Dissertation on

Identification of Musical Instrument and Human Speakers based on Audio Analysis and Pattern Recognition Techniques

Thesis submitted towards partial fulfilment of the requirements for the degree of

Master of Technology in IT (Courseware Engineering)

Submitted by
Shruti Sarika Chakraborty

EXAMINATION ROLL NO.: M4CWE18004
UNIVERSITY REGISTRATION NO.: 137522 of 2016-2017

Under the guidance of

Dr. Ranjan Parekh
School of Education Technology
Jadavpur University

Course affiliated to

Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India
2018

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **“Identification of Musical Instrument and Human Speakers based on Audio Analysis and Pattern Recognition Techniques”** is a bonafide work carried out by **Shruti Sarika Chakraborty** under our supervision and guidance for partial fulfilment of the requirements for the degree of Master of Technology in IT (Courseware Engineering) in School of Education Technology, during the academic session 2016-2018.

--
Dr. Ranjan Parekh
SUPERVISOR
Jadavpur University
Kolkata-700 032

--
Prof. Matangini Chattopadhyay
DIRECTOR
School of Education Technology
Jadavpur University
Kolkata-700 032

--
DEAN-FISLM
Jadavpur University
Kolkata-700 032

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

<u>CERTIFICATE OF APPROVAL</u> **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactorily to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination
for evaluation of Thesis

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of her **Master of Technology in IT (Courseware Engineering)** studies during academic session 2016-2018.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referred all material and results that are not original to this work.

NAME: SHRUTI SARIKA CHAKRABORTY

EXAMINATION ROLL NUMBER: M4CWE18004

THESIS TITLE: IDENTIFICATION OF MUSICAL INSTRUMENT AND HUMAN SPEAKERS BASED ON AUDIO ANALYSIS AND PATTERN RECOGNITION TECHNIQUES

SIGNATURE: DATE:

Acknowledgement

I feel fortunate while presenting this dissertation at School of Education Technology, Jadavpur University, Kolkata, in the partial fulfilment of the requirement for the degree of Master of Technology in IT (Courseware Engineering).

I hereby take this opportunity to express warm thanks from the core of my heart to my guide and mentor Dr. Ranjan Parekh for being so kind and supportive throughout the course period. His advice and guidance have paved the path of success of this thesis. He has encouraged me and has guided me with all possible suggestions and constructive criticism. I will remain grateful to him, for my entire life.

I would like to express my warm thanks to Prof. Matangini Chattopadhyay, Director of School of Education Technology for her support and advices. I would also like to thank Mrs. Saswati Mukherjee and Mr. Joydeep Mukherjee.

My thanks and appreciation goes to my classmates from M.Tech IT (Courseware Engineering) and MMD. I do wish to thank all the departmental support staffs and everyone else who has different contributions in this dissertation.

Finally, my special gratitude to my mother and Anu who stood for me at all my odds and have supported me well enough to achieve this height.

Date:

Place: KOLKATA

SHRUTI SARIKA CHAKRABORTY

M.Tech IT (Courseware Engineering)
School of Education Technology
Jadavpur University
Kolkata-700032

Dedicated to,

BABA

Identification of Musical Instrument and Human Speakers based on Audio Analysis and Pattern Recognition Techniques

CONTENTS

List of Figures

List of Tables

ABBREVIATIONS

Acronym	Full-Name
ANN	Artificial Neural Network
KNN	K-Nearest Neighbor
GMM-UBM	Gaussian mixture model-Universal Background model
NED	Normalized Euclidian distance
MFCC	Mel frequency cepstral coefficient
CC	Cepstral coefficient
LPC	Linear predictive coding
HPCP	High Pitch Class Profiles
SVM	Support Vector Machine
OSTI-SI	Open set test independent speaker identification

Executive Summary

Sound is generated from various sources. It is generated by human beings' animals, vehicles, musical instruments, mechanical instruments, wind, sea and many other sources on this planet earth and beyond. The general range of hearing for human being is 20 Hz to 20 kHz. In this study out of many cases, where audio recognition or classification could had been helpful, two most important and popular cases have been taken under consideration. The first one being identification and further classification of musical instruments based on their audio signals and second one is open set text independent speaker identification with adaptive threshold. This study highlights different challenges faced in above mentioned cases and aims to solve them. The comparative study of how six different features of musical instrument sounds helps in identification of sounds presented in test case by matching against the ones in train case and hence further classifies them into five different groups of instruments with four classification algorithms like ANN,SVM,K-Non-random-Forest is made and it is inferred that CC or cepstral feature along with ANN or Artificial Neural Network suggested in this study best classifies the musical instruments irrespective of the pitch or tone or note in which they are played in .The second part of this study designs an open set text independent speaker identification system with adaptive threshold generated by the system with MFCC, Random forest classifier and multi input Random forest regressor i.e. basically it is the classification of known and unknown voices, with known voices mapped to its counterpart in train database and unknown voices detected as unknown and this process also undoes human interference of manually selecting a threshold for known and unknown classification as that is conducted by the system designed in proposed approach.

1.INTRODUCTION

1.1. Overview

In physics, sound is a vibration that typically propagates as an audible wave of pressure, through a transmission medium such as a gas, liquid or solid. In human physiology and psychology, sound is the reception of such waves and their perception by the brain. Humans can only hear sound waves as distinct pitches when the frequency lies between about 20 Hz and 20 kHz. Sound can be produced by almost all components of nature. It can be produced by both living and non-living entities present in nature. It is also produced by creations of mankind. Human beings and animals communicate through this entity. The sound can be represented in analogue and digital format. An analogue sound, is usually electrical. A voltage level representing air pressure of the wave form. It's a continuous function. On the other hand, a digital representation, expresses the pressure wave form as binary numbers. Thus, it is a discrete function. The digital representation permits the use of microprocessors and computer. The sound after it is converted to digital form, it is treated like any other signals in the digital signal processing domain. The sound after being converted into frequency domain by processes like DFT is represented by a spectrum. The magnitude and phase spectrum of the signal is further processed to obtain desirable features and can be used in other applications and processing algorithms. The FFT, which is the modified and faster version of DFT is aided by processes like windowing for better results. In this study after conducing these necessary actions further actions are taken in order to extract features of sound required for the process of matching with respect to its closest counterpart in the train set and classification of each of several such test samples to their respective groups/classes with which the system has been trained during the training phase.

In recent times, a lot of research is going on in the field of sound processing and recognition or classification. Sound recognition is the task involving recognizing a sound from a group of sounds or by comparing against a single sound. The former is sound identification while the latter is sound verification. Here sound identification task has been dealt with extensively. Sound classification is a sub part of sound identification, where goal is not only recognizing a single sound from a group of sounds but classify them into classes in which they belong to. For example, a train database may contain sounds from five types of animals. Each animal is represented by 10 sounds. The test set, i.e. where the audio signals, which requires to be labelled are kept, may contain 6 sounds. These 6 sounds may or may not belong to 5 classes of train set. The system's task will be to classify the unknown audio signal into unknown category and send a message to the user, telling its origin is unknown and labelling the other sounds, which belongs to one of those 5 classes correctly. So, suppose out of 6 audio signals, first one belongs to cat, second one belongs to dog, third one to elephant, fourth one to tiger and fifth one to lion while sixth one to man. The train database contains the voices of animals excluding man. So, the result will tell that sixth one is unknown while other voices will be matched to its closest counterpart and if the system is 100% accurate, the test voices of these respective animals will be matched to the train voices of the same, hence correctly labelled and classified.

This study is not concerned with identifying animal sound, rather it is concerned with identifying and classifying musical instrument sound and Human voices as these two fields have many applications and an active area of research nowadays.

In this study, the above-mentioned cases are dealt with. In this section the advantages and challenges faced by the systems are discussed briefly along with problem statement and objective of the thesis.

1.2. Applications

Musical instrument identification is the process where a test audio signal derived from a musical instrument is matched to one of the x (in this case x = 5) classes of musical instruments which are trained with their respective sounds during training phase with preferably many train samples encompassing variations found in them. In this way, the test audio signal is labelled and hence if there exists number of test samples, derived from multiple instruments of different families the system classifies each of them to their respective families based on what the machine has learnt from the training phase. This is known as musical instrument classification. As the domain of musical instruments is very wide and it is expanding so manual cataloguing is difficult and prone to errors. The advantages of musical instrument classification are, listed as follows.

1) Musical instrument classification is very important task for musical information retrieval system as the domain of musical instrument is very wide.

2) Musical instrument classification aids in extraction of melody from music sound, recognition and separation of sound sources in polyphonic audio, recognition of musical instrument in the isolation, automatic music transcription, beat tracking, musical information retrieval and many more application of similar types.

3) Researches in this domain helps in audio source separation, automatic music transcription, and genre classification.

Speaker Recognition is the process of recognizing individual based on his/her voice. It can be classified into two types, speaker verification and speaker identification. Speaker verification is the process of verifying a speaker’s claimed identity based on his/her already registered voice whereas speaker identification involves identifying whether a speaker’s voice matches or not with any member of several registered voices. Speaker verification is therefore a one to one matching process whereas speaker identification typically involves performing one to many matches. In this study, the most difficult subset of speaker identification which is open set text independent speaker identification has been dealt with. It involves a two-step process. The first step involves binary classification, where the speaker is detected as known or unknown compared to the already trained voices present in the training set, with which the system has been trained, the second step involves matching the known voices of the test set to the trained voices in the train set, with which it has closest similarity.

The speaker recognition finds its advantages in the following fields.

1) Typical applications of closed-set speaker verification include voice-based authentication systems while open-set speaker identification is required in surveillance and criminal investigations e.g. ransom callers.

2) Applications like access control are for physical facilities, and more recent applications are for controlling access to computer networks (add biometric factor to usual password and/or token) or websites (thwart password sharing for access to subscription sites). It is also used for automated password reset services. The open set speaker identification system, will detect if an unknown person is trying to enter a protected place where access is private and will deter him/her, hence protecting the privacy and protection of the place. A closed set speaker identification system, will fail in this case.

3) Some applications are there such like home-parole monitoring (call parolees at random times to verify they are at home) and prison call monitoring (validate inmate prior to outbound call). There has also been discussion of using automatic systems to corroborate aural/spectral inspections of voice samples for forensic analysis. In a scenario where a crime like kidnapping has taken place, the open set text independent speaker identification system can be used to trace ransom callers. If the caller is one from the train set, it will be detected and if it is not, that will also be said. This process will also help to determine the person in the train set with which the caller has closest similarity with respect to voice and hence further investigations can be conducted based on that.

4) In voice mail browsing or intelligent answering machines, speaker recognition is used to label incoming voice mail with speaker name for browsing and/or action (personal reply). For speech skimming or audio mining applications, recorded meetings or video with speaker labels are annotated for quick indexing and filing.

1.3. Challenges

Musical instrument classification is faces by several challenges which are described briefly as follows.

1) The classification of musical instruments has many problems due to multidimensional nature of musical instruments i.e. one instrument can belong to more than one instrument family.

2) A single musical instrument can be played in a variety of ways producing a variety of sounds. Humans categorizes sound or any quantity based on its characteristic but it becomes difficult for a common person to comprehend and categorize sounds belonging to same family of musical instruments. The machine learns from its train data and hence it is required to input large number of sound samples as train samples ranging in all variety for each instrument to the system in order to attain maximum accuracy. But how a particular instrument will be played, depends upon the artist and this set of variety is ever increasing. Also, the number of dataset available for research purposes is very limited.

3) In case of polyphonic music, it becomes even more difficult to segment and classify individual instruments playing in it and also identifying them. In this study, only solo recordings are considered. In future, instrument recognition from polyphonic music will be considered.
4) Musical instruments vary in shape, sizes, types, geographical locations, cultures, sound produced, playing style and so on. In order to create a musical information retrieval system or melody classifier or musical instrument indexer, it is needed to take all of these

instruments into account. These instruments will have one or more characteristic similar to each another and it will become extremely difficult to reach the goal.

Speaker recognition has also a number of challenges which are briefly listed below.

1) Speaker recognition suffers from problems like false match, false acceptance and false rejection. The human voice is subjected to intra-personal variations which leads to these errors. It is required to train a voice at different conditions for a considerable amount of time to notice it’s variations.

2) Closed set speaker identification does not deal with false acceptance and rejection. This problem comes into existence during open set speaker identification. Manual thresholding tends to produce these errors more as with increase in number of voices in the train dataset it requires to get changed. This manual process is tiresome and unreliable which poses a critical challenge in this field.

3) One of the most important challenges in speaker recognition stems from inconsistencies in the different types of audio and their quality. One such problem, is the problem of channel mismatch, in which the enrolment audio has been gathered using one apparatus and the test audio has been produced by a different channel. These variations often lead to errors.

4) There are also other factors such as, acoustical noise and variations in recording environments which hampers accuracy of the process.

1.4. Problem Statement

Classification of musical instruments and human speakers with audio features extracted with maximum accuracy and robustness is the goal.

.

1.5. Objectives

The objectives are as follows.

1) Study of existing techniques of audio classification and improve on them to address their limitations.

2) Finding the best feature for musical instrument classification as that will subsequently solve other problems.

3) For speaker recognition the focus of this study is to maximize the accuracy and undo human interference. of manually selecting threshold by designing a system with adaptive threshold robust to text spoken. The train and test samples are recorded via same medium.

1.6. Organization of the Thesis

Section 1 presents an introduction and overview of this thesis. Previous works are mentioned in Section 2. Section 3 presents the proposed approach. Section 4 and 5 presents experimental results and analysis, respectively. Section 6 is the conclusion.

2.LITERATURE SURVEY

The two subsets of sound recognition which are speaker recognition and musical instrument classification are active areas of research. Numerous measures have taken to best classify the musical instruments and speakers to their respective classes.

There have been a number of studies done to address musical instrument classification. Bormane and Dusane created a new classification technique for musical instruments by utilizing various techniques involved in wavelet transform for effective classification and for minimization of error. Bormane and Dusane’s classification technique utilizing wavelet transform produced results for 14 of 16 instruments. These instruments consisted of strings, brass, keyboard, and woodwinds. In their experiment, they had 10 testing notes. More so, for the classification of the cornet and tuba, they received 100% accuracy for tuba and 70% for cornet. In this paper, the problem of classifying of musical instruments is addressed. A new musical instrument classification method based on wavelet which represents both local and global information by computing wavelet coefficients at different frequency sub bands with different resolutions has been proposed. Using wavelet packet transform (WPT) along with advanced machine learning techniques, accuracy of music instrument classification has been significantly improved but since 14 out of 16 instruments are correctly determined hence unique features for specific instruments are needed to be found is the future scope of this work [1].

Park and Lee proposed a method utilizing convolutional neural networks. Furthermore, they have explored a new technique in classifying musical instruments using learned features from convolutional neural networks. CNN is the branch of deep learning and is used extensively in image classification. It is very powerful but it requires large amount of data which is often limited. Park and Lee’s method utilizing convolutional neural networks produced high performances in their confusion matrix. The experiment used 20 instruments including woodwinds, strings, brass, and piano. [2]

Therrick-Ari Anderson discusses a method for classifying musical instrument audio signals utilizing a neural network. This research identifies the most salient features to evaluate within a neural network that will quickly detect an instrument from another. He worked with approximately six audio features. They include dynamics, fluctuation, rhythm, spectral, timbre, and tonal. After extracting them, ANN was applied to classify two instruments Tuba and Trumpet.12 note samples were collected for B-flat pitch. The neural network was able to generate correct outputs for the majority of the trumpet samples resulting in 83% of samples producing the desired output. The accuracy was 66% for tuba. It is needed to deal with more of classes and enhance accuracy. [3]

Nagawade and Ratnaparke discusses a method where they used MFCC to extract the features of audio signals arising from musical instruments and used K-NN classifier to classify them into five classes which are Cello, Piano, Trumpet, Flute and Violin. The overall accuracy was more than 80% and they used 90 samples in train set and 60 for test set. The limitation as mentioned in the paper was, accuracy is prone to decrease with increase in number of instruments and instruments in polyphonic recordings will not be distinguished by this method. [4]

Dattatraya Kuralkar and Saurabh Deshmukh et.al presented a method for musical instrument identification with the use of audio descriptors. In the proposed work they used sound samples from five different flute instruments. Hence the study was dedicated to classify subsets of Flute. For the feature extraction MIR toolbox used. Then system was able to identify the sound of a particular flute instrument. [5]

R.S. Kothe, D.G. Bhalke, P.P. Gutal presented a model to detect and distinguish individual musical instrument using different feature schemes. The proposed method considers ten musical instruments. The feature extraction scheme consists of temporal, spectral, cepstral and wavelet features. They developed k-nearest neighbour model and support vector machine model to test the performance of system. The system achieves the 60.43 of recognition rate using k-nearest neighbour classifier with all features. A two-prong approach was taken to the multiclass classification which were SVM-one against rest& SVM-one vs. one. The accuracy of SVM in both cases is 73.73% with all features using radial basis function. Using weight factor method K-NN shows 73% accuracy while SVM shows 90.3% accuracy using exponential kernel function. [6]

Patil and Sanjekar proposed an algorithm for classification task in which they used SVM, MLP and AdaBoost for better result. The system was mainly designed for automatic classification of musical instrument using SVM, MLP and AdaBoost classifiers. Formal Concept Analysis technique was also applied to show relationship between musical instruments and their attributes. The system is evaluated with SVM, MLP and AdaBoost classifiers which show that AdaBoost gives better result than SVM and MLP. [7]

Sefki Kolozali, Mathieu Barthet, Gyorgy Fazekas, Mark Sandler et.al. presented preliminary work on ontology designed for musical instruments. The paper also provided the investigation of heterogeneity and limitations in existing instrument classification schemes. In this paper author implemented two instrument taxonomies based on H-S system in OWL [8].

The past two decades have witnessed researches focusing on speaker recognition and its development. However, at present speech recognition technology is more implemented than speaker recognition technology. F. Y. Leu and G. L. Lin in their work conducted feature matching by employing quantization techniques like Vector Quantization (VQ) and/or speaker modelling techniques like GMM to achieve desired classification of test data. The feature used in the work was MFCC [9].

N. M. Abo Elenein, K. M. Amin, M. Ibrahim and M. M. Hadhoud proposed that Speaker identification is accomplished by proper feature extraction, choosing suitable feature for the purpose followed by feature matching. Feature extraction deals with converting the speech data into acoustic vectors to facilitate feature matching. MFCCs are utilized as features in the study. The proposed approach involves feature extraction followed by gender classification followed by closed set speaker identification using GMM [10]

Amit Kumar Singh, Rohit Singh, Ashutosh Dwivedi stated in their paper that, Automatic Speaker Recognition still remains a confront mainly due to variations in speaker's vocal tract with time and health, varying environmental conditions, disparities in the behaviour and quality of speech recorders etc. In the paper MFCC was chosen as features and were extracted during feature extraction phase. A speaker database containing 30 male and 30 female speakers was used. Two separate experiments were conducted. In the first case the speech features were directly matched. In the second case a VQ codebook was created by clustering the training features, A distortion measure based on the minimum Euclidean distance was used for speaker recognition. The failure rate of speaker recognition in first case was found to be was found to be 10% while in the second case as found to be 14%. [11]

Reza Aulia Sadewa, Tokorda Agung Budi Wirayuda, Siti Saadah in their work came up with a solution for open set speaker identification. They used MFCC as features and those features were modelled by the VQ method. The method was modified with a proposed thresholding method to reject the unknown voice and a Butterworth Filter to handle the noise. MFCC and VQ combination truly distinguished 100% of the speakers in a closed sample which includes only the registered speaker. The proposed thresholding method was effective enough to reject the unknown voice with approximately 90% true rejection but produces only around 70% true acceptance. A threshold of each codebook was trained after the codebook of certain person or speaker had been generated. The threshold consists of minimum and maximum distortion value which was obtained by authenticating several voices of the same speaker. In the verification phase, the resulting lowest distortion was validated. The whole process described above requires human supervision. Finding balance between false rejection and acceptance was the future scope of this paper. [12]

Rawande Karadaghi, Heinz Hertlein and Aladdin Ariyaeeinia in their paper, presented an investigation regarding the relative effectiveness of two alternative approaches to open-set text-independent speaker identification (OSTI-SI). The methods considered are the recently introduced i-vector and the more traditional GMM-UBM method supported by score normalisation. The study was motivated by the growing need for effective extraction of intelligence and evidence from audio recordings in the fight against crime. In the study, the experimental investigations are conducted using a protocol developed for the identification task, based on the NIST speaker recognition evaluation corpus of 2008. In order to closely cover relevant conditions in the considered application areas and investigate the identification performance in such scenarios, the speech data was contaminated with a range of real-world noise. The paper provided a detailed description of the experimental study and presented a thorough analysis of the results. It had been observed that when the test was conducted on clean data 39.5% accuracy was obtained for GMM-UBM while 42.5% for GMM-UBM with TZ (test normalization, zero normalization) norm and 49.5% with i-vector method. The i-vector method has relation to the GMM-UBM technique as a single i-vector is said to be the consolidated representation of an adapted GMM. [13]

Fang-Yei Leu, Guan-Liang Lin in their work brought out that, speaker identification in its current stage is relatively immature compared to speech recognition. The speaker identification technique implemented in their paper first takes the original voice signals of a person, and then normalizes the audio energies of the signals. After that, it was converted from time domain to frequency domain by employing FFT.MFCC was used in feature extraction phase. Further, the probability density function of Gaussian mixture model was utilized to indicate the distribution of the quantified characteristics as a person’s specific acoustic model. The system was a closed set speaker identification model, modelled using GMM. [14]

D. A. Reynolds, R. C. Rose, and T. F. Quatieri, and R. B. Dunn pointed out that in recent years, studies indicate that the energy distribution of human voice signals follows a Gaussian Model which is why GMM is more dominant in the field of speaker identification. For the past twenty

years, GMM-UBM is identified as one of the major approaches, in field of speaker identification. They used GMM for the purpose of closed set identification of voice [15,16].
A. Brew and P. Cunningham, tried to design effective OSTI-SI system with cohort model and UBM based approach. It has been observed that when UBM based approach when unified with cohort model in a projective framework, improves accuracy. Although cohort model performs well in situation where unknown voices belongs to casual imposters but cohort-based speaker model becomes more vulnerable to attack by speakers sounding more similar to registered speakers [17].

