provides the result of the Heap View. As we can see from the figure, most of the objects have small size which is good in terms of memory usage. Larger objects lead to a larger heap size which cause for the Garbage Collector to run more often resulting in larger pause and longer execution time.

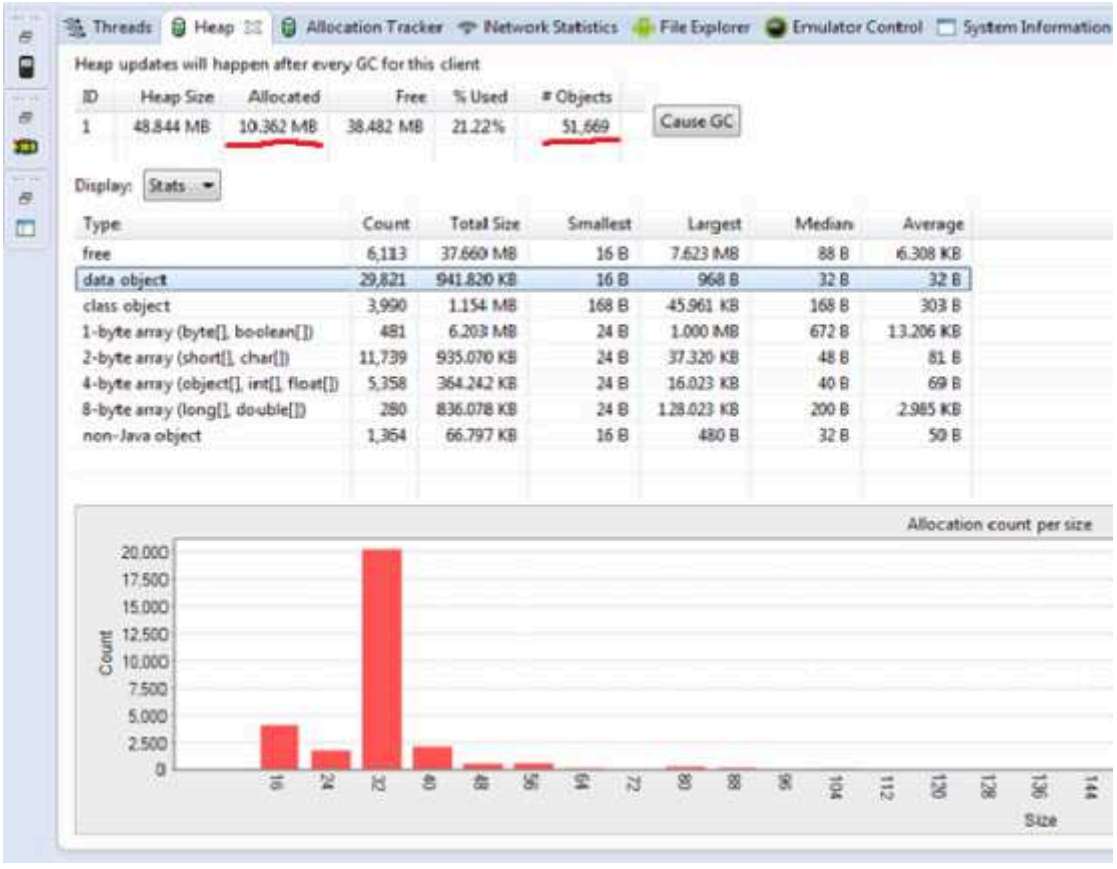
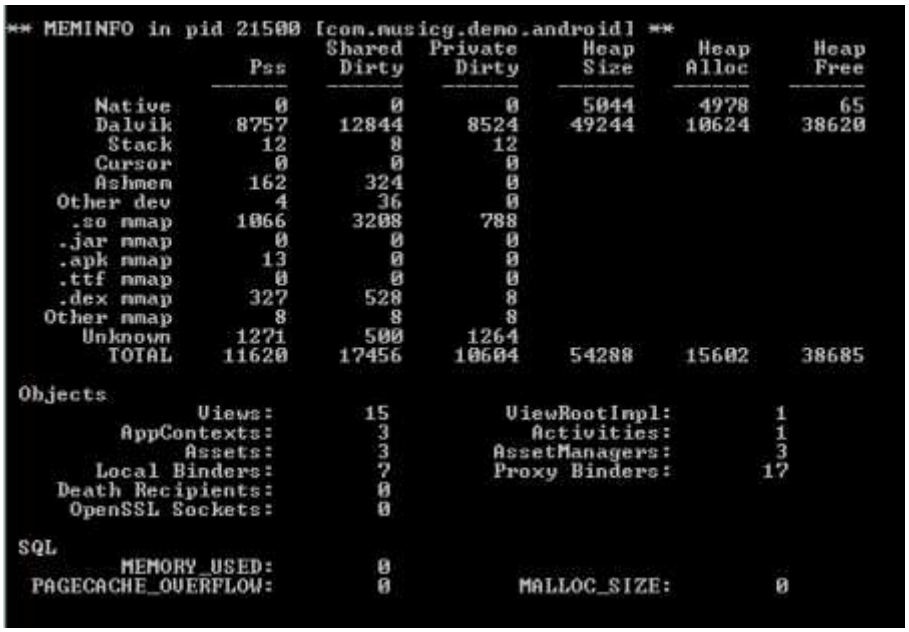


FIGURE 7.5: On-line:- Heap Memory Usage

We use the *adb shell dumpsys meminfo* command in order to view the overall memory allocation of the on-line audio recognition application. The output of the command is presented in Fig. 7.6 and lists all of the app's memory allocations, measured in kilobytes.



```
** MEMINFO in pid 21500 [com.musicg.demo.android] **
                   Pss  Shared  Private     Heap     Heap     Heap
                         Dirty    Dirty     Size    Alloc     Free
                ------  ------  ------  ------  ------  ------
      Native       0       0       0    5044    4978      65
      Dalvik    8757   12844    8524   49244   10624   38620
       Stack      12       8      12
      Cursor       0       0       0
      Ashmem     162     324       0
   Other dev       4      36       0
    .so mmap    1066    3208     788
   .jar mmap       0       0       0
   .apk mmap      13       0       0
   .ttf mmap       0       0       0
   .dex mmap     327     528       8
  Other mmap       8       8       8
     Unknown    1271     500    1264
       TOTAL   11620   17456   10604   54288   15602   38685

Objects
               Views:       15         ViewRootImpl:        1
         AppContexts:        3           Activities:        1
              Assets:        3        AssetManagers:        3
      Local Binders:        7        Proxy Binders:       17
   Death Recipients:        0
    OpenSSL Sockets:        0

SQL
         MEMORY_USED:        0
  PAGECACHE_OVERFLOW:        0          MALLOC_SIZE:        0
```

FIGURE 7.6: On-line:- Overall Memory Usage

In the figure above Pss stands for Proportional Set Size. It refers to the measurement of the app's RAM use that takes into account sharing pages across processes. Private (Clean and Dirty) RAM, on the other hand, refers to the memory that is being used by only the app's process. This is the bulk of the RAM that the system can reclaim when the app's process is destroyed. Generally, the most important portion of this is "private dirty" RAM, which is the most expensive because it is used only by the app's process and its contents exist only in RAM so cannot be paged to storage (because Android does not use swap).

The overall memory used by the app is, thus, approximately, equal to 7395 KB out of which 6636 KB is private dirty, which is not shared with other processes.

Note that the above outputs may vary across different platforms. In our test we used the Samsung Galaxy SII smartphone.

Chapter 8

Conclusion and Future Works

Environmental sounds/audios can provide many valuable cues for context-aware computing applications. From the audio signals we can infer type of activity and its context (e.g., its environment or location). This thesis provides the design and development of an application for correctly detecting and recognizing user activities and environmental context using audio signals on mobile phones. We compared performance of different audio classifiers (k-NN, SVM and GMM) and audio features based on their recognition accuracy and computational speed in order to choose a suitable (optimal) technique for implementing the application on mobile phones. We found out that the k-NN classifier provides high recognition accuracy with much less execution time. Moreover, we evaluated the performance of different types of audio features (both temporal and spectral) in order to select optimal feature set (optimal feature vector) for the audio classifier. As a result, feature set composed of Mel-frequency cepstral coefficients (MFCCs), spectral entropy (SE) and spectral centroid (SC) proved to provide the highest performance (when used with k-NN classifier) and , thus, chosen as optimal feature set. The test is performed off-line using MatLab simulations.