F. Răstoceanu and M. Lazăr, worked with speaker verification a 1:1 match process, where the task was deciding, if the given speech utterance is provided by the hypothesized speaker S or not. The binary classifier can be formulated as follows, where T(x) is denoted as the test ratio (for speaker verification systems using GMM is the likelihood ratio) and η is the threshold value [18].

H0**:** x is from the hypothesized speaker.
H1**:** x is not from the hypothesized speaker.
The decision is given by,

$$T(x) = \frac{f(H0|x)}{f(H1|x)} \geq \eta, accept, T(x) = \frac{f(H0|x)}{f(H1|x)} \leq \eta, reject \tag{2.1}$$

The speaker verification is efficient for 1:1 match. But for 1: n match, the procedure needs improvements.

Farbod and Karthikeyan et.al proposed musical instrument classification using wavelet dependent time scale features. In this first continuous wavelet transform of the signal frame is taken and then features related to temporal variation and bandwidth are considered for feature extraction. [19]

Sumit Kumar, Banchhor and Arif Khan et.al proposed a method of musical instrument identification using short term energy and ZCR. For this analysis first signal is divided into frames and for each frame energy and ZCR is computed. ZCR is nothing but count of how many times signal changes the sign. [20]

M. E. Ozbek, N. Ozyurt, and F. A. Savaci et.al proposed a technique of instrument identification using wavelet ridges. For this first 3 level wavelet decomposition is performed This wavelet decomposition gives one approximate coefficient and three detailed coefficients. [21]

Slim Essid, Gael Richard, and Bertrand David et.al proposed use of MFCC features of sound samples. Delta MFCC features are used which are obtained by taking time derivative of MFCCs. Spectral features such as spectral centroid, spectral width etc. are also computed. For the classification SVM algorithm is used. In SVM one versus one mapping is used to create trained data. [22]

Priyanka S. Jadhav et.al proposed the use of MFCC and Timbral related audio Descriptors for the musical instrument identification. For feature classification k- nearest neighbour, support vector machine and binary tree are used. Identification accuracies for different combinations of features extraction and classification method are compared. The accuracy was found to decreased from 90% to 75% with increase in number of instruments from five to fifteen [23]

Meinard Müller, Daniel P. W. Ellis, Anssi Klapuriand Gaël Richardet.al presents paper on signal processing techniques for music analysis. This paper summarizes various research fields and the proposed woks in the field of music signal processing. The use of MFCCs for the musical instrument identification is also overviewed. So, this paper gives the idea about how choose the specific area for the research. [24]

James Bergstra, Norman Casagrande et.al. presented an algorithm that predicts musical genre and artist from an audio waveform in Aggregate Features and AdaBoost for Music Classification. They used the ensemble learner AdaBoost to select from a set of audio features that have been extracted from segmented audio and then aggregated which proved to be the most effective method for genre classification at the recent MIREX 2005 international contests in music information extraction, and the second-best method for recognizing artists [25]

Keith D. Martin and Youngmoo E. Kim present musical instrument identification: A pattern-recognition approach. In this paper they applied statistical pattern-recognition technique to the classification of musical instrument tones within a taxonomic hierarchy. They used data set which included examples from the string, woodwind, and brass families. Their experiments simulating results shows that fisher projection method resulted in successful classifiers at all levels of the taxonomy [26].

It was introduced to speaker recognition by Soong et al (1985). In speaker verification, Vector quantization (VQ) model were applied in Soong and Rosenberg [1], It is one of the simplest text-independent speaker models and usually used for computational speed-up techniques, it also provides competitive accuracy when combined with background model
adaptation [27] [28] [29] [30]

In 1966 L. CW Pols's research had given a way to results that suggested that the phonetically important characteristics of speech could be represented in a compact manner by a set of Mel-frequency Cepstrum Coefficients (MFFCs). In 1980 Davis and Mermelstein showed that a very high performance of the Mel Frequency Cepstrum coefficients is due to the reason that the perceptually useful and pertinent features of the short-term speech spectrum can be represented in a better manner as compared to a linear frequency Cepstrum or a linear prediction spectrum [31]

Davis and Mermelstein also gave a conclusion that the Cepstrum parameters (MFCC, LFCC and LPCC) which apply frequency smoothed representations of the log magnitude. The spectrum performs better than the LPC and reflect Coefficients (RC) in representing the significant acoustic data. Since MFCCs were found to be very efficient in speaker recognition and thus various other modifications of MFCCs based on the window band design, log compressed window bank output and the approximate models of the nonlinear pitch sensitivity of human were proposed by various researchers. Young et aI., 1995 described the HTK MFCC FB-24(they are computed through a window bank of 24 windows) [32]

Mukherjee H., Obaidullah S.M., Phadikar S., Roy K. in their work used Mel Scale Cepstral Coefficient (MFCC) based features coupled with a Multi-Layer Perceptron (MLP) based classifier has been used for categorization of 2716 clips from 7 different instruments obtaining an average accuracy of 98.38% accuracy. [37]

3.PROPOSED APPROACH

3.1. Motivation

A recognition problem deals with identification of the test data, when compared against the train data stored in train database, with which the system has been modelled. The intuition behind sound recognition is the same. In this study two special and most popular cases of sound recognition which are musical instrument classification and open set text independent speaker identification have been dealt with. The applications of musical instrument classification and OSTI-SI are many. Musical instrument classification aids in extraction of melody from music sound, recognition and separation of sound sources in polyphonic audio, recognition of musical instrument in the isolation, automatic music transcription, beat tracking, musical information retrieval and many more application of similar types while speaker identification can be used in biometric security devices like access control in a lab or forensic investigations and detecting ransom callers. The challenges faced by these two field of applications are also gruelling as for musical instrument classification accuracy falls with increase in number of instruments while speaker recognition also faces certain errors like false acceptance, false rejection and false matching. To design an effective robust system, these limitations must be minimized hence the goal of this study is to deal with these challenges and find an effectual solution to these problems so that these technologies can be applied in real world.

3.2. System Components

In order to accomplish the task of sound recognition, the system must have the following components.
1.Model Database
2.Feature Extractor Module
3.Classifier Module

<u>a)Model Database</u>

In this case, the model database is made of two sub databases. i)Train Database. ii)Test Database. In the train set there are audio samples from all classes. Required features have been extracted from this audio signals which represents the acoustic vectors and are trained, so that the system learns about individual classes based on the feature extracted.
The test set consists of sound samples, which is to be tested against the train data and the aim of the system is to label each of these test data and classify each of them to the correct class they belong to. The audio signals from test data also undergoes the process of feature extraction. The last part of the of any recognition system deals with classification, where test data is compared against train data and the required task is achieved.

<u>b)Feature Extractor Module.</u>

In this phase, the raw sound data collected from several sources are processed. Sound can be represented with many features. But all the features are not useful for all tasks. So, for a given system in design, features must be chosen wisely. In this phase, the raw sound samples from train set as well as test set undergoes sound processing module and are converted to some matrices or vectors. Each sample is represented by a matrix or vector of a certain length and that is the feature data for that particular sound. For example, if MFCC is required to be extracted, each vector of size [1× 1280] represents each sound sample. Again, the size 1280 depends on certain metrics chosen earlier like number of frames and number of overlapping frames taken during Windowing phase, number of centroids in the codebooks during Vector Quantization which is a process of compression of data, number of filter banks or coefficients etc. Typically, all sound samples undergo the process of Windowing and FFT after which for desired feature required steps are followed. These vectors achieved are called acoustic vectors and during classification phase, comparison is made, taking these as the representative mathematical entity of each sound sample.

c)Classifier Module

This module may be regarded as the 'decision' module of any recognition system. In this phase the representatives of train set and test set are compared against each other. Each sound sample from test set is compared against all the sound samples of train set and with which, the maximum similarity is achieved the sound sample is labelled to be representative of that class. In open set identification problem, where the sound might be assumed to have come from unknown sources, a threshold is employed against a metric and a sub-decision system is designed and accordingly a sound is classified as known and unknown.

3.3. Block Diagram

The block diagram of a general sound recognition system is given below.

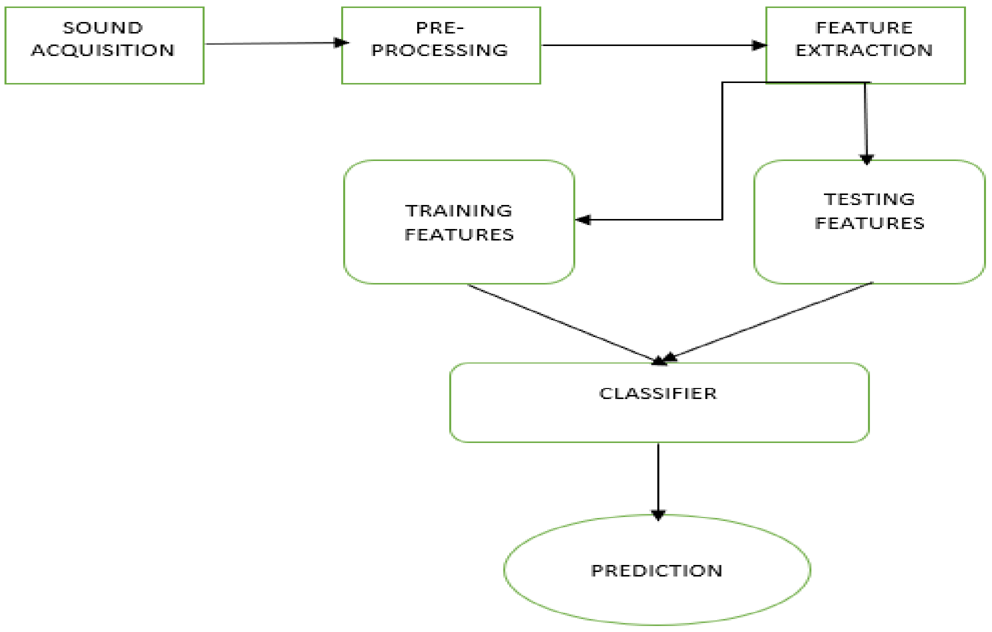

Figure 3.1:Generalised Block Diagram

In sound acquisition phase, care must be taken that the sound samples are collected from a distinguished online repository so that the mentioned characteristic of the sounds are well proofed.

If the sounds are required to be created, they must be created in a noise free environment with efficient microphones and must be ensured that train and test samples are both acquired by same or similar functioning device in order to undo mismatch due to change of channel medium.

In pre-processing phase, certain measures such as silence removal, combining many samples into one sample to reduce the number of train samples, yet not compromising the prediction accuracies etc are undertaken.

The feature extraction phase is the first crucial phase in the process of recognition of any object be it image, sound or video. In this phase, as explained earlier the processed raw data is transformed into some mathematical entities representing one or few features of each sound sample. In this case, each sound sample is represented by one feature that is again represented by a vector. The feature extraction phase is interceded by a sub module called feature detection in which, the best feature amongst all the features relevant and experimented in required cases of recognition is chosen to design the final version of the system. The decision is taken upon the factors like accuracy, robustness, desirability, reproducibility etc. In case of musical instrument classification, the sound samples acquired from five different instruments are experimented with six features which are relevant in this field of study, known as MFCC(Mel frequency cepstral coefficients), Spectral Centroid, HPCP(Harmonic Pitch Class Profile), LPC(Linear Predictive Coding) coefficients, Pitch Salience and Cepstral Coefficients(CC).It was found that out of all these prevalent features CC best classifies the instrumental sounds leading to least overlapping between classes providing maximum accuracy, reproducibility and robustness. For open set text independent speaker identification, the system was experimented with both CC and MFCC.CC gives good result in closed set classification but for open set, it is not suitable. The reasons will be discussed in later chapters. The likelihood obtained for test samples with train samples were random. Hence MFCC was chosen as the final feature for that problem.

The sound acquired either by recording or from an external database are divided into two databases. The train set and the test set. The train set contains one or more representatives from all classes and the system is trained on these samples so that the machine understands the difference between each class. The sound samples contained in test set are unknown to the machine or system and hence based on its learning it attempts to classify these samples into one of those classes learnt earlier. The test set can contain one or thousand voices as per requirement.

The classifier module or the decision module takes each sound sample from test set and compares it against all the sound samples present in train set. It then assigns the sound to the class with which it has maximum similarity. There are several machine learning and deep learning classification algorithm helpful for classification of binary or categorical data. In our study, for both cases experimentation has been carried out with K-NN (number of neighbours = 5), SVM(linear), Random Forest and ANN. The results of these experimentations will be shown in next chapter but it is to be mentioned here that for the first case ANN or Artificial Neural Network and in the second case, Random Forest is chosen as final classifier against the final features selected. The reason behind this selection will be discussed in subsequent chapters.

3.4. Architecture of The Musical Instrument Classification system

The musical instrument classification is an active field of research and has uses in extraction of melody from music sound, recognition and separation of sound sources in polyphonic audio, recognition of musical instrument in the isolation, automatic music transcription, beat tracking, musical information retrieval and many more application of similar types. In this study an extensive research has been carried on different features of instrumental sound and their contributions in this field. A number of machine learning classifiers have been applied in order to design a system having maximum accuracy and robustness.

3.4.1 Feature Extraction

In this module all the features chosen for this test case of musical instrument classification are discussed. There are many audio features. The audio features can be categorised in six sets of audio descriptors

1) Spectral descriptors: Bark Bands, Mel Bands, ERB Bands, MFCC, GFCC, LPC, HFC, Spectral Contrast, Spectral Centroid, Inharmonicity and Dissonance etc
2) Time-domain descriptors: Effective Duration, ZCR, Loudness etc
3) Tonal descriptors: Pitch Salience Function, HPCP, Tuning Frequency, Key, Chords Detection etc
4) Rhythm descriptors: Beat Tracker, Beat Tracker Multi-Feature, Histogram Descriptors Novelty Curve, Onset Detection etc.
5) SFX descriptors: Log Attack Time, Max to Total, Min to Total etc
6) High-level descriptors: Danceability, Dynamic Complexity, Fade Detection etc

This categorisation is made by Essentia which is an open-source library and tools for audio and music analysis, description and synthesis. From all these features, MFCC, LPC and Spectral Centroids are chosen from Spectral descriptors and Pitch Salience, HPCP is chosen from Tonal descriptors. The CC falls in the category of Spectral descriptors according to its characteristic nature.

a) MFCC(Mel-Frequency-Cepstral-Coefficients)

The first step in any sound recognition system is to extract features. MFCCs are useful in identifying the linguistic content and timbre of the sound and discarding the background noise, emotion etc. In case of musical instrument sound, the MFCC represents the envelope or timbre of the power spectrum of each frame of the sound.

MFCCs are commonly used in speech recognition and are finding increased use in music information recognition and genre classification systems and also speaker recognition applications. The MFCCs represents Timbre of the sound which is the property by which a listener distinguishes two sounds in spite of them being played with same pitch. This property is very powerful in classification of sound as it can distinguish two sounds based on the nature of their spectral envelope. It the fundamental nature of the sound from the spectrum and does not concentrate on pitch or other properties. The spectrum is then applied to triangular Mel-Filterbanks after which the logarithmic of Mel spectrum (as human response to signal level is logarithmic) is converted to coefficients using DCT. A Mel is a psychoacoustic unit of frequency which relates to human perception, the Mel scale can approximate from a Hz value. The equation is given below

Calculating MFCCs is performed as follows:

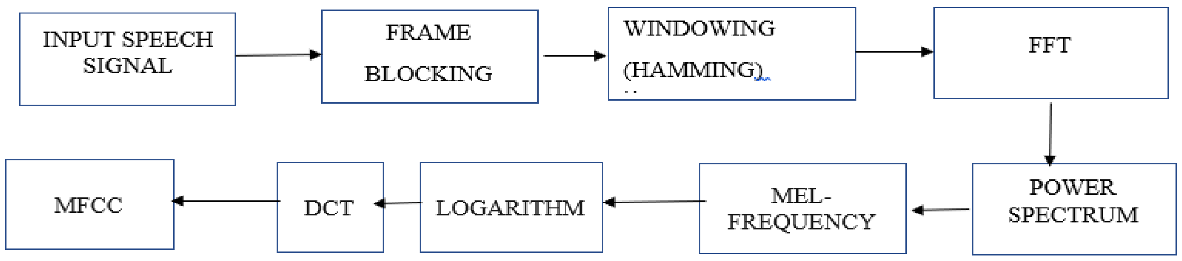

Figure 3.2: Block Diagram of MFCC

The steps followed in order to extract MFCC is briefly described below.

i) Framing:

An audio signal is constantly changing, so to simplify things it is assumed that on short time scales the audio signal doesn't change much (statistically i.e. statistically stationary). This is why the signal is framed into 20-40ms frames. If the frame is much shorter enough samples to get a reliable spectral estimate will not be there, if it is longer the signal changes too much throughout the frame. This frame length has been proposed in many papers and generally this is used. Adjacent frames are separated by M $(M < N)$ in a voice signal which is segmented into frames of N samples. Selected values for $N$and $M$are (N = 25ms.M = 10ms). The sample rate was 16000 samples per sec.

ii) Windowing:

After frame blocking, each frame is windowed with a Hamming window in order to taper the first and last points of the frames in order to reduce signal discontinuities. The main-lobe width is 4 bins and side-lobe level: -42.7 dB in case of Hamming window which is decent. In a typical case, the signal in a frame is denoted by (n), where $n \ = \{0, \dots, N-1\}$, and the signal after windowing is given by $s(n) * t(n)$, where $t(n)$ is the representation of Hamming window defined by (3.2)

$$t(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \ ; 0 \leq n \leq N-1 \tag{3.1}$$

iii) Discrete Fourier Transform:

The Fast Fourier Transform (DFT) transforms each frame of Nsamples from the time domain to the frequency domain as x represents the discrete time sequence while X represents the coefficients in frequency domain. The main idea behind FFT is to find out with which basis vectors or frequencies the questioned signal has most similarity. It is made out with similarity measure doing the dot product. $e^{j.\frac{2\pi ki}{N}}$ are the basis vectors. The basis vectors are the vectors which with the combination of certain coefficients can represent any signal or vector in a Euclidian or Hilbert space.

Hence the DFT expression is obtained as follows.

$$X_k = \sum_{i=0}^{N-1} x_i . e^{-j.\frac{2\pi k i}{N}} \tag{3.2}$$

iv) Power Spectrum:

The next step is to calculate the power spectrum of each frame. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on the location in the cochlea that vibrates (which wobbles small hairs), different nerves fire informing the brain that certain frequencies are present. Periodogram estimate performs a similar job for us, identifying which frequencies are present in the frame.

$$P_i(k) = \frac{1}{N} |X_k|^2 \tag{3.3}$$

v) Mel Frequency Wrapping:

Human perception of the contents of frequency of sound for speech signals does not follow a linear scale. This fact has been proven from psychological studies. For each tone with an actual frequency f measured in Hz, a subjective pitch is measured on a scale called the 'mel' scale. This *Mel-frequency* scale follows a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz as given by the following formula:

$$M = 2595. \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.4}$$

The number of *Mel* spectrum coefficients, *K*, is generally chosen to be 20. This filter bank is applied in the frequency domain, which is equivalent to applying the triangle-shape windows to the spectrum. To smooth the magnitude spectrum and to minimize the size of the features are two major reasons of using triangular bandpass filters.
It is inferred from Mel-scale exactly how to space filter banks and how wide to make them.

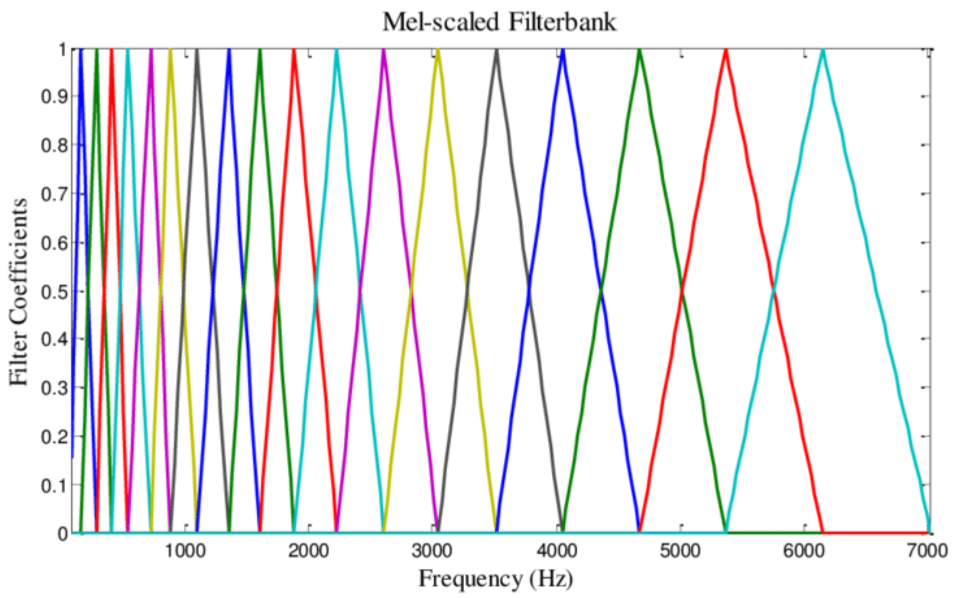

Figure 3.3: Mel Filter-bank

The Mel scale approximates the frequency resolution of the auditory system. It relates the perceived frequency of each of a pure tone to its actual measured frequency. Humans, are much better at discerning small changes in pitch at low frequencies than at high frequencies. So, incorporating the scale, is actually making the spectral features much closer to what we actually hear. So, in this diagram the horizontal axis is the linear scale. And then, on the vertical axis is the new scale. This modified frequency scales. It puts more emphasis on low frequencies and less emphasis on high frequencies. So, this is the redistribution of the frequency components and the energies of the frequency components. The following graph is derived from the equation 3.4.