The on-line audio recognition application, which is implemented as an Android app, uses k-NN as audio classifier and the selected optimal feature set as audio feature vector. The application provides continuous real time classification results (every 2 seconds) by analyzing environmental sounds sampled from smartphone's microphone.

This work has investigated and proved the implementation feasibility of environmental audio based context recognition applications on mobile phones. However, there are still

a number of challenges that need to be addressed in the future. The performance and capabilities of the audio event recognition application can further be improved, in the future, by introducing a number of additional features and capabilities listed below:

- Adding audio segmentation and noise reduction component is important in order to further increase the recognition accuracy. Currently, the application performs poorly when there is audio overlapping (more than one sound event occur simultaneously) and background noise. Thus, introducing noise reduction and audio segmentation techniques at the pre-processing stage can improve the recognition accuracy (though at a cost of some computational time and complexities).

- Using on-line training technique in order to adapt to the dynamic nature of the environment and changing audio types. Our application uses off-line training (pre-recorded training dataset) which limits its ability to adapt to dynamic environment and new sound types. This limitation can be mitigated using on-line training techniques.

- It is evident that as the number of audio classes that the audio recognition application has to recognize increases, the efficiency and accuracy of the application decreases. This is because the more audio classes the application needs to recognize, the longer time it takes to train and classify. Also, as more similar audio classes are trained in the application, the difference between these audio classes will be too fine to allow a distinction, meaning that accuracy will decrease. In order to combat this decrease in efficiency as the number of audio class types increase, an environmental sound taxonomy, which classifies sounds on several levels before recognition, can be developed. The advantage of this approach is that each classification level contains a smaller set of audio classes, increasing the accuracy and efficiency of the audio classification.

Bibliography

[1] Woo-Hyun Choi, Seung-Il Kim, Min-Seok Keum, David K Han, and Hanseok Ko. Acoustic and visual signal based context awareness system for mobile application. *Consumer Electronics, IEEE Transactions on*, 57(2):738–746, 2011.

[2] Dan Smith, Ling Ma, and Nick Ryan. Acoustic environment as an indicator of social and physical context. *Personal and Ubiquitous Computing*, 10(4):241–254, 2006.

[3] Waltenegus Dargie. Adaptive audio-based context recognition. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(4):715–725, 2009.

[4] Antti J Eronen, Vesa T Peltonen, Juha T Tuomi, Anssi P Klapuri, Seppo Fagerlund, Timo Sorsa, Gaëtan Lorho, and Jyri Huopaniemi. Audio-based context recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):321–329, 2006.

[5] Ling Ma, Ben Milner, and Dan Smith. Acoustic environment classification. *ACM Transactions on Speech and Language Processing (TSLP)*, 3(2):1–22, 2006.

[6] Waltenegus Dargie and Tobias Tersch. of complex settings by aggregating atomic scenes. 2008.

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

[8] Jie Huang. Spatial auditory processing for a hearing robot. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 2, pages 253–256. IEEE, 2002.

[9] Chloé Clavel, Thibaut Ehrette, and Gaël Richard. Events detection for an audio-based surveillance system. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1306–1309. IEEE, 2005.

[10] Joaquín Gonzalez-Rodriguez, Julian Fiérrez-Aguilar, and Javier Ortega-Garcia. Forensic identification reporting using automatic speaker recognition systems. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–93. IEEE, 2003.

[11] Rong Dong, David Hermann, Etienne Cornu, and Edward Chau. Low-power implementation of an hmm-based sound environment classification algorithm for hearing aid application. In *Proc. EUSIPCO*, 2007.

[12] Jianfeng Chen, Alvin Harvey Kam, Jianmin Zhang, Ning Liu, and Louis Shue. Bathroom activity monitoring based on sound. In *Pervasive Computing*, pages 47–61. Springer, 2005.