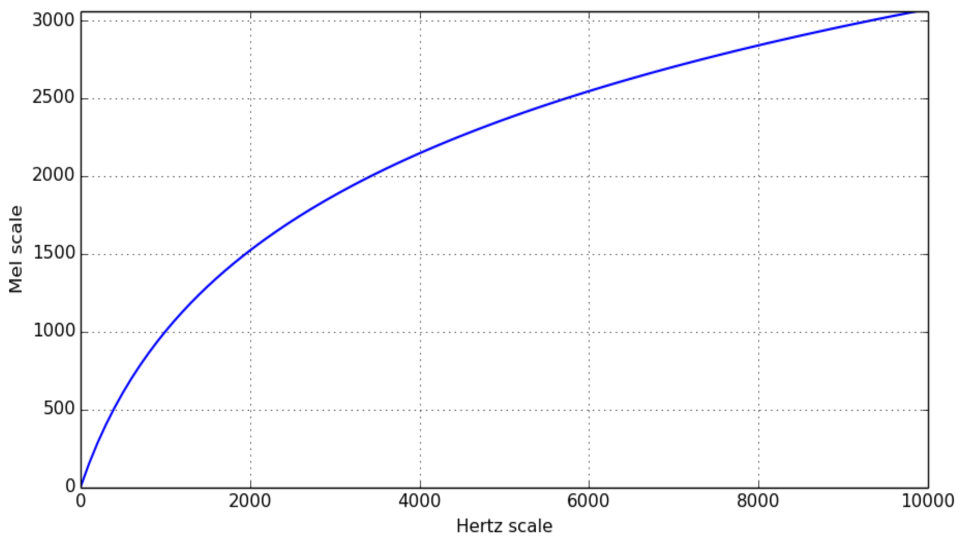

Figure 3.4: Mel scale

vi) Logarithm of Mel-Spectrum:

From filter bank energies, the logarithm is taken. This is also motivated by human hearing as loudness is not heard on a linear scale by us. Generally, to double the perceived volume of a sound it is needed to put 8 times as much energy into it. This means that large variations in energy may not sound all that different if the sound is loud to begin with. This compression operation makes the features match closer to what humans actually hear. The logarithm allows us to use cepstral mean subtraction, which is a channel normalisation technique. The logarithm also removes the multiplicative effect into additive effect thus increasing simplicity.

vii) Discrete Cosine Transform:

The final step is to compute the DCT of the log-filterbank energies. There are 2 main reasons for this. As the filterbanks are all overlapping, the filterbank energies are quite correlated with each other. The DCT decorrelates the energies which means diagonal covariance matrices can be used to model the features in e.g. an HMM classifier. But only 12 of the 26 DCT coefficients are kept. This is because the higher DCT coefficients represent fast changes in the filterbank energies and it turns out that these fast changes actually degrade the system performance.
So, each input utterance is transformed into a sequence of the acoustic vector. Forward DCT is defined by the following where $\propto$ is a constant dependent on N

$$X_k = \propto . \sum_{i=0}^{N-1} x_i . \cos\left\{\frac{(2i+1)\pi k}{2N}\right\} \tag{3.5}$$

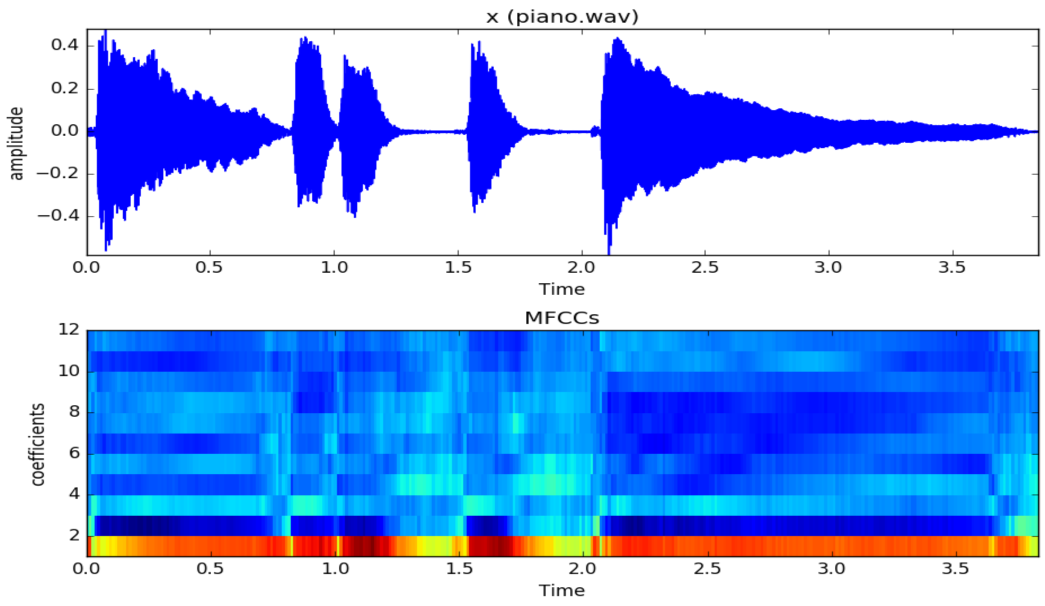

Figure 3.5: Representation of MFCC of a piano

The above is the visualization of the MFCC analysis on piano sound. In fact, the issue is that every coefficient is a representation of a different level of obstruction of the spectral shape. So here, for example, first 12 coefficients are plotted. And in fact, the zero coefficient is not shown as that relates to the loudness or to the energy of the sound. The first coefficient is the one that describes the bigger picture of the spectrum, and towards higher end, more small changes in the spectrum are described. And so, this is normally used as a vector including all these coefficients at every frame. It is a very compact representation of the characteristics of sound.

The reason for experimenting with this feature in this study are the follows:

1) It is widely used to represent sound features and are used by many in identification and speech processing problems
2) It mimics human hearing system and hence represents the features caught by human ear while hearing. Human ear can classify sounds based on those features, hence this is a powerful feature for identification problems.
3) It is a timbre related feature and hence is used in this study

<u>b) LPC (Linear Predictive Coding) coefficients</u>

A common approach for obtaining a filter that approximates spectral characteristics of a sound is to obtain LPC coefficients. The LPC coefficients gives a set of filtered coefficients(a_k), and the frequency response of the resulting filter approximates spectrum of the input sound. So, a signal can be approximated by LPC model as the linear combination of past samples.

$$\hat{y}[n] = \sum_{k=1}^{K} a_k y[n-k] \tag{3.6}$$

This is basically the expression of IR filter (infinite response filter) that is a linear combination of previous samples. The goal of LPC is to find the coefficients that best approximates the signal in question. The predictor coefficients are determined by(over a finite interval)minimizing the sum of squared differences between the actual speech samples and the linearly predicted ones.The error measure, aids to identify the coefficient that minimizes the error signal. $\hat{y}[n]$ is the predicted signal while a_k are the estimated coefficients and y[n] is the original signal while K is the number of coefficients required for faithful estimation of the original signal and n is the discrete time sequence of the signal. In this study K is chosen as 20.

$$\text{Error} = \sum_{n=-\infty}^{n=+\infty} (y[n] - \hat{y}[n]\,)^2 \tag{3.7}$$

LPC works quite well for few types of signals but does not work so well for many other types of signals. The autocorrelation function in general is given by,

$$Z\,[\,\text{k}] = \sum_{n=N-1}^{n=0} x[n]x[n+k] \;\; for\; k = -N+1\,, \dots.\,, N-1 \tag{3.8}$$

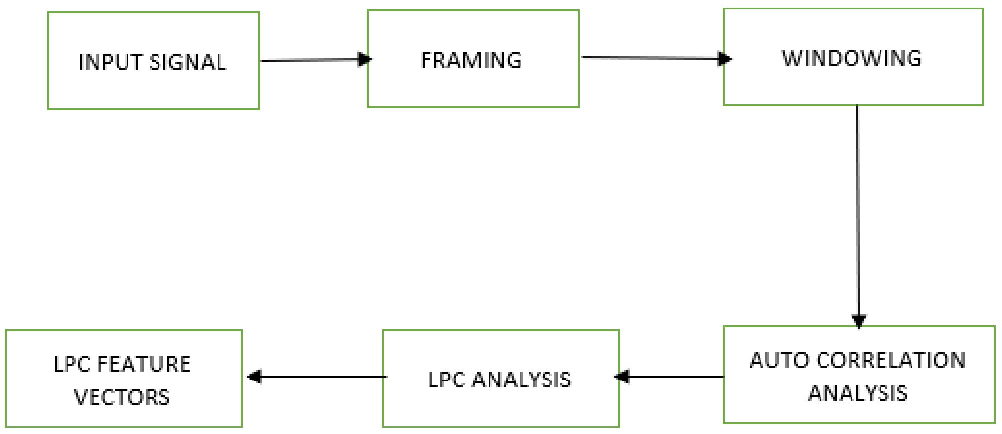

Figure 3.6: Block Diagram of Extraction of LPC coefficients

i) Framing:

The input signal at first is framed and then windowed with the pre-defined values on M = 10ms and N = 25ms with M<N and after many experiments conducted these values are generally chosen.

ii) Windowing:

The window in the case was hamming window for the reasons for which has already been described during explanation of MFCC.

iii) Auto-Correlation Function:

After that an autocorrelation function was applied that is responsible to represent a stochastic signal and it is a measure of degree of repetitions in a signal. After that LPC analysis is done based on Levinson -Durbin method, a method used to calculate LPC coefficients feature using recursive procedure.

iv) LPC Analysis:

For a discrete time second-order stationary process, the Levinson–Durbin recursion is used to determine the coefficients of the best linear predictor of the observation at time(k+1), given k previous observations, best in the sense of minimizing the mean square error. The coefficients determined by the recursion define a Levinson–Durbin sequence.

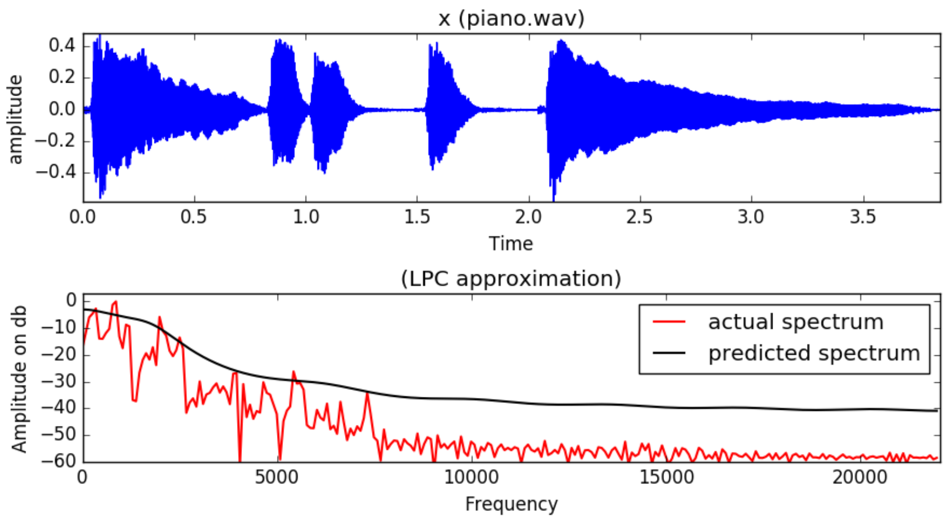

Figure 3.7: LPC Approximation of Piano

The above is the visualization of the LPC prediction on piano sound. In the second subplot the red line shows actual spectrum of the signal while the back line shows estimated spectrum or predicted spectrum based on LPC coefficients. The first sub. plot is the time domain representation of the signal. It is a very compact representation of the characteristics of sound.

The reason for experimenting this feature in this study are the follows:

1) It is widely used to represent sound features and are used by many in identification and speech processing problems
2) It is a prediction of actual spectrum of the sound, therefore it is the prediction of actual sound and hence it actually predicts the frequency components or basis vectors with which the signal has maximum similarity.
3) It is a timbre related feature and hence is used in this study.

c) Spectral Centroid

The centroid, of the frequency spectrum, is equivalent of a human perception of ‘brightness’. It is derived by multiplying the value of each frequency by its magnitude in the spectrum, then taking the sum of all these and again dividing the whole numerator by the magnitude in the spectrum of the signal. Centroid is the descriptor feature that tries to characterize the spectral shape of a particular sound. This is the spectral centroid, which indicates where the center mass of the spectrum is. Perceptually, it is very much related with impression of brightness of sound. It is a measure of how sound varies in time and how the brightness of that sound changes in time. The following equation explains the above description. Number of centroids used in this study is 50.

$$\text{Centroid} = \frac{\sum_{k=0}^{N/2} k|X_l[k]|}{\sum_{k=0}^{N/2} |X_l[k]|} \tag{3.9}$$

k = frequency, $X_l[k]$ = magnitude in the spectrum

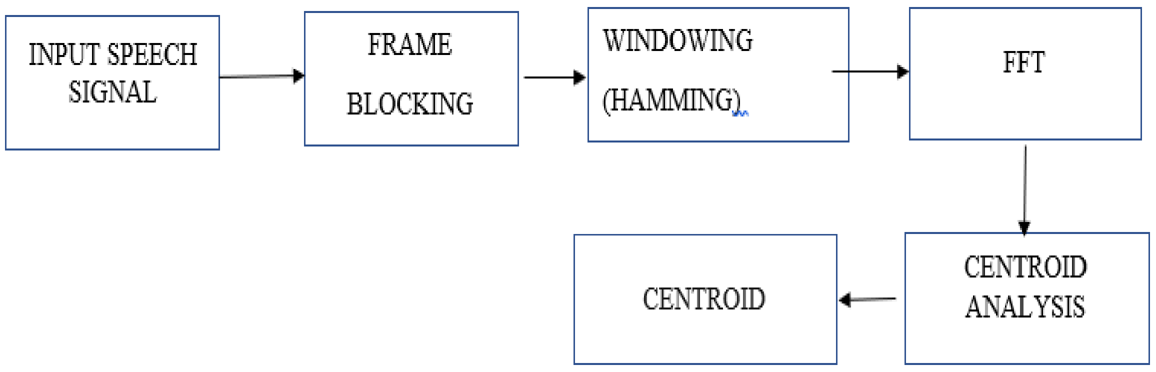

Figure 3.8: Block Diagram of Centroid Extraction

i) <u>Framing:</u>

The input signal at first is framed and then windowed with M = 1024 and N = 512 with M<N. In many similar works conducted these values are generally chosen. These are optimum values and decided after some trial and errors.

ii) <u>Windowing:</u>

The window in the case was hamming window for the reasons for which has already been described during explanation of MFCC.

iii) <u>Discrete Fourier Transform:</u>

The Fast Fourier Transform (DFT) transforms each frame of Nsamples from the time domain to the frequency domain as x represents the discrete time sequence while X represents the coefficients in frequency domain. The main idea behind FFT is to find out with which basis vectors or frequencies the questioned signal has most similarity. It is made out with similarity measure doing the dot product. $e^{j.\frac{2\pi ki}{N}}$ are the basis vectors. The basis vectors are the vectors which with the combination of certain coefficients can represent any signal or vector in a Euclidian or Hilbert space.

iv) <u>Centroid Analysis:</u>

In this phase the equation (3.9) is applied to bring out the centroid of the audio signal.

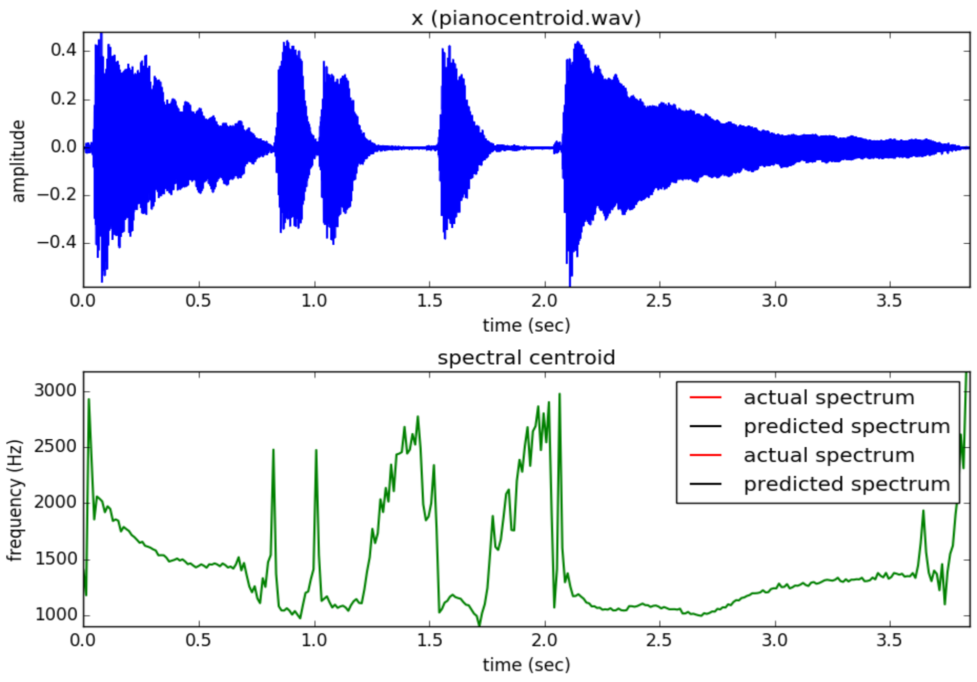

Figure 3.9: Spectral Centroid of Piano

So, the first part is the sound of piano and a time domain representation of it. The spectral centroids are plotted below. It represents the center point of the spectrum as it changes in time. So, this is a good measure and can be used to characterize quite a few aspects of the sound and in particular ‘brightness’ of the sound.

The reason for studying this feature in this study are the follows:

1) This is not a widely used feature for identification problems but the characteristics of sound which this brings out can be useful.
2) It measures the brightness of the sound while varies for different classes
3) It also describes the spectrum of sound like above two and hence can prove useful in identification problems.

d) Pitch Salience Function

Pitch salience is a measure of the presence of a pitch sounds in signal. It is useful to find the possible harmonics that are present in a particular peak. Of the many methods proposed for melody extraction the largest group are the salience-based methods, which derive an estimation of pitch salience over time and then apply tracking or transition rules to extract the melody line without separating it from the rest of the audio. Such systems follow a common structure—first the spectrum of the signal is obtained. The spectral representation is used to compute a salience function (time–frequency representation of pitch salience). The peaks of the salience function are considered as potential F0 candidates for the main melody. Different approaches exist for computing the salience function.

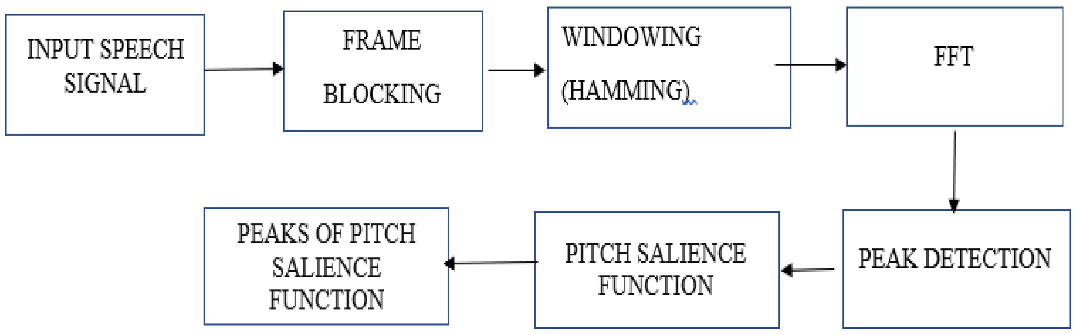

Figure 3.10: Block Diagram for Extraction of Pitch Salience Function Peaks

i) Framing:

The input signal at first is framed and then windowed with M = 1024 and N = 512 with M<N. In many similar works conducted these values are generally chosen. These are optimum values and decided after some trial and errors.

ii) Windowing:

The window in the case was hamming window for the reasons for which has already been described during explanation of MFCC.

iii) Discrete Fourier Transform:

The Fast Fourier Transform (DFT) transforms each frame of Nsamples from the time domain to the frequency domain as x represents the discrete time sequence while X represents the coefficients in frequency domain. The main idea behind FFT is to find out with which basis vectors or frequencies the questioned signal has most similarity. It is made out with similarity measure doing the dot product. $e^{j.\frac{2\pi ki}{N}}$ are the basis vectors. The basis vectors are the vectors which with the combination of certain coefficients can represent any signal or vector in a Euclidian or Hilbert space.

iv) Peak Detection:

This step detects local maxima (peaks) in an array. The algorithm finds positive slopes and detects a peak when the slope changes sign and the peak is above the threshold. It optionally interpolates using parabolic curve fitting.

v) Pitch Salience Function:

The salience function is given by,

$$S[b] = \sum_{h=1}^{H} \sum_{p=1}^{P} e(Ap)g(b\ , h\ , f\ p)(Ap)^{\beta} \tag{3.10}$$

where
S[b]=salience at bin frequency b (b expressed in cent scale)
e()=magnitude threshold function
g()=weighting function applied to peak p
β= magnitude compression value
A_p = Amplitude of Peak
P = Number of peaks
H = Number of harmonics
The extracted spectral peaks are used to construct a salience function—a representation of pitch salience over time. The peaks of this function form the F0 candidates for the main melody. The salience computation is based on harmonic summation and the salience of a given frequency is computed as the sum of the weighted energies found at integer multiples (harmonics) of that frequency. The important factors affecting the salience computation are the number of harmonics considered and the weighting scheme used. The idea of peak salience normally relates to how much of a peak is present at a particular frame.

vi) Peaks of Pitch Salience Function:

The peaks are obtained from pitch salience function. This step computes the peaks of a given pitch salience function. This algorithm is intended to receive it’s input from the Pitch Salience Function. The peaks are detected using Peak Detection algorithm. The outputs are two arrays of bin numbers and salience values corresponding to the peaks.50 peaks are taken in this study

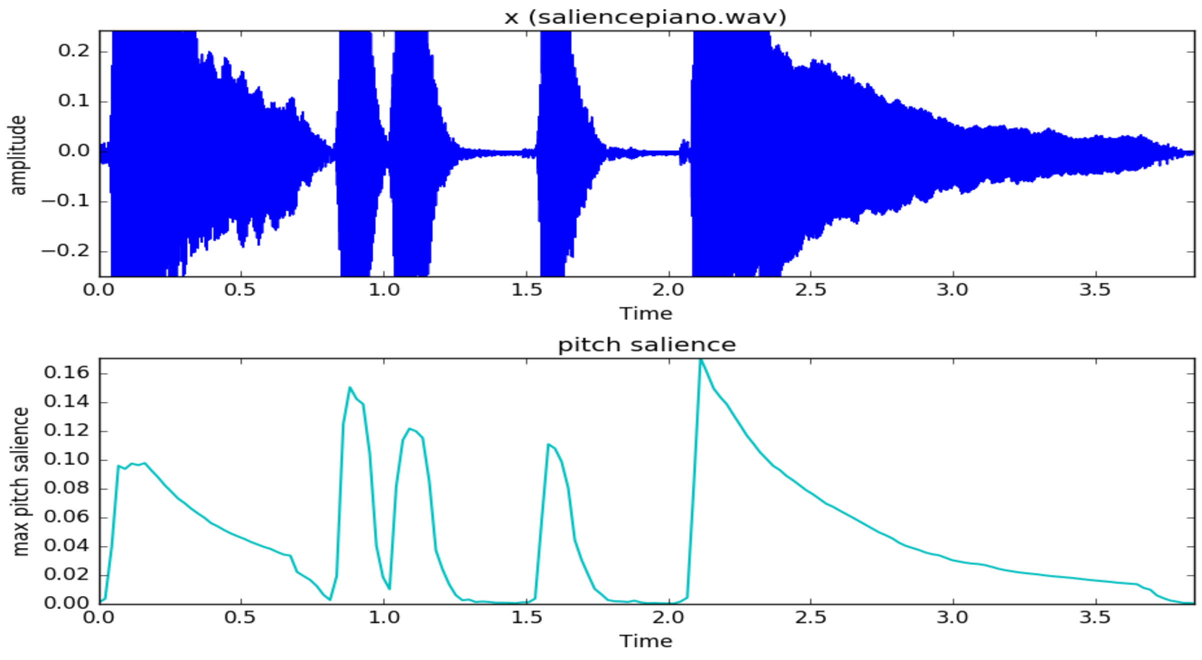

Figure 3.11: Pitch Salience Function of Piano

The above picture is from a piano sound. So, by looking at this function, visualization and estimation of the presence of the pitch sounds in every frame is possible and that can be quite useful to characterize quite a number of sounds. Pitch salience is a measure of the presence of a pitch sounds in signal. It tries to find the possible harmonics that are present of a particular peak. It is useful in extraction of melody of a sound. The peaks of the salience function are considered as potential F0 or fundamental frequency candidates for the melody.

This feature is used in this study for experimentation are for the following reasons:

1) It is a pitch related feature. Although accuracy falls when different sounds of similar pitch are used yet it is useful in characterization of different sounds also differing in pitch.
2) The potential candidates of fundamental frequency for the main melody will differ in different sounds derived from different instruments.
3) It is useful in melody extraction and pitch related distinction of different instrumental sounds

e) Chroma (Harmonic Pitch Class Profile)

HPCP is a vector representing the intensities of the twelve semitone pitch classes (corresponding to notes from A to G#), or subdivisions of these.

It is a group of features extracted from an audio signal, based on a pitch class profile which is a descriptor proposed in the context of a chord recognition system. HPCP are an enhanced pitch distribution feature that are sequences of feature vectors that, to a certain extent, describe tonality, measuring the relative intensity of each of the 12 pitch classes of the equal-tempered scale within an analysis frame. Often, the twelve pitch attributes are also referred to as chroma and the HPCP features are closely related to chroma features or Chroma grams.

HPCP features are used to estimate the key of a piece, measuring similarity between two musical pieces, in extracting the musical structure and to classify music in terms of composer, genre or mood. The process is related to time-frequency analysis. In general, chroma features are robust to noise (e.g., ambient noise or percussive sounds), independent of timbre and instrumentation and independent of loudness and dynamics.

.

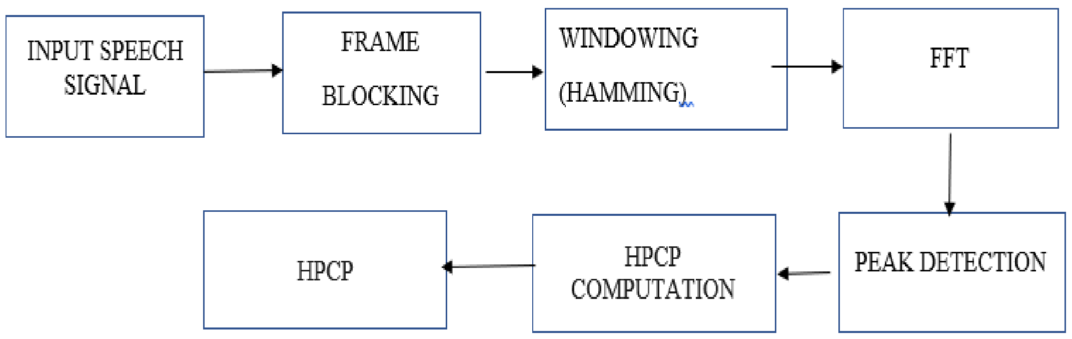

Figure 3.12: Block Diagram for computation of HPCP

i) <u>Framing:</u>

The input signal at first is framed and then windowed with M = 1024 and N = 512 with M<N. In many similar works conducted these values are generally chosen. These are optimum values and decided after some trial and errors.

ii) <u>Windowing:</u>

The window in the case was hamming window for the reasons for which has already been described during explanation of MFCC.

iii) <u>Discrete Fourier Transform:</u>

The Fast Fourier Transform (DFT) transforms each frame of Nsamples from the time domain to the frequency domain as x represents the discrete time sequence while X represents the coefficients in frequency domain. The main idea behind FFT is to find out with which basis vectors or frequencies the questioned signal has most similarity. It is made out with similarity measure doing the dot product. $e^{j.\frac{2\pi ki}{N}}$ are the basis vectors. The basis vectors are the vectors which with the combination of certain coefficients can represent any signal or vector in a Euclidian or Hilbert space.

iv) <u>Peak Detection:</u>

This step detects local maxima (peaks) in an array. The algorithm finds positive slopes and detects a peak when the slope changes sign and the peak is above the threshold. It optionally interpolates using parabolic curve fitting.

v) <u>HPCP Computation.</u>

This is explained with the equation as follows

$$\text{HPCP}[k] = \sum_{p=1}^{P} w(k, f_p) {A_p}^2 \tag{3.11}$$

Ap=amplitude of spectral peak p
P = total number of peaks
w (k, f p) =weight of the peak frequency fp for bin k
k=spectral bin locations of the chosen HPCP frequencies
HPCP is the sum of weighted square of amplitudes of peaks along all peaks.12 HPCP filters are taken to represent a sound sample.

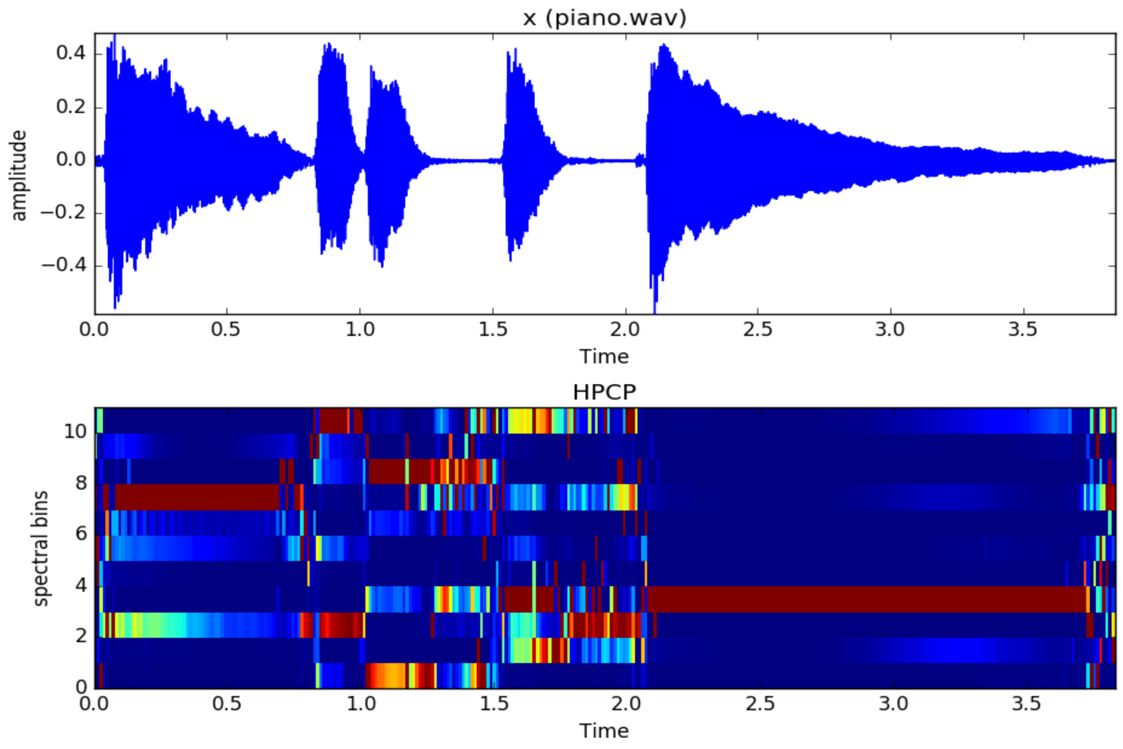

Figure 3.13: HPCP of Piano

The above picture contains the HPCP plot of piano. It can be observed here that the greater portion of the sound lies within the 2 and 4 pitch classes of 12 pitch classes. The first subplot is the time domain representation of the piano.

The reason for choosing this feature for experimentation in this study are the following:

1) This feature is independent of timbre and is related to the pitch of the sound. If two dissimilar sounds are played in different semitones then this feature can easily distinguish them.
2) The pitch salience and HPCP are studied as they can be implemented in special areas where it is needed to distinguish two varieties of same instruments differing mostly in pitch having similar timbre.
3) The pitch related features are less used in classification than timbre related features yet the pros and cons of these feature are studied to reach a conclusion.

f) Cepstral Coefficients

The Cepstral Coefficients is studied with the aim to increase the accuracy of the system in case of limited availability of data. The power spectrum for each frame is taken like MFCC to identify frequencies present in a frame that works like human cochlea. The Mel filterbanks are not applied next as they are more useful in automatic speech recognition and they represent the phonemes produced in the sound. The Mel filterbank causes some spectral obstruction and hence a model in derived similar to human system. In particular the cochlea cannot discern the difference between two closely spaced frequencies. This effect becomes more pronounced as the frequencies increase. For this reason, clumps of periodogram bins are taken and summed up to get an idea of how much energy exists in various frequency regions. This is performed by the Mel filterbank and since the human ear is more sensitive to low frequencies than high frequencies and the Mel scale is designed likewise. So instead of using a system that models human ear, a different system that actually takes all the information contained in periodogram estimate and pays equal attention to low and high frequency components is used to make a system that machines can process in order to bring out a feature containing more information about the source of sound and hence increasing it's potential for identification task. This feature is useful for extraction of Timbre and capable of processing it in a way that aids classification problem. The logarithm of power spectrum is taken so that the DCT computation is simplified from multiplicative to additive operation also higher the values more better is its representation. The number of coefficients is limited to 20 to 40 as higher than that degrades system performance, takes more computational time. In this study 40 coefficients of CC are taken. The block diagram of the extraction of this feature is given as follows

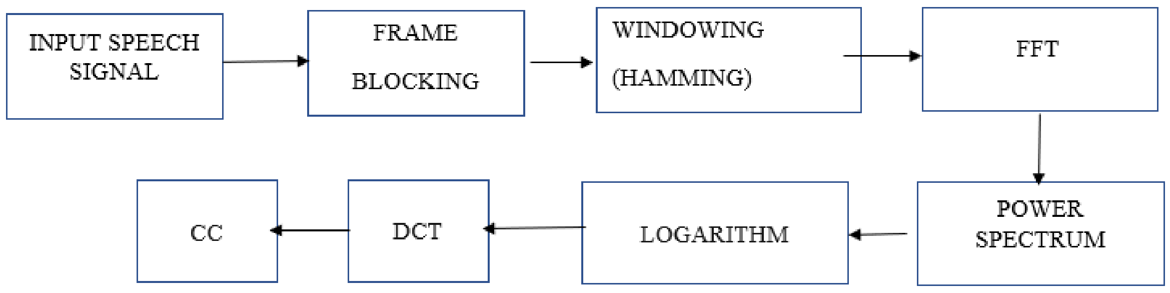

Figure 3.14: Block Diagram for Extraction Of CC

i) Framing:

An audio signal is constantly changing, so to simplify things it is assumed that on short time scales the audio signal doesn't change much (statistically i.e. statistically stationary). This is why the signal is framed into 20-40ms frames. If the frame is much shorter enough samples to get a reliable spectral estimate will not be there, if it is longer the signal changes too much throughout the frame. This frame length has been proposed in many papers for MFCC and generally this is used. In this study after some trial and error this decision is taken. Adjacent frames are separated by M $(M < N)$ in a voice signal which is segmented into frames of N samples. Selected values for $N$and $M$are (N = 25ms.M = 10ms). The sample rate was 16000 samples per sec.

ii) Windowing:

After frame blocking, each frame is windowed with a Hamming window in order to taper the first and last points of the frames to reduce signal discontinuities. The main-lobe width is 4 bins and side-lobe level: -42.7 dB in case of Hamming window which is decent. Hamming window defined by equation (3.1)

iii) Discrete Fourier Transform:

The Fast Fourier Transform (DFT) transforms each frame of Nsamples from the time domain to the frequency domain. X represents the discrete time sequence while X represents the coefficients in frequency domain. The main idea behind FFT is to find out with which basis vectors or frequencies the questioned signal has most similarity. It is made out with similarity measure doing the dot product. $e^{j.\frac{2\pi ki}{N}}$ are the basis vectors. The basis vectors are the vectors which with the combination of certain coefficients can represent any signal in a Euclidian or Hilbert space. The DFT is expressed in equation (3.2).

iv) Power Spectrum:

The next step is to calculate the power spectrum of each frame. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on the location in the cochlea that vibrates (which wobbles small hairs), different nerves fire informing the brain that certain frequencies are present. Periodogram estimate performs a similar job for us, identifying which frequencies are present in the frame. The equation (3.3) describes it.

v) Logarithm of Power Spectrum:

From filter bank energies, the logarithm is taken. This is also motivated by human hearing as loudness is not heard on a linear scale by us. Generally, to double the perceived volume of a sound it is needed to put 8 times as much energy into it. This means that large variations in energy may not sound all that different if the sound is loud to begin with. This compression operation makes the features match closer to what humans actually hear. The logarithm allows us to use cepstral mean subtraction, which is a channel normalisation technique. The logarithm also removes the multiplicative effect into additive effect thus increasing simplicity.

vi) Discrete Cosine Transform:

The final step is to compute the DCT of the log-filterbank energies. There are 2 main reasons for this. As the filterbanks are all overlapping, the filterbank energies are quite correlated with each other. The DCT decorrelates the energies which means diagonal covariance matrices can be used to model the features in e.g. an HMM classifier. So, each input utterance is transformed into a sequence of the acoustic vector. It is expressed with equation (3.4).
At the end of this step,40 CC coefficients are obtained.The whole process can be defined with an equation as follows:

$$CC = DCT(\log(P_i(k))) \qquad (3.12)$$

Where ($P_i(k)$) is the power spectrum of the signal defined in equation (3.3)

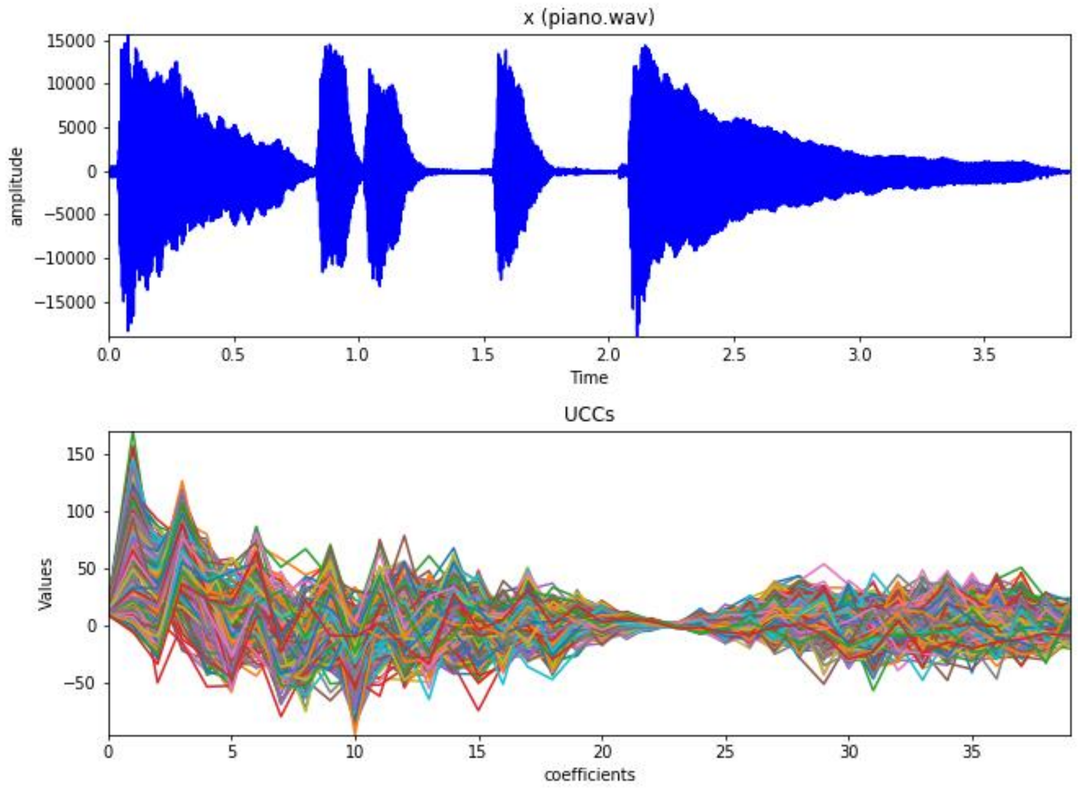

Figure 3.15: CC coefficients of Piano

The above picture is the picture of 40 coefficients and their values for the piano signal those time domain representation is given in above sub plot. The value varies between 150 to around -100.

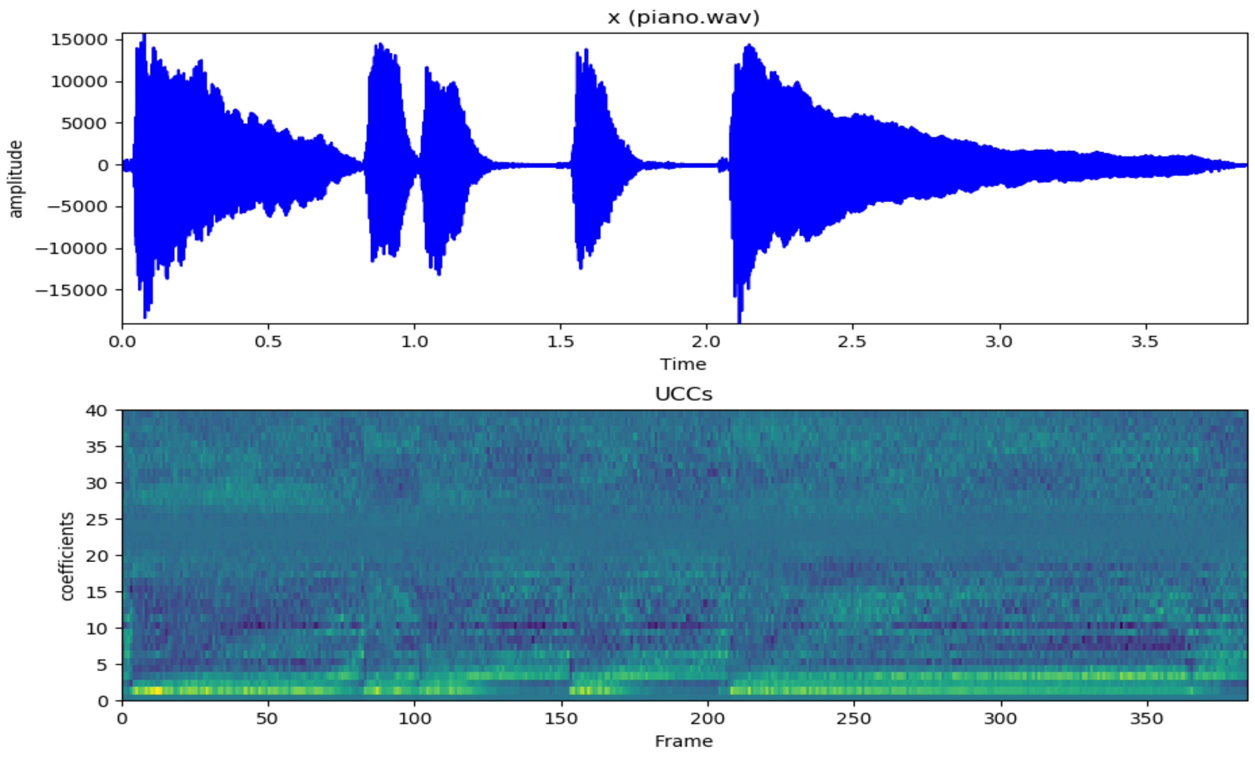

Figure 3.16: CC coefficients with respect to frames of Piano sound

In the above picture the variation of cepstral coefficients up to frame length 350 is shown. It can be observed that both higher and lower frequency components of the sound is gathered. Unlike MFCC more emphasis is not paid on lower frequencies, equal emphasis is paid to both of them.