[13] Michel Vacher, François Portet, Anthony Fleury, and Norbert Noury. Challenges in the processing of audio channels for ambient assisted living. In *e-Health Networking Applications and Services (Healthcom), 2010 12th IEEE International Conference on*, pages 330–337. IEEE, 2010.

[14] Jia-Ching Wang, Hsiao-Ping Lee, Jhing-Fa Wang, and Cai-Bei Lin. Robust environmental sound recognition for home automation. *Automation Science and Engineering, IEEE Transactions on*, 5(1):25–31, 2008.

[15] Tampere University of Technology Audio Research Team. Audio research. 2014. URL http://arg.cs.tut.fi/research/environmental-audio.

[16] Douglas Preis and Voula Chris Georgopoulos. Wigner distribution representation and analysis of audio signals: An illustrated tutorial review. *Journal of the Audio Engineering Society*, 47(12):1043–1053, 1999.

[17] Stephen J Preece, John Y Goulermas, Laurence PJ Kenney, Dave Howard, Kenneth Meijer, and Robin Crompton. Activity identification using body-mounted sensors—a review of classification techniques. *Physiological measurement*, 30(4): R1, 2009.

[18] Tâm Huynh and Bernt Schiele. Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pages 159–163. ACM, 2005.

[19] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.

[20] Stephen J Preece, John Yannis Goulermas, Laurence PJ Kenney, and David Howard. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. *Biomedical Engineering, IEEE Transactions on*, 56(3):871–879, 2009.

[21] Tae Hong Park. *Introduction to digital signal processing: Computer musically speaking*. World Scientific, 2010.

[22] Khaled El-Maleh, Mark Klein, Grace Petrucci, and Peter Kabal. Speech/music discrimination for multimedia applications. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 6, pages 2445–2448. IEEE, 2000.

[23] Tong Zhang and C-CJ Kuo. Audio content analysis for online audiovisual data segmentation and classification. *Speech and Audio Processing, IEEE Transactions on*, 9(4):441–457, 2001.

[24] Dongge Li, Ishwar K Sethi, Nevenka Dimitrova, and Tom McGee. Classification of general audio data for content-based retrieval. *Pattern recognition letters*, 22(5): 533–544, 2001.

[25] Lie Lu, Hong-Jiang Zhang, and Hao Jiang. Content analysis for audio classification and segmentation. *Speech and Audio Processing, IEEE Transactions on*, 10(7): 504–516, 2002.

[26] Costas Panagiotakis and Georgios Tziritas. A speech/music discriminator based on rms and zero-crossings. *Multimedia, IEEE Transactions on*, 7(1):155–166, 2005.

[27] Hong Lu, Wei Pan, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile

phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 165–178. ACM, 2009.

[28] José M Martínez, Rob Koenen, and Fernando Pereira. Mpeg-7: the generic multimedia content description standard, part 1. *MultiMedia, IEEE*, 9(2):78–87, 2002.

[29] William A Sethares, Robin D Morris, and James C Sethares. Beat tracking of musical performances using low-level audio features. *Speech and Audio Processing, IEEE Transactions on*, 13(2):275–285, 2005.

[30] André Gustavo Adami and Dante Augusto Couto Barone. A speaker identification system using a model of artificial neural networks for an elevator application. *Information Sciences*, 138(1):1–5, 2001.

[31] Tobias Andersson. Audio classification and content description. *Lulea University of Technology, Multimedia Technology, Ericsson Research, Corporate unit, Lulea, Sweden*, 2004.

[32] Spectral roll off. 2014. URL http://sovarr.c4dm.eecs.qmul.ac.uk/wiki/Spectral_Rolloff.

[33] Vesa Peltonen, Juha Tuomi, Anssi Klapuri, Jyri Huopaniemi, and Timo Sorsa. Computational auditory scene recognition. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 2, pages II–1941. IEEE, 2002.

[34] Hemant Misra, Shajith Ikbal, Hervé Bourlard, and Hynek Hermansky. Spectral entropy based feature for robust asr. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 1, pages I–193. IEEE, 2004.

[35] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[36] A. Moore. Statistical data mining tutorial on gaussian mixture models. 2014. URL http://www.autonlab.org/tutorials/gmm.html.