The reasons for using this feature in this study are as follows:

1) This is an improved version of MFCC as MFCC was made specially for Speech recognition problems as the phonemes uttered by human are better captured by application of Mel scale which gives more emphasis to areas of lower frequencies. This feature can be used in generalised sound recognition problems.
2) The CC represents the Timbre of the sound by the envelope of the spectrum. Again, the short time power spectrum enhances the process of Timbre extraction from the sound.
3) The Timbre is the property which works independent of tone or pitch to identify sounds. Hence correct representation of Timbre will lead to better identification.

For these reasons this feature is used in this study. Since in the extraction process no filter is applied and at the end cepstral coefficients are extracted hence the name Unfiltered Cepstral Coefficients.

3.4.2 Feature Selection

Amongst all the features CC is chosen for the final system design for the following reasons

1) The accuracy observed for this feature compared to others are much better.
2) The overlapping between different sound samples belonging to different instruments are least in this case and lesser than even MFCC. This point will be discussed extensively in next chapter with relevant plots.
3) This feature brings out the timbre of the sound which distinguishes sound irrespective of pitch played.
4) Dissimilar instruments played in same pitch often is identified as same by pitch related features such as HPCP and Pitch Salience hence they are not chosen as the best one for they are case specific. They are useful where pitch of each instrument differs from one another.
5) The MFCC is also another best tool for musical instrument classification but the rate of decrease of accuracy in MFCC is more than that of CC for it suffers some overlapping between different classes for although it represents timbre yet due to application of Mel scale focuses more on areas low frequencies than high frequencies.
6) The LPC coefficients has much lower accuracy compared to MFCC and CC for the prediction of spectrum of sound is bound to suffer lower classification rates compared to ones which represents the actual spectrum of sound in one or other way.
7) The Spectral Centroids suffers limitations with increase in number of instruments as brightness is not such a unique feature to categorise sound efficiently in complex situations. It works well in cases of less number of classes.

3.4.3. Compression of Data

This step although a minor step yet is very useful and required for a efficient system. The features like MFCC, LPC coefficients, CC and HPCP are very high dimensional. To reduce their dimensions yet not compromising system’s accuracy lossy compression technique like Vector quantization is used.

a) Vector Quantization

Vector Quantization (VQ) is a data reduction method. It is useful for reduction of dimension so as to reduce redundancy of data. It may be regarded as a process by which vectors from a large vector space is consolidated into the limited number of regions present in that space. The centre of each region so obtained (known as clusters) is called a codeword. A codebook is the collection of all codewords of the vectors. VQ technique is implemented through LBG algorithm. Feature vectors of both train and test data extracted from MFCC is applied to VQ.

b) Flattening:

In order to input the data in several machine learning algorithm the data of MFCC, LPC, CC and HPCP is converted to a vector. At first, they were three dimensional matrices and then they were flattened to a vector.
The number of Centroids used for MFCC, CC and LPC is 64 while for HPCP it is 32. The vector size of MFCC is [1×1280] for there are 64 centroids and 20 coefficients. In general, for a single sound there is a [20×64] matrix which after flattening [1×1280] is obtained for a single sound sample. For CC the vector size is is [1×2560] for there are 64 centroids and 40 coefficients. In general, for a single sound there is a [40×64] matrix which after flattening [1×2560] is obtained for a single sound sample. The vector size of LPC is [1×1280] for there are 64 centroids and 20 coefficients. In general, for a single sound there is a [20×64] matrix which after flattening [1×1280] is obtained for a single sound sample. For HPCP the vector size is [1×384] for there are 32 centroids and 12 coefficients. In general, for a single sound there is a [12×32] matrix which after flattening [1×384] is obtained for a single sound sample.

3.4.4 Classification

This is a vital step in any recognition problem. In this step the features from the test data and train data are compared against each other to measure similarity in order to achieve classification. There are many classifiers present but the choice of most effective one for a specific problem which helps in achieving the goal of the system, has better accuracy compared to others and prevents overfitting along with producing reproducibility is a complex task involving many experiments and observations. In this study four major and most prevalent machine learning classifiers are used which are SVM(linear), K-NN (number of neighbours = 5), ANN and Random Forest. Most of these except Random Forest are used in majority of papers. Here after observing the accuracy, reproducibility, level of solving problem requirement the choice of best classifier for each case of musical instrument classification and speaker identification is made.

a) K-nearest Neighbours

The K-nearest neighbours algorithm (k-NN) is a popular non-parametric algorithm used for solving classification as well as regression problems. Here k-NN is used as a classifier. The input to k-NN are two vectors each deriving from test set and train set. Each vector is of size [1×2560]. As 40 coefficients are taken and 64 centroids are taken in Vector quantization step for codebook formation so [40×64 = 2560] is the size. The output of a k-NN classifier is a class membership. An object is classified by a majority vote of its neighbours. That means an object is assigned to the class that consists of the highest number of common elements among its nearest neighbours. is a positive integer. If K = 1, then the object is assigned to the class of that single nearest neighbour. Here one member from test set is compared against all members of the train set. The class for that member of test set is assigned corresponding to that class in train set with which it receives maximum number of nearest neighbours. In k-nearest neighbour classification, the training dataset is used to classify a testing dataset. The algorithm is described as follows:

1.Number of neighbours is to be selected.

2.According to Euclidian distance or any other appropriate distance (in this case Euclidian), K nearest neighbours of the member of test data to be classified is taken (say a new data point).

3.Among these K neighbours number of data point in each category is calculated.

4.The new data point is assigned to the category where it finds maximum neighbours.

Here the distance used is Euclidian distance.

For two n-dimensional vectors $P = \{p_1, p_2, \dots, p_n\}$ and $Q = \{q_1, q_2, \dots, q_n\}$ Euclidean distance is defined as:

$$d(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{3.13}$$

b) Support Vector Machine(SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression problems. However, it is mostly used in solving classification tasks. Each data item is plotted as a point in an n-dimensional space (where n is number of features) with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that differentiates the two classes with the maximum margin.

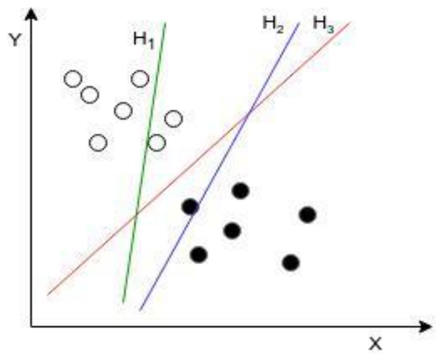

Figure 3.17: Among the three hyper-planes (H1, H2 and H3) H3 separates the training data with maximum margin.

The sum of distances between two boundary points of two classes has to be maximum in order to choose that hyperplane as optimal hyperplane.

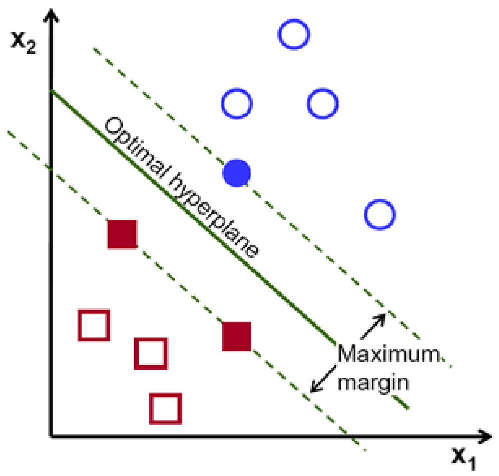

Figure 3.18: Choice of Optimal Boundary

The advantage of SVM is that it deals with points which are most unlike to its group. It is a risk taker and hence a very efficient algorithm. In this study we applied linear SVM and achieved good results.

c)Random Forest Classification

Random Forests or Random Decision Forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class based on maximum votes achieved from the number of decision trees used. Random forests are very efficient but has a tendency to overfit training data. In this study train data has not been overfitted for it is tested with multiple test cases in order to reach best conclusion. When the data is scattered, Random Forest is considered to be one of the best methods for classification.The intuition behind Random Forest classification are as follows
1.Random K data points from the training set are picked.
2.A Decision Tree is built based on these K data points
3.The number of trees required to build for optimal classification output is chosen and steps 1 and 2 are repeated for that number of times for those trees. Number of trees chosen be equal to N.
4.For the new data point, say a test case each one of the N-tree trees predicts the category or class to which the data point belongs and the new data point is assigned to the category that wins the majority vote.
Number of trees chosen in this case is 10000.

i) Decision Tree Algorithm

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches and represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). The algorithm uses entropy

to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.
The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).The intuition behind Decision tree construction is explained with an example below.

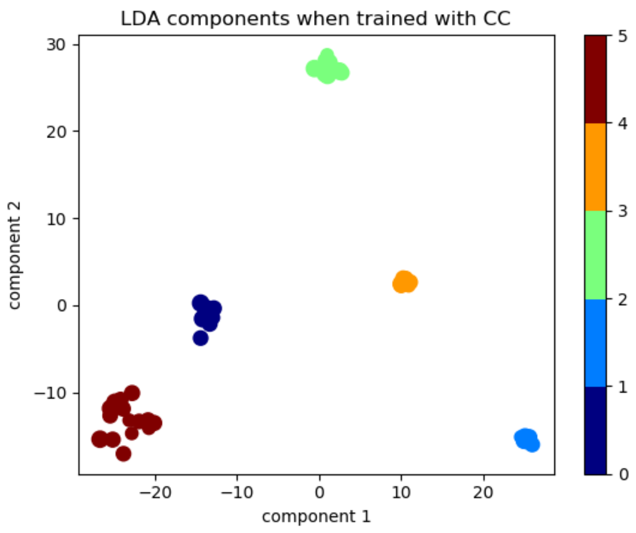

Figure 3.19: LDA components of 5 different musical instruments trained with CC

Five different musical instruments were taken to design the system. For each musical instrument 15 train samples are 3 test samples were there. Each test sample was compared against all train samples and class was assigned as the one with which that particular test case had maximum similarity. Random Forest, SVM, K-NN and ANN all four algorithms works very well with higher dimensional situation. As each vector was of size [1×2560] so although these algorithms worked pretty well with them yet it is not possible to plot such data. To plot these higher dimensional data there are number of techniques present. PCA or Principal Component Analysis is the unsupervised dimensionality reduction technique while LDA or Linear Discriminant Analysis is the supervised dimensionality reduction technique. As here, it is needed to observe the overlapping caused between different musical instruments, LDA on train set is applied. Least overlapping occurs when the acoustic vectors are achieved from Unfiltered Cepstral Coefficients.Here with this plot an intuition behind the workflow of Decision Tree is obtained.
Class 1 or Dark blue represents drum class
Class 2 or Sky blue represents flute class
Class 3 or Green represents trumpet class
Class 4 or Orange represents violin class
Class 5 or Brown represents piano class

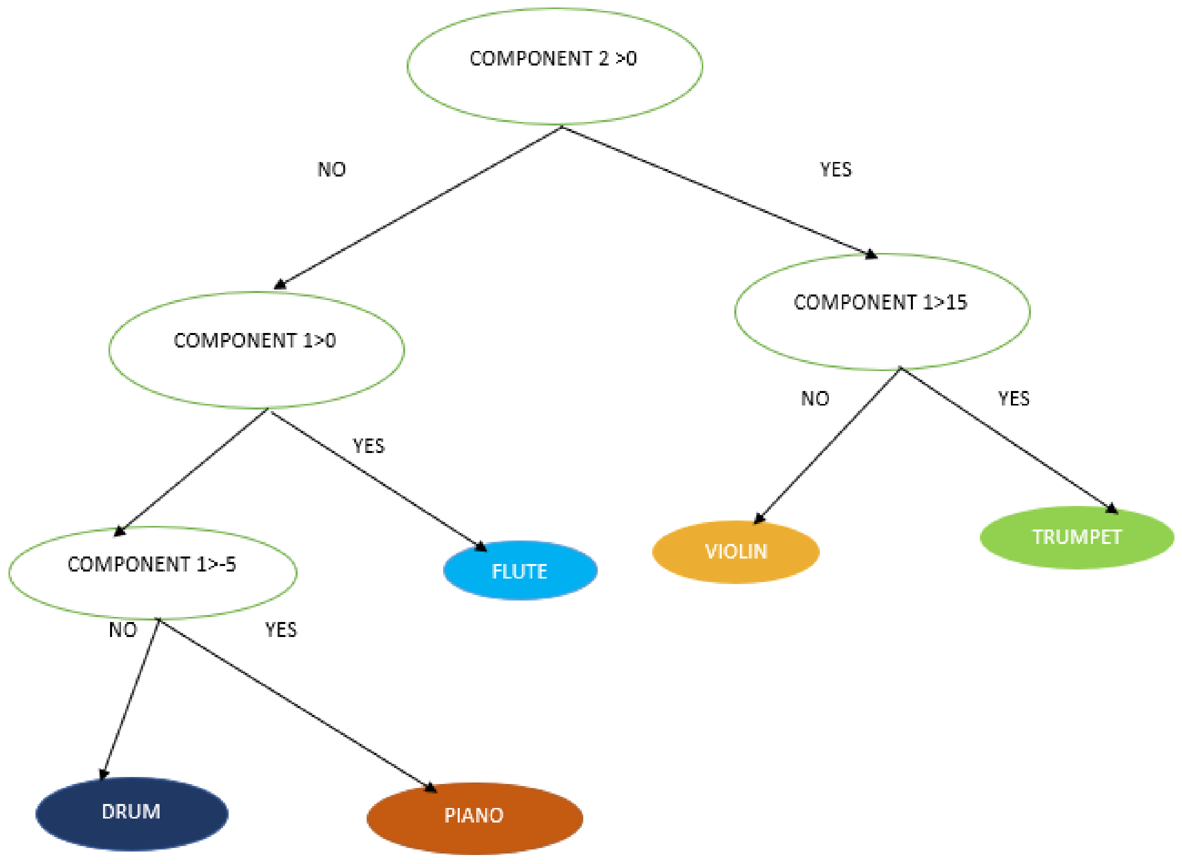

Figure 3.20: Decision Tree construction for 5 musical instruments

The Decision Tree works like the above.

d)Artificial neural networks

These are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" (i.e. progressively improve performance on) tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analysing example images that have been manually labelled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any a priori knowledge about cats, e.g., that they have fur, tails, whiskers and cat-like faces. Instead, they evolve their own set of relevant characteristics from the learning material that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons (a simplified version of biological neurons in an animal brain). Each connection between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it.

The input nodes connect to layer of neurons which connects again to the next hidden layer of neurons (if number of hidden layers>1) and after connecting to n hidden layers (specified by user) at last the network connects to output layer. Now there are weight associated to each connection or synapse and that gets modified due to backpropagation in order to minimise error. So, in this study MLP or Multi-Layer Perceptron is used. A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not separable. In this study four hidden layers are used apart from input and output layer. The Rectifier activation function is used for input and hidden layers while SoftMax activation function which can be vaguely said as categorical version of sigmoid function where sigmoid is used for binary classification SoftMax function is used for multi class classification.

The loss function used here is categorical Cross entropy as the Cross entropy uses logarithm so it helps the network to reach the goal faster by faster training.

At first the weights assigned to the system are random. So, the feedforward propagation occurs and system achieves percentage of likeliness for each class. It then compares it against the actual values which is one for that particular class and zero for others. The network in order to reduce the error backpropagates to adjust the weights. This occurs for several epochs taking a batch size. The process of learning takes place with samples from train set and hence is a supervised learning mechanism. During the test case the system applies it’s learning to predict the output. In this study the Adam optimiser is used. Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models.
The Rectifier function is given by,

$$\Phi_x = \max(x,0) \tag{3.14}$$

Where x is the independent variable of input from input/hidden layers
The Sigmoid function is given by,

$$\Phi_x = \frac{1}{1+e^{-x}} \tag{3.15}$$

Where x is the independent variable. This is mostly used in binary classification.

$$\sigma(z)_j = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zk}} \text{ for } j = 1, \ldots\ldots, K \tag{3.16}$$

z = $w_0\varkappa_0 + w_1x_1 + \cdots\cdot\cdots w_mx_m$ where w are the weights is the input variable and j = 1, 2…...K are the categories or classes.
SoftMax function calculates the probabilities distribution of the event over ‘n’ different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

The main advantage of using SoftMax is the output probabilities range. The range will 0 to 1 and the sum of all the probabilities will be equal to one. If the SoftMax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.

The Cross-entropy loss function is the loss function that actually calculates the error between actual and predicted value of output and makes the network reassign the weights by backpropagation.

It is given by,

$$H(p, q) = -\sum_{x} p(x) log q(x) \tag{3.17}$$

where q is the predicted output value and p is the actual output value for a independent variable x. In this case of categorical classification q is the probability of likeliness found by SoftMax. It is between 0 to 1. For the same class probability is higher and p is equal to 1 for that class

3.4.5. Classifier Selection

The selected classifier obtained for this study is ANN for the following reasons.

1) It gives maximum accuracy against both MFCC and CC in all cases. As these two are the most important feature in this case so this point is taken under consideration.
2) ANN has ground for improvement. If better tuning is done, the system will have the potential for better accuracy.
3) When the system is trained with ANN, the loss function acquired is minimum and system is well trained.
4) Alternative to ANN, K-NN can also be used-NN will be helpful in situations where there is a need to increase the train set. For ANN any increase in train set, requires further tuning.
5) The other classifiers have lesser accuracy than ANN and K-NN.

3.5. Architecture of OSTI-SI system

Open set text independent speaker identification is the most challenging sub class of speaker identification where it is assumed that the test voice may come from any source. In the open-set case, it is not known from beforehand whether the test voice pattern is actually present in the database or not. Open-set matching is therefore more challenging as it not only involves a comparison technique but also requires appropriate thresholds to prevent false matching of new voices with existing voices. Typical applications of closed-set speaker verification include voice-based authentication systems while open-set speaker identification is required in surveillance and criminal investigations e.g. ransom callers.

3.5.1. Feature Extraction

In this study only two best features obtained from previous study is used. Those are MFCC and CC. These features have many good properties and with these best results were obtained in previous case.

a) MFCC

As discussed earlier. MFCCs extracts the phonemes of uttered speech disregarding background noise and emotion. In case of sound produced by humans the speech is that sound which generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. Each human being has his /her own way of speaking. The accent of speaking, the pitch and tone differ for every individual. The shape of vocal tract determines what sound comes out. Determining the shape accurately, should give us an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope. MFCCs are commonly used in speech recognition and speaker recognition problems. The MFCCs represents Timbre of the sound which is the property by which a listener distinguishes two sounds in spite of them being played with same pitch. This property is very powerful in classification of sound as it can distinguish two sounds based on the nature of their spectral envelope.

MFCC is calculated by the same steps using same frame numbers and window as discussed before in previous case. So, the details are skipped here.

The reasons for using this feature for experimentation in this study are as follows:

1) Mel scale mimics human ear and hence the lower frequencies are paid more attention than the higher ones hence it actually is useful to identify voices like humans do.

2) In case of sound produced by humans it is needed to understand that speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. Determining the shape accurately, should give us an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and MFCC represents this envelope well enough.

3) Timbre of the sound which is the property by which a listener distinguishes two sounds in spite of them being played with same pitch. MFCC is a timbre related feature hence has a broad-spectrum application.

4) The periodogram spectral estimate still contains a lot of information not required for tasks like ASR. In particular the cochlea cannot discern the difference between two closely spaced frequencies. This effect becomes more pronounced as the frequencies increase. For this reason, clumps of periodogram bins are taken and are summed up to get an idea of how much energy exists in various frequency regions. This is performed by Mel filterbank: the first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies get higher our filters get wider as we become less concerned about variations. We are only interested in roughly how much energy occurs at each spot.

For all these reasons MFCC is chosen as a feature in this study.

b) CC

In the previous study the motivation behind CC was explained. A good accuracy is observed in musical instrument classification.

CC is extracted following the steps as discussed in previous case.

The reason for using CC in this case study for experimentation are as follows:

1) CC is less used in this field so it is required to understand, how this feature works in different situations.

2) CC is a timbre related feature so it has the potential to give good accuracy in this case also.
The plot of MFCC is given below for a human speech sound

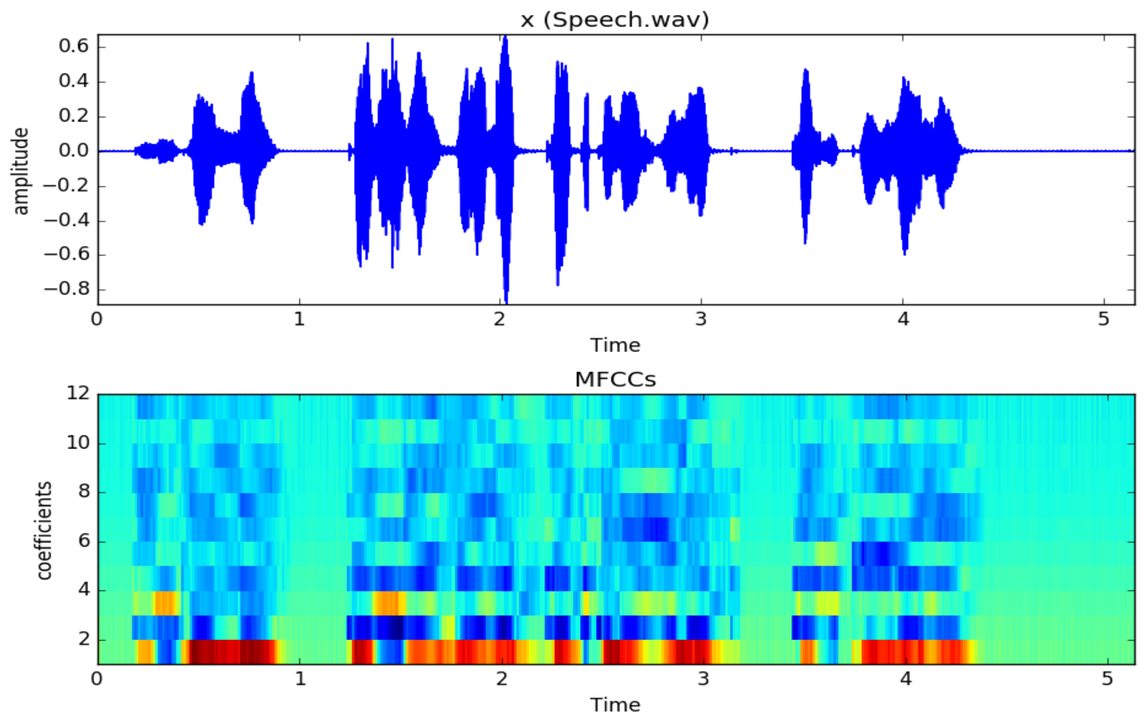

Figure 3.21: MFCC of a speech signal

The above is the visualization of the MFCC analysis on speech sound. In fact, the issue is that every coefficient is a representation of a different level of obstruction of the spectral shape. So here, for example, first 12 coefficients are plotted. The lower frequency regions are better plotted than the higher frequency regions. It is a very compact representation of the characteristics of sound.

The plot of CC for a speech signal is given below

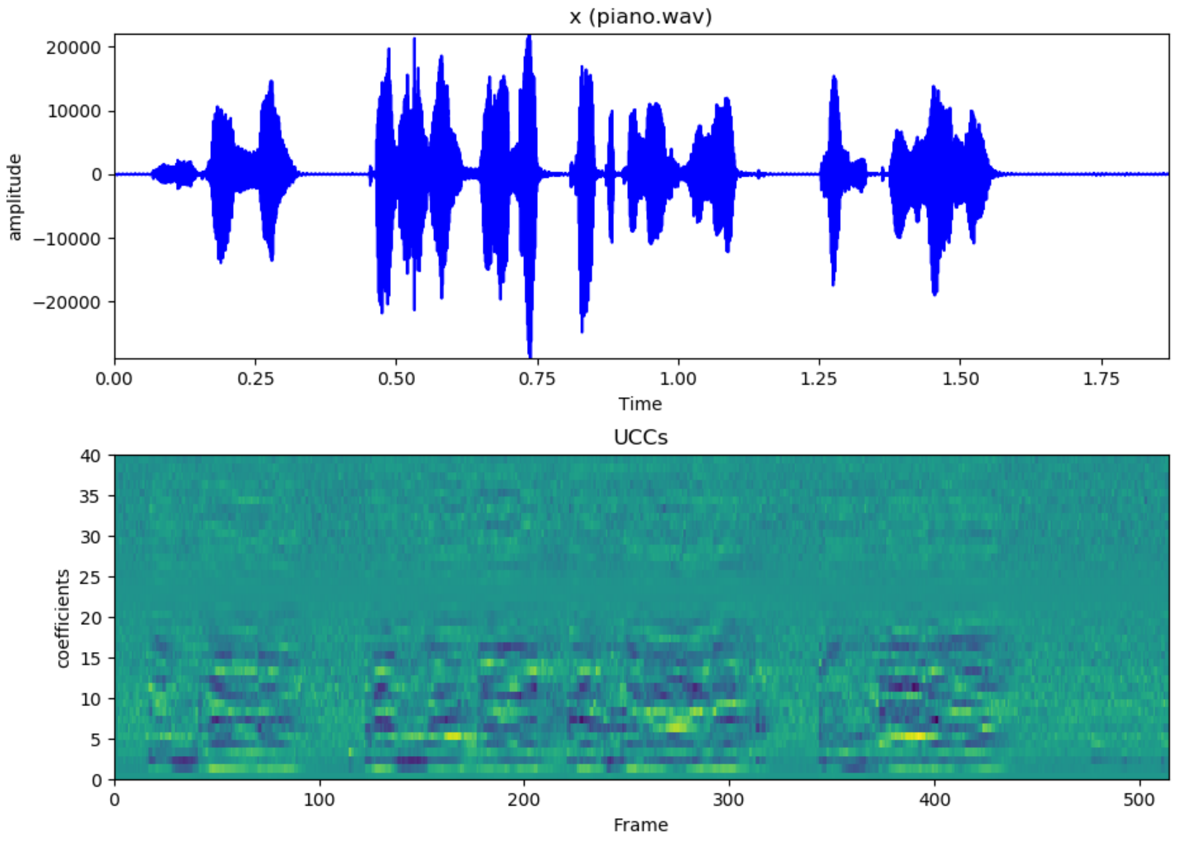

Figure 3.22: CC representation of speech with respect to Frames

In the above picture the variation of cepstral coefficients up to frame length 500 is shown. It can be observed that both higher and lower frequency components of the sound is gathered. Unlike MFCC more emphasis is not paid on lower frequencies, equal emphasis is paid to both of them.

3.5.2. Feature Selection

The feature selected for final design of the system is MFCC for the following reason.

1) Although CC gives a good accuracy in closed set speaker identification even better than MFCC in some cases but accuracy steeply falls in case of open set identification of voices. The likeliness probability obtained for MFCC during classification phase is reproducible and non-random and also maintains integrity while for CC it is not possible to design an open set system with it as probability of likeliness is random and for even one case good threshold couldn’t be predicted even manually and selecting threshold was not easy and not accomplished for this new model as the likeliness probability did not form a pattern like MFCC hence it is not possible to design an adaptive threshold system with such a feature. It will be discussed with examples in next chapter

Hence MFCC is chosen.

3.5.3. Compression of Data

This step although a minor step yet is very useful and required for a efficient system. The features MFCC and CC are very high dimensional. To reduce their dimensions yet not compromising system's accuracy lossy compression technique like Vector quantization is used followed by flattening in order to input them into classifiers.
The number of Centroids used for MFCC and CC is 64. The vector size of MFCC is [1×1280] for there are 64 centroids and 20 coefficients. In general, for a single sound there is a [20×64] matrix which after flattening [1×1280] is obtained for a single sound sample. For CC the vector size is [1×2560] for there are 64 centroids and 40 coefficients. In general, for a single sound there is a [40×64] matrix which after flattening [1×2560] is obtained for a single sound sample.

3.5.4. Classification

This is a vital step in any recognition problem. In this step the features from the test data and train data are compared against each other to measure similarity in order to achieve classification. There are many classifiers present but the choice of most effective one for a specific problem which helps in achieving the goal of the system, has better accuracy compared to others and prevents overfitting along with producing reproducibility is a complex task involving many experiments and observations. In this study four major and most prevalent machine learning classifiers are used which are SVM(linear), K-NN (number of neighbours = 5), ANN and Random Forest. The intuition behind K-NN, ANN, SVM and Random Forest classification has been discussed before. So, this part is skipped here. It needs to be mentioned here that during the first phase of our study a different classifier was used. The metric for classification was normalized minimum Euclidian distance. This was adapted for better open set identification as thresholds became easier to predict.

The sum of the Euclidean distances between each feature vector to the nearest centroid in a codebook was measured as the distortion lying between a codebook and a group of feature vectors. The Euclidean distance was measured for each test sample against all train samples to obtain distortion. The Euclidean distances obtained for a certain test sample when compared against all train samples was normalized. Normalization sets all Euclidean distances within the range of 0-1. This simplified the thresholding process. Let $a = \{a_1, a_2, \dots, a_n\}$ denote the Euclidean distances of a certain test sample against n train data. When each Euclidian distance was normalized, it was normalized as follows:

$$D = \frac{a_i}{\sqrt{\sum_{i=1}^{n} (a_i)^2}} \; ; 1 \leq i \leq n \tag{3.18}$$

Minimum Euclidean distance is the distortion which defines the similarity measure. The train data against which the test sample processes minimum Euclidean distance in a set of n train samples is the best match for that test sample. In closed set speaker identification, the test sample is matched against that train data. But in open set the scenario is different. The decision criteria are the minimum normalized Euclidian distance. If it is greater than the threshold the voice is rejected as unknown else it is matched with its best match. The open set identification was based on the following decision.

$$D(X, S_i) > \sigma_i \rightarrow reject$$
$$D(X, S_i) < \sigma_i \rightarrow accept \tag{3.19}$$

Here S_i is the set of train data/registered speakers stored in the database and X is the current test sample, D denotes normalized Euclidian distances, σ_i are the threshold values chosen to obtain equal error rate.
The threshold gradually decreased with increase in number of train data for the normalization factor. The system gave better accuracy than the conventional GMM-UBM.

GMM or Gaussian Mixture Model is trained for each train samples in the training set. A speaker-independent model, or UBM or Universal Background Model, is trained from the out-of-set speakers using the EM or Expectation Maximization algorithm which is an iterative optimisation technique. Every speaker has a model which is represented with certain parameters using a GMM.A score that decides whether a given utterance $Q = \{q_1; q_2; \dots, q_n\}$ originates with speaker 'a' using the mean log-likelihood 1/n log(P(Q|θa)) where N is the number of models in the mixture, θa ={μa,$\sum_{a,}$$\alpha_a$}.The decision function for a UBM is given as follows, where T_θ is the threshold.

$$\frac{1}{n}\log[P(Q|\theta)] - \frac{1}{n}\log[P(Q|\theta_{UBM})] > T_\theta \tag{3.20}$$

The accuracy of this method is actually more than GMM-UBM method which is one of the most dominating method in the field of OSTI-SI. (ii) Unlike GMM-UBM it is independent of UBM model. For GMM-UBM,we had to choose an appropiate UBM for better accuracy (iii) The method provides a simpler approach to speaker identification, where the threshold can be determined from the rate of its decrease with increase in train data and vice versa. Due to normalization, the threshold value lies within 0-1 and it's value for different datasets are quite similar to each other.

In the later part of our study,motivated by this method some new techniques are dealt with.Some of the limitations faced by this method was as follows:

1) The closed set and open set accuracy obtained with K-NN,SVM,ANN and RandomForest was more than this method.
2) This method although was based on Bayes optimal decision yet the metric of posterior probability was minimum euclidian distances.Probability of likeliness obtained in Random Forest or ANN is much more intuitive than the distance.
3) The latter methods took much less time of computations than this approach.
4) The adaptive threshold designed using Random forest regression was facilitated by random forest classification.The computation time and accuracy was much higher compared to this.
5) The K-NN is the improved version of this method.But in this study Random Forest is the best classifier chosen.

3.5.5 Classifier Selection

The selected classifier in this case is Random Forest classifier for the following reasons.

1) For K-NN no metric could be extracted with which open set speaker identification would had been possible, hence it is not chosen as final classifier.

2) The accuracy obtained with SVM was less compared to other methods, hence it was discarded.

3) The ANN although provides a good accuracy but this requires a lot of tuning. In this problem both train and test set are constantly subject to changes hence tuning is required which will make system non-robust and complex. Moreover, as this is a stochastic process accuracy tends to vary. So, this was discarded.

4) The Random Forest classifier with 10,000 trees gives best prediction in this case study and as this is an ensemble learning process, hence accuracy and robustness obtained is more compared to others. The Random forest also provides the likeliness probability which is used later in adaptive threshold design. Hence this classifier has robustness, gives good accuracy, provides a good metric for threshold selection and adaptive threshold design and has not suffered overfitting which is evident from the accuracy obtained in three different cases.

The accuracy obtained in each case is not discussed here purposely. It will be discussed in next chapter.

3.5.6 A System Which Predicts Threshold Automatically

This section is the most innovative part of this case. The open set text independent speaker identification needs a threshold which will separate the known and unknown voices. This threshold selection is done mostly manually. But manual threshold selection is tedious and prone to errors so in order to undo human intervention this system which automatically predicts threshold from some train samples for any test case is generated.

a)The Design

The system contains multiple independent variables as input variable and Random forest regression is applied on them. The reason for using Random forest regression is that, it works with data that are scattered in space unlike linear regressions. In this case, the input variables are four which are number of train data, number of test data, mean of probability of likeliness for known data obtained across their matching pair and mean of the probability of likeliness for known and unknown data obtained against their best match.

This can be simplified. It is to be noted here that for computational efficiency and faster response 7 train samples from each speaker was combined into one. So, the sound sample from each speaker is trained for 1-2 mins duration. The test sample was 10-20 secs each.

So, suppose we take 10 train classes (so 10 samples) in train dataset and 20 test classes (so 20 samples) in test dataset out of which 10 are known and 10 unknowns. Let us arrange the voices in test set in an order that 1 to 10 are known voices and 11 to 20 are unknown voices.
So, the first factor, i.e. number of train data is 10 in this case and number of test data is 20 in this case. The mean of known data is the mean of probability of similarity obtained for 10 known test data with their closest match. This can be called as closed set mean. So, this is the

mean of 10 probabilities. As closed set accuracy is very high hence the error arising due to false mismatch can be neglected. The mean of known and unknown data is the mean of probability of similarity obtained when 20 test cases are compared against all train cases. This can be called as open set mean. In the first case, the mean is high as probability of likeliness between like voices are high. In the second case, mean decreases as the probability of likeliness also decreases as the probability of likeliness is less for unlike voices. After these are fed the threshold is predicted by the system. The system is able to do this for previously 15 to 20 threshold values for corresponding input values has been trained. The threshold values were manually obtained such that there was a good trade-off between false acceptance and false rejection. The system works with any set of voices up next. For the system to work it is needed to specify the number of train data and test data and in the program, it is required to give the name of folders containing train and test data. The system then automatically detects the threshold applicable in such case and mean calculation is done internally. Any train and test dataset can be applied so the system is robust. The system is robust of text spoken and also language spoken.

b)Theory

This system is designed such that it can observe the change in likeliness when two like sounds are compared and when two unlike sounds are compared. The system assigns the new data to the class with which it has maximum similarity. This process works as described earlier, based on majority votes of the decision trees. The next thing it does is, takes mean of the probabilities obtained against each class when compared against that new data. The probability obtained against all classes sums up to one so with increase in train data probability decreases but this is a relative feature not due to less similarity hence this is also inputted to the system. The number of test data is given otherwise the system will not understand the mean of open set. It will not understand how much addition of unknown data is leading to such changes in the mean likelihood. The trade-off obtained between false acceptance and false rejection while designing manual threshold values by trial and error method used in training of the system will determine the predicted threshold. The error which arises in first stage of OSTI-SI during closed set identification process is the error of mismatch. The cause of it can be many but not decision or threshold. So, this error is overlooked in this paper. The second error is False acceptance error(FA), where an unknown voice is mistakenly matched against a train data. The third error is False Rejection error(FR), where the known voice is mistakenly rejected as unknown. These errors rise either due to decision or threshold. In order to gain an equal error rate, where the number of false acceptance rate (FAR) and false rejection rate (FRR) error is balanced the threshold needs to be correct.

c)Random Forest Regression

The multiple input Random forest regression works for this problem as the main intuition for Random Forest regression is to predict output from data which are scattered in space and they work well with high dimensional data. In fact, they are designed for such purpose. The Random forests are again constructed using the following steps.
1.Random K data points from the training set are picked.
2.A Decision Tree is built based on these K data points
3.The number of trees required to build for optimal classification output is chosen and steps 1 and 2 are repeated for that number of times for those trees. Number of trees chosen be equal to N.

4.For the new data point, say a test case each one of the N-tree trees predicts the value of output and the output assigned to it is the average across all the predicted output or y values.

The decision tree splits the regions based on entropy and information gain and takes mean of the output value in those regions. So, for all the regions splatted, there exists corresponding means of output and so when a new data is received, it observes the region where it falls and assigns the output corresponding to the new data as the mean in that region.
As Random forest is used so the final output value is the average of the output values obtained from n such trees. In this case n = 10000 is chosen. Hence this is the system in brief.

3.6 Dimension Reduction Technique

A brief discussion on dimension reduction techniques used for proper visualization of results from cases needed to be discussed here

a)LDA or Linear Discriminant Analysis

Linear discriminant analysis (LDA) or discriminant function analysis is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. Discriminant analysis is used when groups are known a priori (unlike in cluster analysis). Each case must have a score on one or more quantitative predictor measures, and a score on a group measure. In simple terms, discriminant function analysis is classification - the act of distributing things into groups, classes or categories of the same type. It is the supervised version of PCA. It is used on train samples whose labels are known beforehand to observe the cluster formed by intra-class samples and observe the overlapping or similarity between clusters or members of between-classes.

b)t-SNE or t-distributed stochastic neighbour embedding (t-SNE)

It is a machine learning algorithm for visualization developed by Laurens van der Maaten and Geoffrey Hinton. It is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modelled by nearby points and dissimilar objects are modelled by distant points with high probability. t-SNE has been used for visualization in a wide range of applications, including computer security research music analysis cancer research, bioinformatics, and biomedical signal processing. It is often used to visualize high-level representations learned by an artificial neural network.

3.7. Chapter Summary

In summary of musical instrument classification, it can be said that the final system is designed with CC and ANN which is the proposed approach. The input signal after frame blocking and windowing undergoes the process of CC extraction. The CC features after extracted are vector quantized and flattened to obtain a vector of size [1×2560] representing each sample of train set and test set. After this, these features are fed into ANN where there are four hidden layers and Rectifier activation function is applied to input layer while SoftMax function is applied to hidden and output layers. The loss function used here is Cross entropy and Adam optimizer. The metric used is accuracy. ANN and K-NN gives maximum accuracy across all cases followed by SVM and Random Forest. The accuracy obtained in different three cases will be discussed in next chapter. It is capable of distinguishing instrumental sound irrespective of what pitch or tone or dynamics in which they are played. In the next chapter this will be discussed in more details.

In the study concerning human voices, the MFCC of the voices of speakers are extracted and are treated as acoustic vectors. After flattening them they are inputted into Random forest classifier and from there the threshold is automatically calculated by Random forest regression based on some train data which can be treated alike for all cases. After employing the predicted threshold accuracy of open set text independent speaker identification system is observed. The error of false rejection is less but the error of false acceptance is minimal. There is also negligible false mismatch error. Hence MFCC for feature extraction, Random forest classification algorithm for classification and multi input Random forest regressor for automatic threshold generation is the ultimate proposed methods used in this study.

4.EXPERIMENTATIONS AND RESULTS

This section is a major part of this study as it consists of all the results and plots that has led us to the conclusion. It is this part, which has helped us in choosing appropriate feature and classifier for the system. The experiments are mostly done with Python code in Spyder environment with only a few are carried out in MATLAB environment during the first phase of the project. Python 3.6 and MATLAB R2015a are the versions used.

4.1. Experimentations of Musical Instrument Classification system

4.1.1 Dataset for Musical Instrument Classification

The most early and one of the vital step of designing a system is the collection of relevant, solid, reliable dataset required for the process. In this case, the dataset for musical instrument classification is derived from Philharmonia Orchestra instruments and collected from [33] and Freesound music database [34]. The pitch of the instruments ranged from A#3 to G5 and the dynamic was of two types, Forte and Piano. The articulation was normal. Dynamics means how loud or quiet the sound is. In music, articulation is the direction or performance technique which affects the transition or continuity on a single note or between multiple notes or sounds. Pitch may be quantified as a frequency, but pitch is not a purely objective physical property; it is a subjective psychoacoustical attribute of sound. From the whole set experimentation is done only with ten instruments. They are Organ, French-horn, Cello, Clarinet, Tambourine Drum Flute, Trumpet, Violin, Piano. They belong to five families of instruments.
Keyboards and Harp – Organ, Piano
Woodwind Family-Flute, Clarinet
Brass Family-Trumpet, French-Horn
String Family-Violin, Cello
Percussion Family-Drum, Tambourine

4.1.2 Experimentations

Three experiments have been carried out varying the characteristics of sounds. Two instruments are carried out with same set of five instruments with varying conditions to study the effect of change of pitch and dynamics in the classification process and the third experiment is made with 10 instruments.

Experiment 1

The first experiment was carried out with Drum, Flute. Trumpet. Violin. Piano which belongs to five different families of musical instrument mentioned earlier. For each instrument or class 15 samples were taken during training and 3 samples were taken for testing so a total of 75 samples were taken during train phase and 15 samples were used in test phase to check the accuracy and robustness of the system. The samples belonged to random pitch and dynamics. The experiment was carried out with six features and four classifiers as specified in the previous section. The features are MFCC, CC, LPC coefficients, Centroid, HPCP and Pitch salience. The classifiers are K-NN, ANN, Random Forest and SVM.

The results of the first experiment are tabulated below.

TABLE 4.1: Tabulation of Results of First Experiment

	PITCH SALIENCE	HPCP	CENTROID	LPC	MFCC	CC
K-NN	7	6	8	8	13	13
SVM	6	9	5	10	13	12
Random Forest	9	8	10	8	12	12
ANN	7	9	11	8	13	14

The table portrays how many test samples are classified correctly amongst the 15 test samples used in this experiment.

The results show that the accuracy is quite low for Pitch salience peaks, HPCP, LPC coefficients and Centroid while it is quite good for MFCC and CC. It can be observed that maximum accuracy is obtained against ANN followed by KNN for CC and MFCC. Almost all samples were correctly classified with these two features. In this experiment CC yields maximum accuracy followed by MFCC, Centroid, LPC coefficients and HPCP and Pitch salience. The accuracy for the first two are almost same. They obtained maximum accuracy with Random forest while LPC coefficients gained maximum accuracy with SVM and Centroid with ANN. The ANN if better tuned can lead to some improvement.

The Linear Discriminant Analysis which is a kind of supervised dimension reduction technique is used to observe how much overlapping occurs between the classes for each of these features. This is applied to train samples. The degree of overlapping tells us how well each instrument is trained and how much that particular feature is capable of distinguishing the characteristics of the class of instruments under consideration. Here LDA is used as the label for train samples is known beforehand, so the algorithm clusters each instrument but brings out the degree of likeliness between different instruments. The less the overlapping is, better is the feature and more potential the system has, in accurate classification.