[37] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[38] Algorithms dynamic time warping. 2014. URL http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorithm.htm.

[39] Sergios Theodoridis, Aggelos Pikrakis, Konstantinos Koutroumbas, and Dionisis Cavouras. *Introduction to Pattern Recognition: A Matlab Approach: A Matlab Approach*. Academic Press, 2010.

[40] Kazuo Hattori and Masahito Takahashi. A new edited k-nearest neighbor rule in the pattern classification problem. *Pattern Recognition*, 33(3):521–528, 2000.

[41] Hanan Samet. K-nearest neighbor finding using maxnearestdist. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):243–252, 2008.

[42] Aslı Apaydın. *GMM Based Environmental Sound Recognition Using MFCC and MPEG-7 Audio Low-Level Descriptions*. PhD thesis, Eastern Mediterranean University, 2010.

[43] Ling Xing, Min Zhu, and Jinjun Hu. A multi-semantic audio classification method based on tensor space. *Journal of Information and Com-putational Science*, 9(4): 969–975, 2012.

[44] S SELVA NIDHYANANTHAN and R SHANTHA SELVA KUMARI. Language and text-independent speaker identification system using gmm. *Wseas Trans. Signal Process*, 4:185–194, 2013.

[45] DA Reynolds. Gaussian mixture models. encyclopedia of biometric recognition, 2008.

[46] Marcelo N Kapp, Robert Sabourin, and Patrick Maupin. A dynamic model selection strategy for support vector machine classifiers. *Applied Soft Computing*, 12(8):2550–2565, 2012.

[47] Himani Bhavsar and Mahesh H Panchal. A review on support vector machine for data classification. *International Journal of Advanced Research in Computer Engineering & Technology*, 1(10), 2012.

[48] C Lim and J-H Chang. Enhancing support vector machine-based speech/music classification using conditional maximum a posteriori criterion. *IET signal processing*, 6(4):335–340, 2012.

[49] Johannes Fürnkranz. Round robin classification. *The Journal of Machine Learning Research*, 2:721–747, 2002.

[50] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[51] Johannes Fürnkranz. Round robin classification. *The Journal of Machine Learning Research*, 2:721–747, 2002.

[52] Miin-Shen Yang, Chien-Yo Lai, and Chih-Ying Lin. A robust em clustering algorithm for gaussian mixture models. *Pattern Recognition*, 45(11):3950–3961, 2012.

[53] Thiruvengatanadhan Ramalingam and P Dhanalakshmi. Speech/music classification using wavelet based feature extraction techniques. *Journal of Computer Science*, 10(1):34, 2013.

[54] Manish P Kesarkar. Feature extraction for speech recognition. In *Tech. Credit Seminar Report, Electronic Systems Group, EE. Dept, IIT Bombay*, 2003.

[55] Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89, 2010.

[56] Mykola Pechenizkiy. The impact of feature extraction on the performance of a classifier: knn, naïve bayes and c4. 5. In *Advances in Artificial Intelligence*, pages 268–279. Springer, 2005.

[57] Huan Liu and Hiroshi Motoda. *Feature extraction, construction and selection: A data mining perspective*. Springer, 1998.

[58] MN Nyan, Francis EH Tay, AWY Tan, and KHW Seah. Distinguishing fall activities from normal activities by angular rate characteristics and high-speed camera characterization. *Medical engineering & physics*, 28(8):842–849, 2006.

[59] Uwe Maurer, Anthony Rowe, Asim Smailagic, and Daniel Siewiorek. Location and activity recognition using ewatch: A wearable sensor platform. In *Ambient Intelligence in Everyday Life*, pages 86–102. Springer, 2006.

[60] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. Feature selection and activity recognition from wearable sensors. In *Ubiquitous Computing Systems*, pages 516–527. Springer, 2006.

[61] William M Fisher, George R Doddington, and Kathleen M Goudie-Marshall. The darpa speech recognition research database: specifications and status. In *Proc. DARPA Workshop on speech recognition*, pages 93–99, 1986.

[62] Android Developers. Profiling with traceview and dmtracedump. 2014. URL http://developer.android.com/tools/debugging/debugging-tracing.html.