The LDA plots obtained for this set of instruments which has random pitch and dynamics are given and explained below for all six features.
Class 1 or Dark blue represents drum class
Class 2 or Sky blue represents flute class
Class 3 or Green represents trumpet class
Class 4 or Orange represents violin class
Class 5 or Brown represents piano class

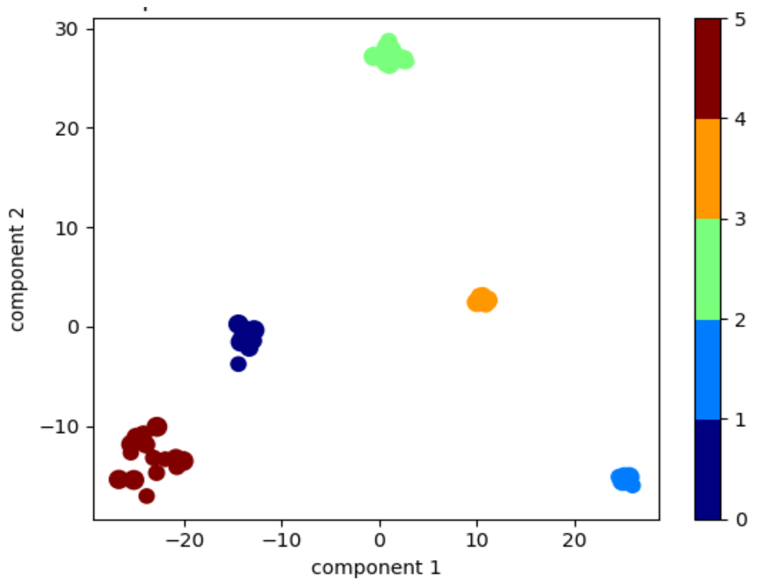

Figure 4.1: LDA plot of CC features

In the above plot it can be observed that there is no overlapping between members of different classes and hence it proves that this feature has the potential to produce very good accuracy. The plot proves the accuracy obtained in table 1. It has potential for even better accuracy.

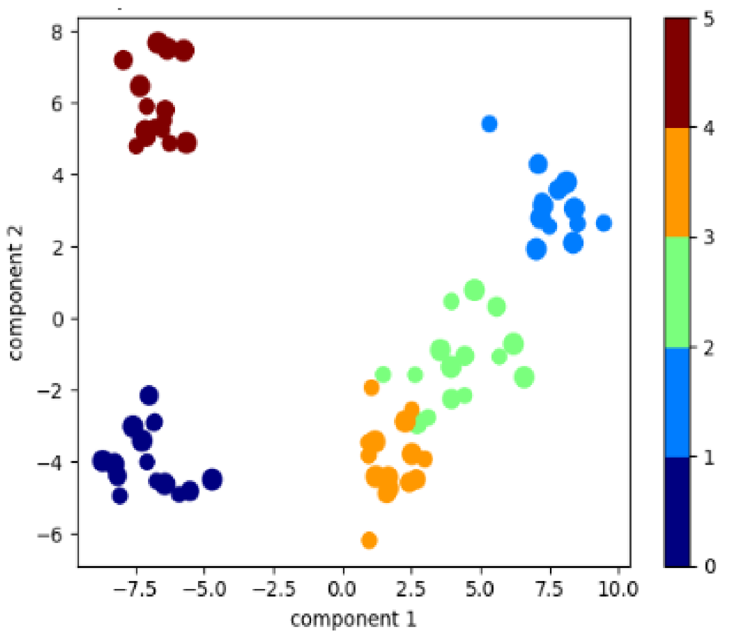

Figure 4.2: LDA plot of MFCC features

In the above plot it can be observed that there is very less overlapping between members of different classes and hence it proves that MFCC has the potential to produce very good accuracy but CC has more potential than it as the degree of overlapping in previous case was nil. The clusters are less tight than the CC. The plot proves the accuracy obtained in table 1. It has potential for even better accuracy.

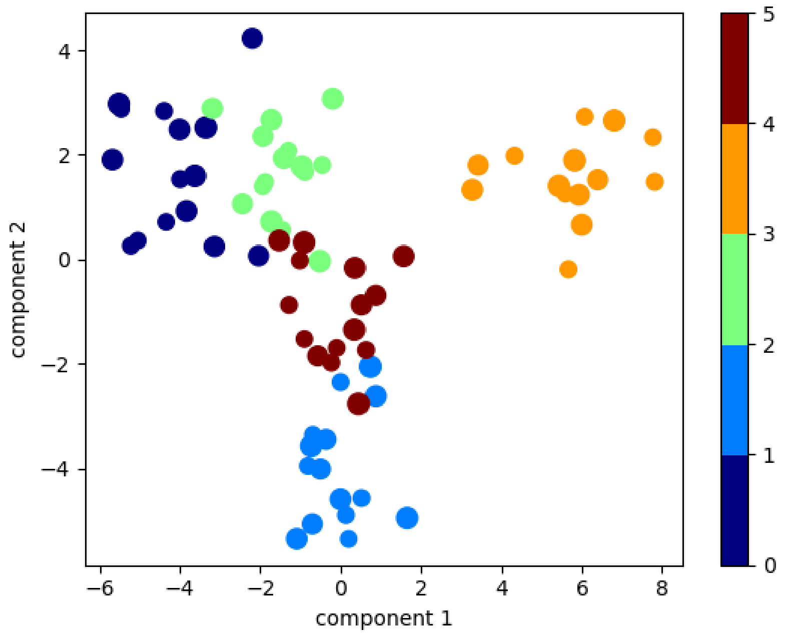

Figure 4.3: LDA plot of LPC coefficients

In the above plot it can be observed that there is some overlapping between members of different classes specially the first third and fifth class members have overlapping regions also the clusters are less tight. Hence the decrease of accuracy as observed from the table.1 has its reasons.

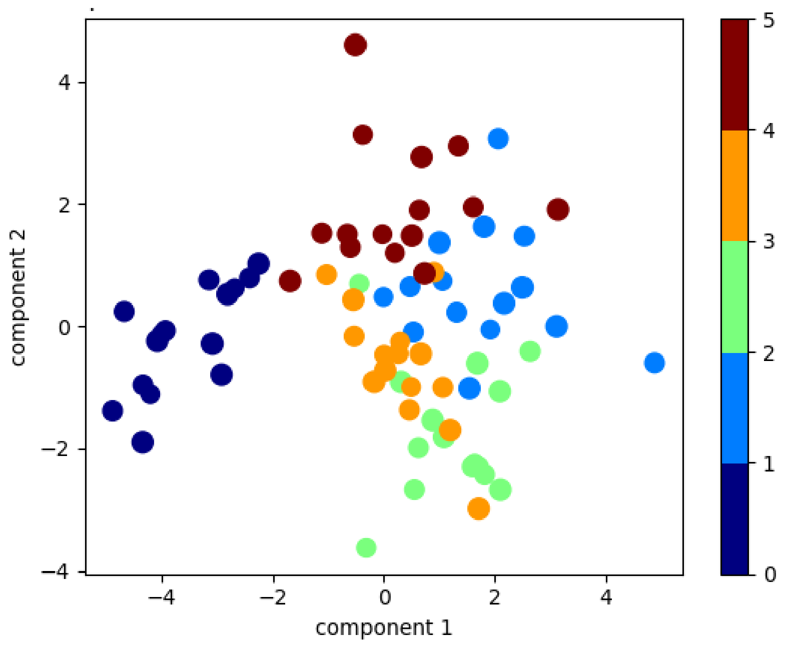

Figure 4.4: LDA plot of HPCP

In the above plot it can be observed that there is a good overlapping between members of different classes and also the clusters are less tight. Hence the decrease of accuracy as observed from the table.1 has its reasons. It proves that this feature is not good enough to bring out uniqueness of the musical sound under the said conditions where pitch and dynamic chosen was random.

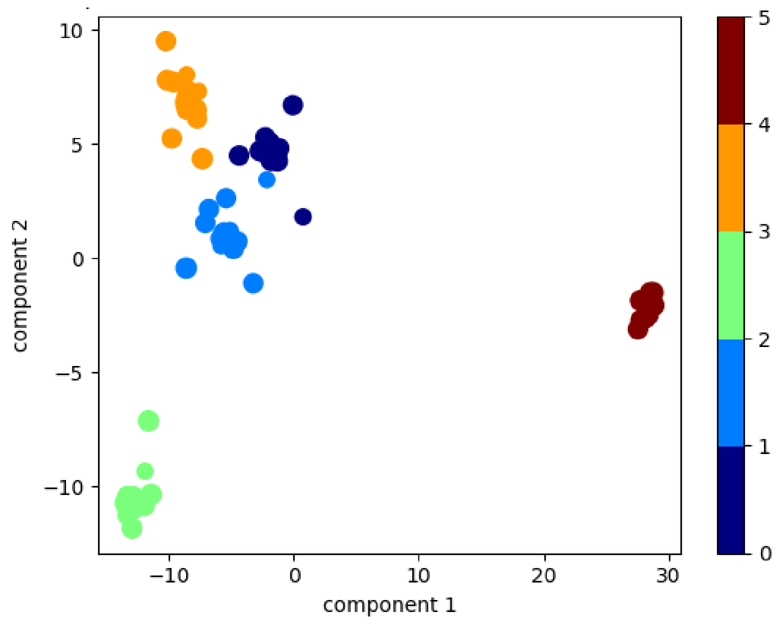

Figure 4.5: LDA plot of Centroid

In the above plot it can be observed that there is very less overlapping between members of different classes and also the clusters are very tight. Hence the good accuracy as observed from the table.1 has its reasons. This feature for this class of instruments and conditions has potential for more accuracy but it should be kept in mind that the brightness attribute is subject to changes under little changes in condition.

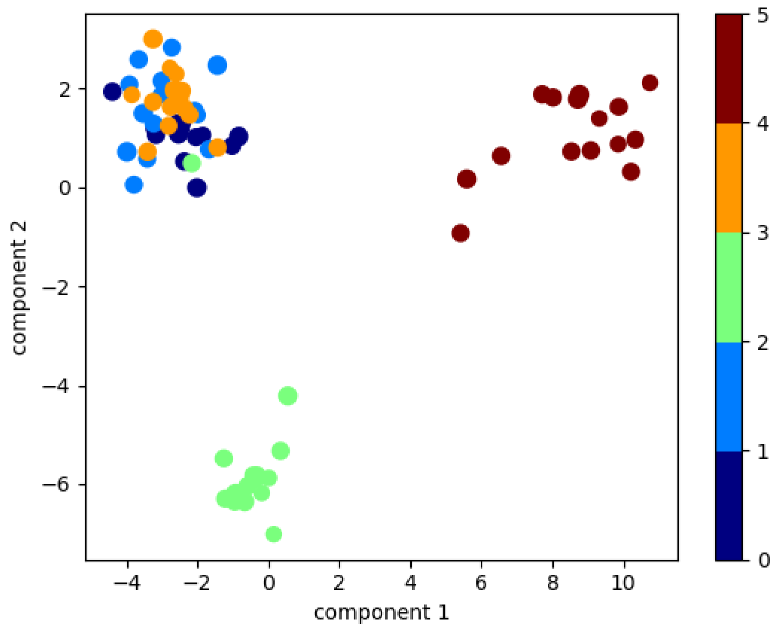

Figure 4.6: LDA plot of Pitch salience peaks

In this case there is maximum overlapping between members of different classes and also the clusters are less tight. Hence the decrease of accuracy as observed from the table has its reasons. It proves that this feature is not at all good enough to bring out uniqueness of the musical sound under the said conditions where pitch and dynamic chosen was random. The accuracy obtained in table.1 has potential to decrease.

<u>Experiment 2.</u>

The second experiment was carried out with Drum, Flute. Trumpet. Violin. Piano belonging to five different families of musical instrument. For each instrument or class 15 samples were taken during training and 3 samples were taken for testing so a total of 75 samples were taken during train phase and 15 samples were used in test phase to check the accuracy and robustness of the system. Each musical instrument had random dynamics but a fixed pitch.
Drum- Unpitched
Flute – B4 & B5
Trumpet-C5&C6
Violin- A3& A4
Piano-D4
The pitch information of each instrument is different. This was done deliberately to observe the capability of classification of HPCP and Pitch Salience specially. If CC and MFCC gives good accuracy under this condition also, it can be inferred that they are robust features as the train samples and test samples are different from the previous case.
The experiment was carried out with six features and four classifiers as specified in the previous section. The features are MFCC, CC, LPC coefficients, Centroid, HPCP and Pitch salience. The classifiers are K-NN, ANN, Random Forest and SVM.
The results of the second experiment are tabulated below.

TABLE 4.2: Tabulation of results of second experiment

	PITCH SALIENCE	HPCP	CENTROID	LPC	MFCC	CC
K-NN	8	12	8	10	12	14
SVM	7	10	7	7	12	12
Random Forest	8	10	7	9	11	13
ANN	9	12	6	8	13	14

The table portrays how many test samples are classified correctly amongst the 15 test samples used in this experiment.

The results show that the accuracy is quite low for Pitch salience peaks, LPC coefficients and Centroid while it is quite good for MFCC, HPCP and CC. It can be observed that maximum accuracy is obtained against ANN followed by K-NN for CC and MFCC. Almost all samples were correctly classified with these two features. In this experiment CC yields maximum accuracy followed by MFCC, HPCP, LPC, Pitch salience and Centroid. The accuracy of HPCP has risen quite steep compared to previous case. As the pitch information was different hence this feature which brings out the dominant pitch classes present in musical instruments and was able to distinguish different instruments. CC and MFCC dominantly gave very good accuracies. In this case also the accuracy of CC was more than MFCC. Accuracy has dropped for Centroid. Pitch salience peaks shows little potential in classification under varied pitch conditions also LPC accuracy has been constant.The LDA plots are given below.
Class 1 or Dark blue represents Piano class
Class 2 or Sky blue represents Drum class
Class 3 or Green represents Flute class
Class 4 or Orange represents Trumpet class
Class 5 or Brown represents Violin class

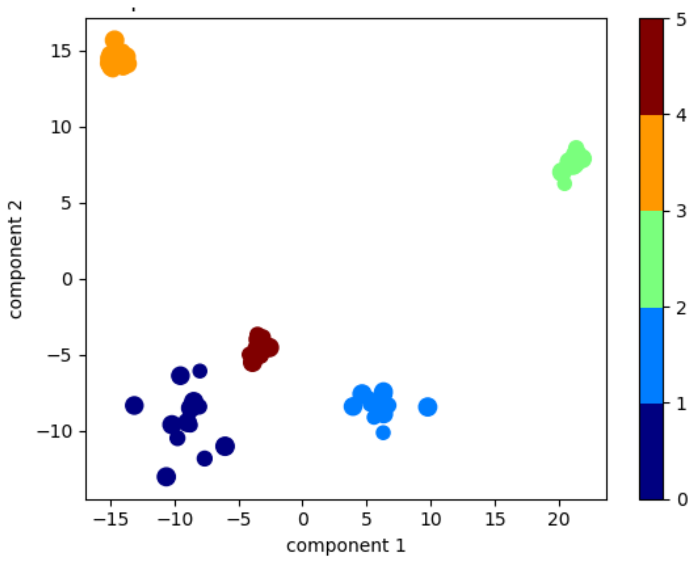

Figure 4.7: LDA plot for CC in experiment 2

In the above plot it can be observed that there is no overlapping between members of different classes and hence it proves that this feature has the potential to produce very good accuracy. The plot proves the accuracy obtained in table 2. This also proves the robustness of CC features at least with these five classes of instruments for in two experiments different samples and conditions were used. It has potential for even better accuracy.

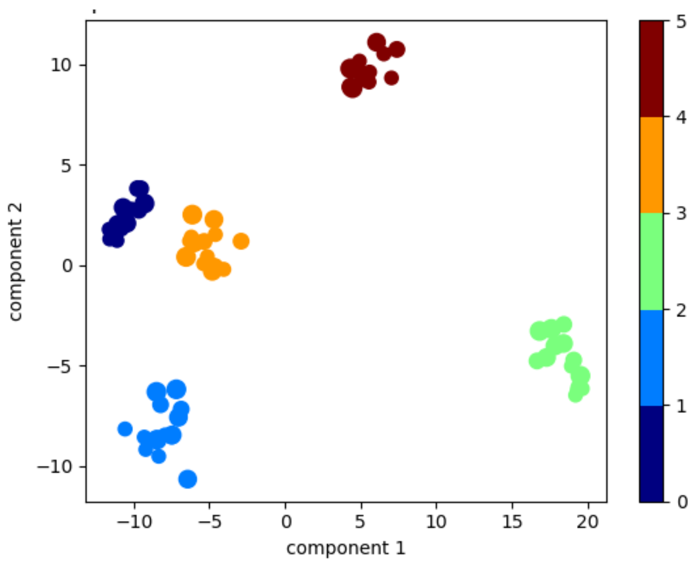

Figure 4.8: LDA plot for MFCC in experiment 2

In the above plot it can be observed that there is no overlapping between members of different classes and hence it proves that this feature has the potential to produce very good accuracy. The plot proves the accuracy obtained in table 2. This also proves the robustness of MFCC features at least with these five classes of instruments for in two experiments different samples and conditions were used. In this experiment due to varying pitch characteristics the classification is even better. This is due to the fundamental nature of MFCC capturing the frequency information from spectrum along with Timbre by Short time power spectrum analysis. It has potential for even better accuracy.

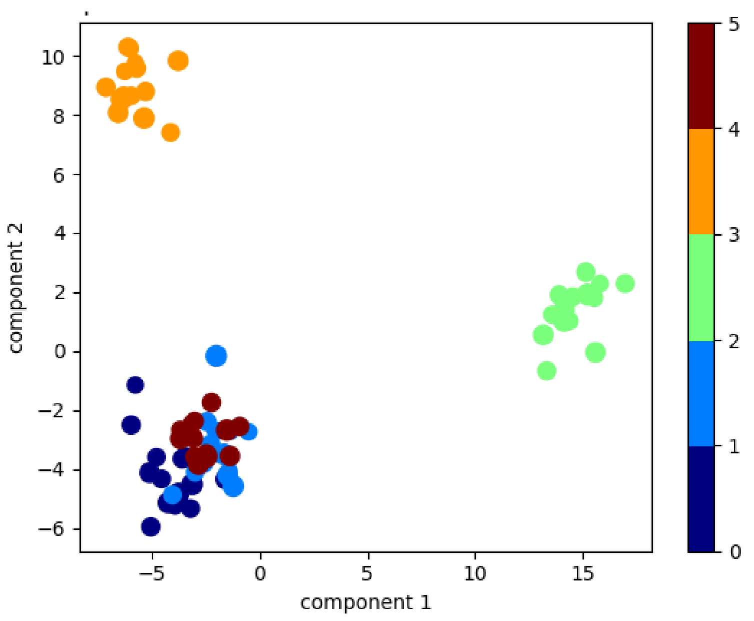

Figure 4.9: LDA plot for HPCP in experiment 2

In the above plot although overlapping is there but it can be observed that the clusters are tighter and overlapping is far less compared to previous case. Only two class 2 and 5 is overlapping while other classes are well separated from each other. This explains the accuracy int table 2. So, it can be inferred that HPCP can be used in classification where the pitch information is different for the instruments.

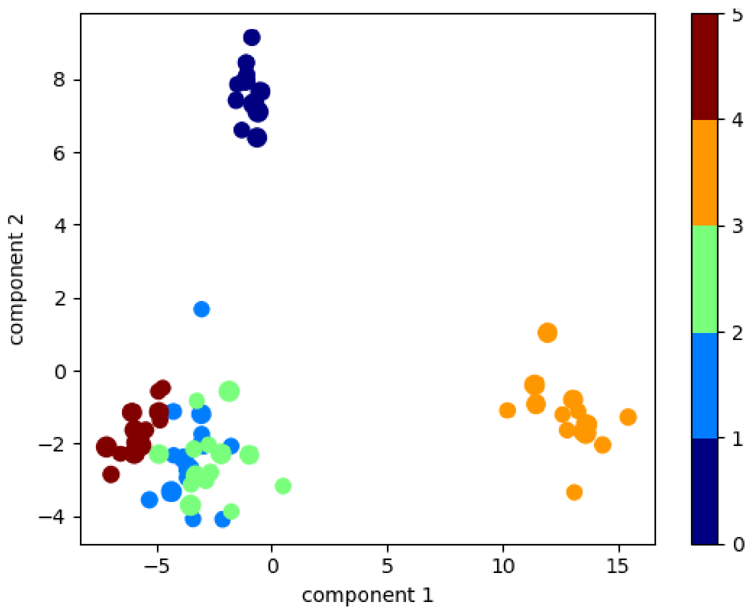

Figure 4.10: LDA plot for Pitch Salience peaks in experiment 2

In the above plot although overlapping is there but it can be observed that the clusters are tighter and overlapping is less compared to previous case. Only second and third class is overlapping.

Although low accuracy is achieved for pitch salience peaks in this case, yet varying pitch information has caused improvement, so in brief it can be conferred that the pitch salience has potential for improvement in case where pitch information of different instruments is also different.

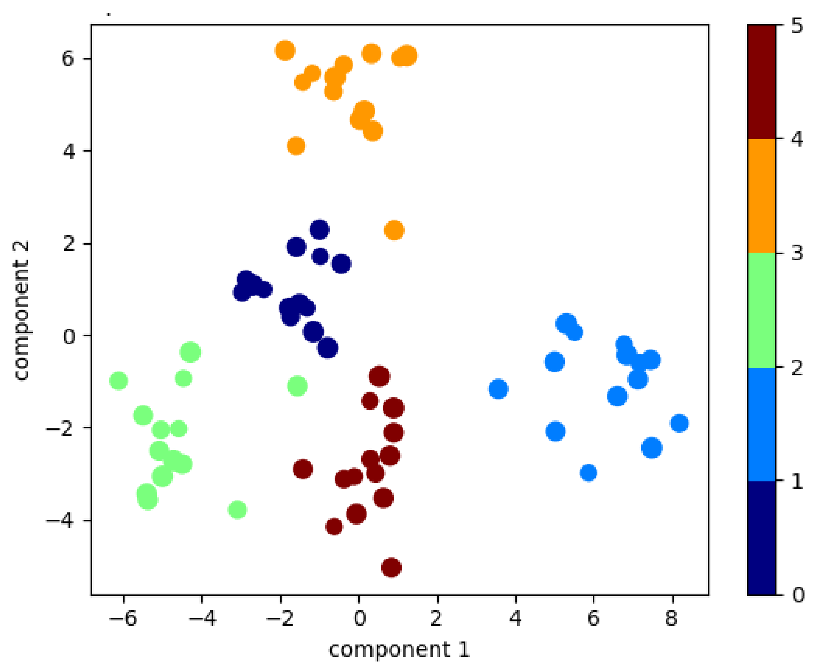

Figure 4.11: LDA plot for LPC coefficients in experiment 2

There is no overlapping although the clusters are not very tight. Since the clusters are not very well defined hence this feature has suffered low accuracy but it has potential for improvement in case where pitch information of different instruments is also different.

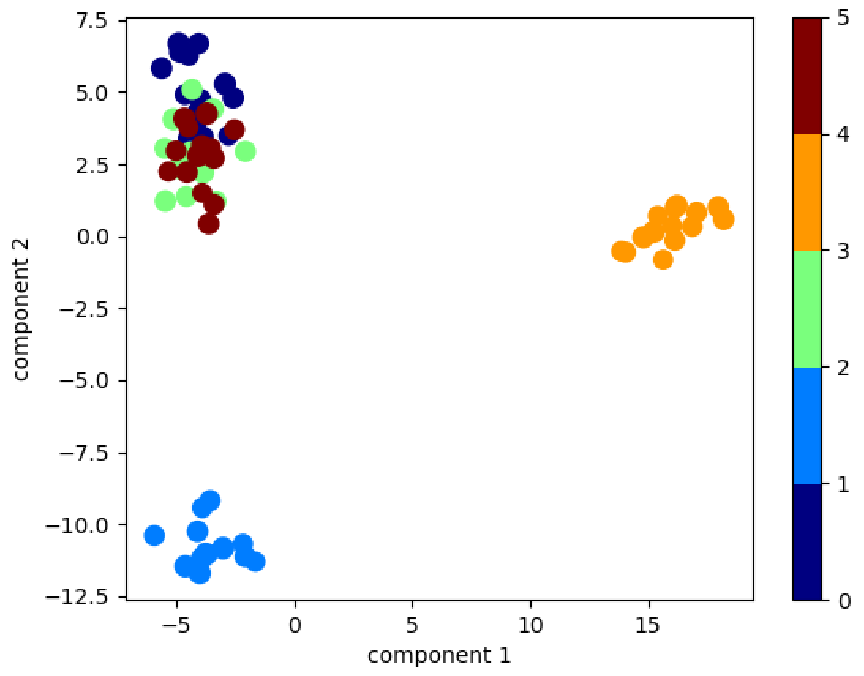

Figure 4.12: LDA plot for Centroid in experiment 2

In this case maximum overlapping is observed. The three classes (1,5&3) are overlapped in a way that is will be very hard to distinguish them. The accuracy of centroid has decreased compared to previous case and the reason it, brightness is not a unique characteristic of sound and Centroid is not a robust feature.

Experiment 3.

The third experiment is carried out with ten instruments which are Organ, French-horn, Cello Clarinet, Tambourine Drum Flute, Trumpet, Violin, Piano. For each instrument or class 15 samples were taken during training and 3 samples were taken for testing so a total of 150 samples were taken during train phase and 30 samples were used in test phase to check the accuracy and robustness of the system. The experiment was carried out with six features and four classifiers as specified in the previous section. The features are MFCC, CC, LPC coefficients, Centroid, HPCP and Pitch salience. The classifiers are K-NN, ANN, Random Forest and SVM. The pitch and dynamics are random.The result of experiment 3 is tabulated below.

TABLE 4.3: Tabulation of results of third experiment

	PITCH SALIENCE	HPCP	CENTROID	LPC	MFCC	CC
K-NN	18	14	14	17	24	25
SVM	12	17	15	15	23	23
Random Forest	20	19	20	19	22	23
ANN	12	19	18	17	25	27

The table portrays how many test samples are classified correctly amongst the 30 test samples used in this experiment.
This shows that the accuracy was maximum in case of CC followed by MFCC when compared with ANN and KNN. These two features have highest general accuracy compared to others. The accuracy for Centroid and Pitch Salience is more than HPCP and LPC coefficients by one vote. It proves the robustness of CC and MFCC again but CC has maintained maximum accuracy throughout along with least overlapping and best clustering features until now.

The LDA plot of MFCC and CC for ten instruments are observed, compared and studied as these two are the best features proven by now so the best amongst these two needs to be decided.
Class 1 represents Organ class
Class 2 represents French-Horn class
Class 3 represents Cello class
Class 4 represents Clarinet class
Class 5 represents Tambourine class
Class 6 represents Drum class
Class 7 represents Flute class
Class 8 represents Trumpet class
Class 9 represents Violin class
Class 10 represents Piano class

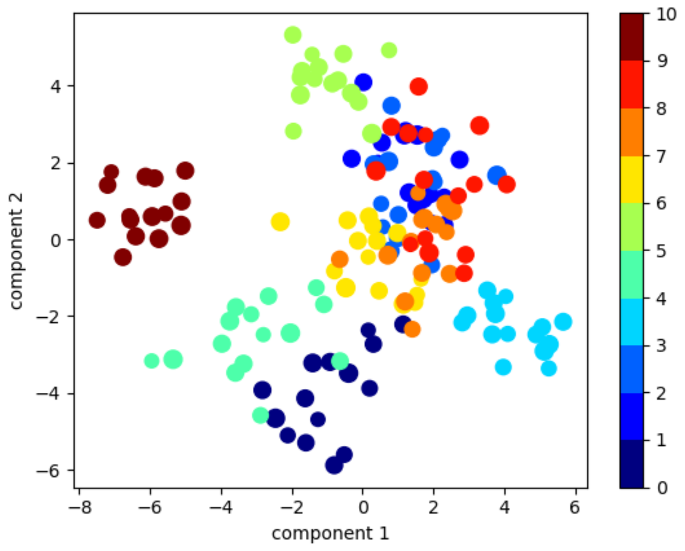

Figure 4.13: LDA plot for MFCC with 10 classes

Some overlapping is observed and the cluster boundary is not well defined so there are chances of false mismatch. Classes 9,1,2,8 are overlapping quite a lot and hence with increase in number of instruments accuracy will fall much steeply.

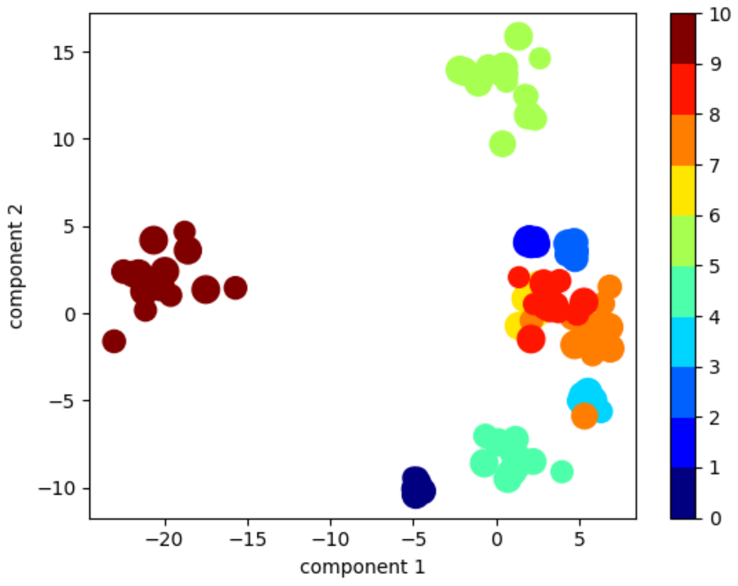

Figure 4.14: LDA plot for CC with 10 classes

Clearly this is the best feature as the overlapping is much lesser compared to MFCC and the clusters are tight and well defined. Only class 9 and 8 has some overlapping otherwise other classes are well distinguished. So, the accuracy will fall much less steeply in compared to MFCC and it has potential for better results.

The plot below shows how each of ten test samples belonging to ten different classes are correctly classified based on the maximum probability obtained when cepstral coefficients are used as features and ANN is used as classifier.

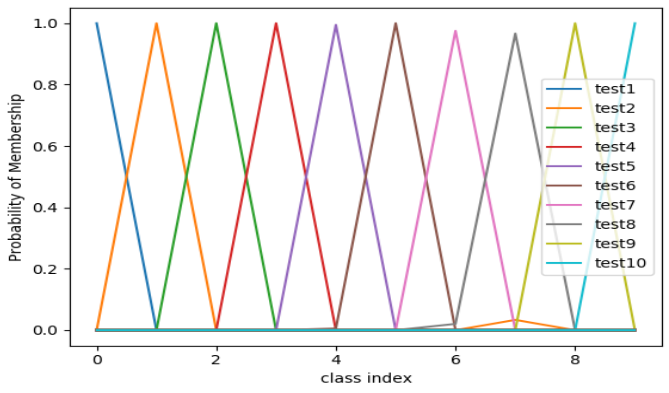

Figure.4.15. Classification plot for test samples

It can be observed that the test samples are matched to the class with which it has maximum probability. In the above case, the accuracy obtained is 100% and hence each test class is classified to its respective train classes. The plot below shows the classification effectiveness of ANN with 30 test samples in last experiment where 27 were correctly classified to their respective classes.

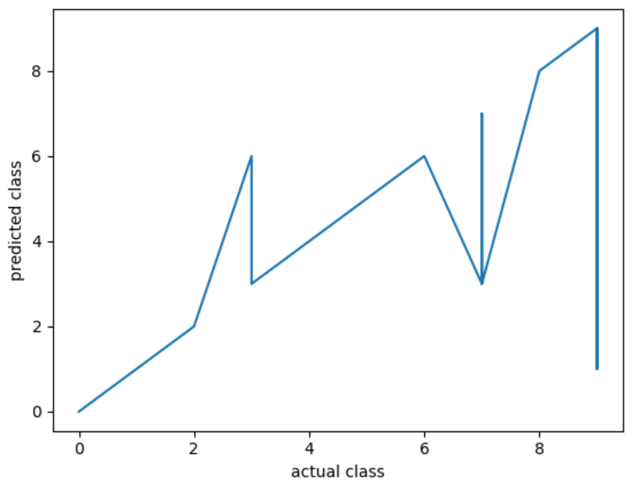

Figure 4.16: Classification effectiveness of CC with ANN of 30 test samples from experiment 3

It can be summarized that CC proves to be the best feature in classifying ten musical instruments of five different families mentioned above. The accuracy was maximum with ANN and KNN but ANN is chosen as the final classifier for better tuning can lead to improved results. The loss in case of CC for all three cases was in order of e^{-6} during training which suggests that the systems were well trained. An analysis based on these results are performed in next chapter.

4.2. Experimentations for OSTI-SI system

4.2.1 Dataset Description

The most early and one of the vital step of designing a system is the collection of relevant, solid, reliable dataset required for the process. In this case two datasets are used to test the performance and efficacy of the proposed approach. The first is the Uyghur dataset. It contains the voice of 200 speakers equally divided into male and female class. It is a subset of THYUG-20 SRE [35]. It contains 3003 recorded voices of 200 people. The environment is noise free. For each speaker, his/her train data is formed by merging his/her voice from multiple trials producing a speech signal with the length varying between 60-150 sec. The test data is kept between 10-20 sec. The male and female datasets are further subdivided into two to conduct the experiment. The second dataset is the Librispeech dataset formed from Librispeech recordings [36]. All samples are clean data i.e. without noise. The train data are of 60-150 sec in length while test data is of 8-10 sec in length. The number of training sample is one per class and the number of testing sample is also one per class. Each training sample is the combined sample of 6 to 7 trials or samples. This was done to make the process faster and avoid complexity. In case of GMM-UBM, the UBM model used here is another set of Librispeech recordings totally different from the previous one. It also consists of a few custom recordings recorded in noise free environment. A total of 50 speech signals are used for training the UBM model and each has a duration of 30-60 sec. The training and testing speeches are different. Hence the dataset is fit for open set text independent speaker identification.

4.2.2 Experimentations

Three sets of experiments are conducted in this section excluding the predictive threshold part. The experiments are conducted with an aim to select the best feature and the classifier in the final system design which acts as input to the process of automatic threshold prediction.

Experiment 1.

Here closed set speaker identification test has been conducted on two features with five classifiers. This test is done in order to observe the efficiency of MFCC and CC with speaker recognition provided no test case is unknown. The test when conducted with MFCC gives the following output.

Table 4.4 Closed set speaker identification with MFCC

	Uyghur male	Uyghur female	Librispeech
KNN	86	91	39
ANN	75	75	35
SVM	81	87	39
Random Forest	90	91	43
Norm Euclidian	90	90	42
GMM+log-likelihood	85	85	39

The test when conducted with CC gives the following output

Table 4.5 Closed set speaker identification with CC

	Uyghur male	Uyghur female	Librispeech
KNN	90	84	43
ANN	78	79	38
SVM	87	84	43
Random Forest	92	90	46
Norm Euclidian	89	91	44
GMM+log-likelihood	85	86	40

It can be observed that all the classifiers are efficient but the most accurate result is predicted by Random Forest and Normalised Euclidian Distance classifier. ANN can provide better result if well-tuned but the problem with ANN is that when the number of train data changes tuning is required which makes the system non-robust and complex. In this case the number of trees selected for Random Forest was 10000 in all cases and the number of neighbours of K-NN was between 1 to 5 with metric as Euclidian distance in all cases. The SVM used here was linear and for ANN four hidden layers were used apart from input and output with Rectifier function at input and hidden layers while SoftMax function was applied at output layer which can be vaguely termed as categorical version of sigmoid function. The loss function was cross entropy and the optimizer was Adam optimizer. The loss in case of CC and MFCC for three cases was in order of 0.05 to 0.06.

Hence observing the result, it can be inferred that, both CC and MFCC are efficient in closed set speaker identification and accuracy and CC even provides a better accuracy. So, to decide the final feature for final classification experiment 2 is conducted.

The plot below shows how each of hundred test samples belonging to hundred different speakers are classified based on the maximum probability obtained when MFCC coefficients are used as features and Random Forest classifier is used as classifier. It is shown for Uyghur female dataset

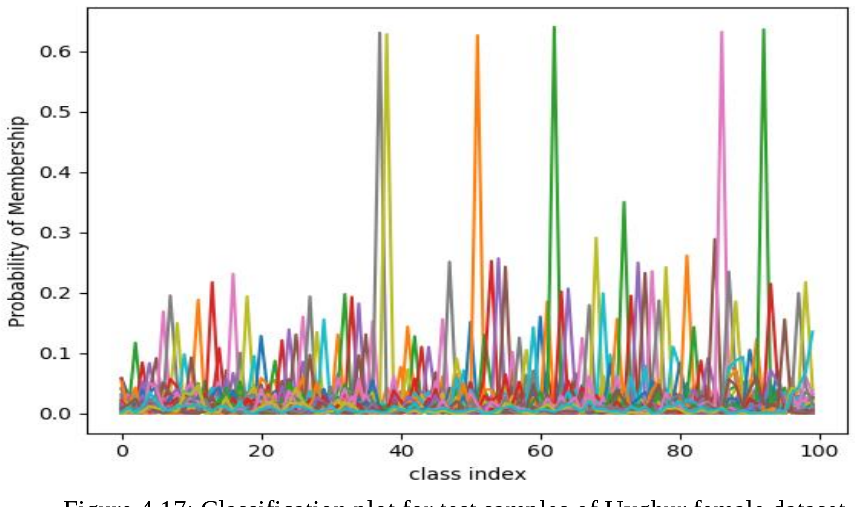

Figure 4.17: Classification plot for test samples of Uyghur female dataset

Since 91 out of 100 were correctly identified, hence the above plot describes that 91 out of 100 speakers have maximum similarity with their respective train voices and likewise they are identified while 9 have maximum similarity with wrong class or speaker and are falsely mismatched to them.

The plot below shows the classification effectiveness of Random Forest with 100 test samples where 91 were correctly identified with their respective classes for the Uyghur female dataset.

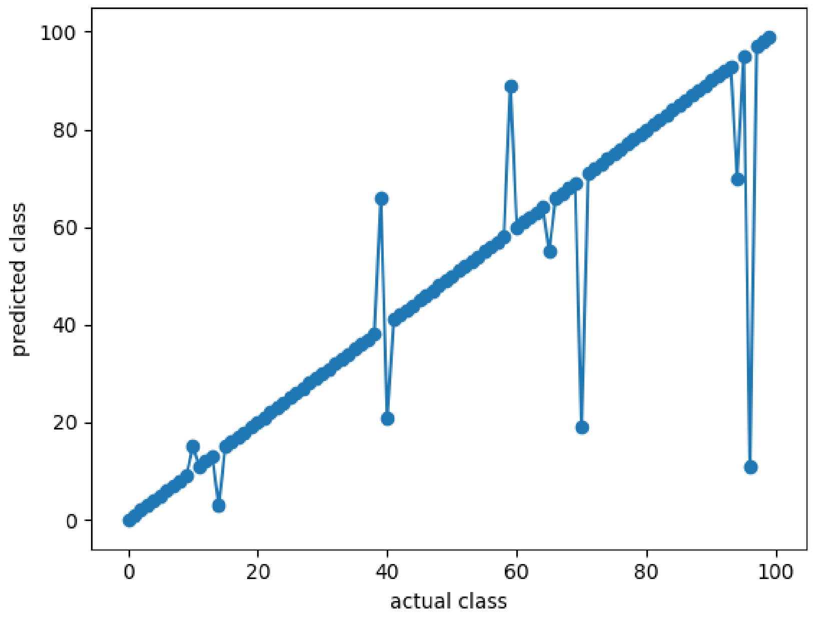

Figure 4.18: Classification effectiveness of Random Forest with 100 samples

<u>Experiment 2.</u>

This experiment is an open set speaker identification test with manual thresholding. The test was conducted with Random forest as classifier for both features which are MFCC and CC. The number of training sample is kept exactly half of the number of test sample. Each training sample represents one class and are made by formed by merging his/her voice from multiple trials and is of length between 60-150 secs. One test sample actually represents one class but since this is an open set test so number of test class is double that of train class and hence number of unknown voices is exactly double than that of number of known voices.

Table 4.6 Open set speaker identification with MFCC and CC

S.NO	Name of Dataset	Number of Train Voices	Number of Test Voices	Feature Used	Number of Correct Identifications		
					TOTAL	CLOSED	OPEN
1	Uyghur Male Dataset	25	50	MFCC	43	20	23
2	Uyghur male dataset	50	100	MFCC	84	39	45
3	Uyghur Male Dataset	25	50	CC	39	19	20
4	Uyghur male dataset	50	100	CC	72	30	42
5	Uyghur Female Dataset	25	50	MFCC	41	20	21
6	Uyghur Female dataset	50	100	MFCC	83	40	43
7	Uyghur Female Dataset	25	50	CC	35	12	23
8	Uyghur Female dataset	50	100	CC	61	21	40
9	Librispeech Recordings	15	30	MFCC	25	12	13
10	Librispeech Recordings	25	50	MFCC	44	21	23
11	Librispeech Recordings	15	30	CC	25	12	13
12	Librispeech Recordings	25	50	CC	38	18	20

The following table tabulates the result obtained from two cases for each dataset. So, six tests are conducted for each feature giving a total of 12 cases. The classifier used here is Random Forest for the time taken by CC with normalized Euclidian distance is very large.

The above result justifies the use of MFCC as the final feature a marked difference in accuracy observed in both cases. The accuracy obtained with MFCC is quite high compared to CC. Also, the threshold follows a pattern in case of MFCC, decreasing with increase in train voices as the sum of probability of likeliness obtained for each test class against all train classes sums up to 1 in Random forest classification. Although the same theory holds for CC yet manual thresholding for these six cases was quite difficult.

Hence MFCC is chosen as final feature. In analysis section, this will be discussed again.

The t-SNE plot of MFCC with 50 voices from train set is given below.

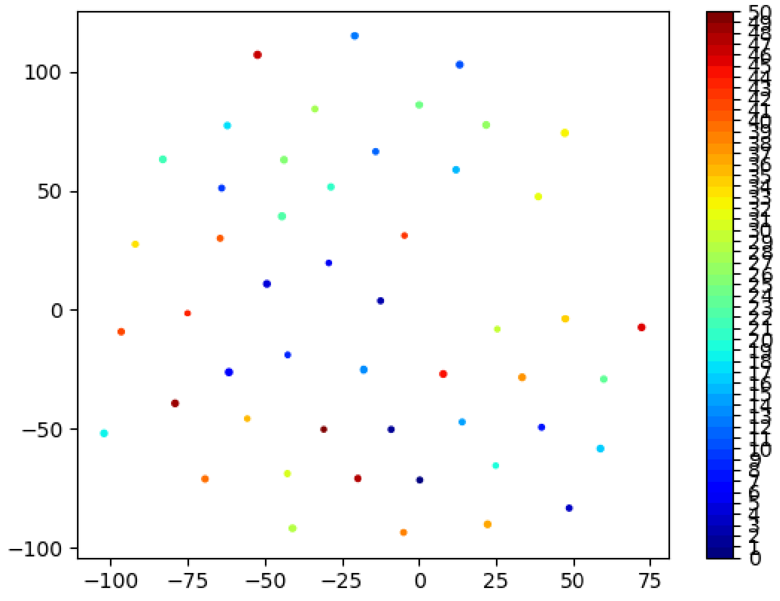

Figure 4.19: t-SNE plot of MFCC features of 50 train samples of Uyghur female dataset

As discussed earlier t-SNE models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modelled by nearby points and dissimilar objects are modelled by distant points with high probability. So, since there is adequate space between each class hence it can be justified that they are well trained with MFCC and hence robust. This plot justifies the result obtained earlier.

Experiment 3.

After obtaining MFCC as the final feature it is required to decide the final classifier. An open set speaker identification test is hence conducted with Random Forest classifier and Normalised Euclidian Distance with MFCC.

The NED yielded better accuracy than GMM-UBM and had several advantages over GMM-UBM the chief one was that unlike GMM-UBM it is independent of UBM model. For GMM-UBM, an appropriate UBM for better accuracy was needed to be chosen. The method also provided a simpler approach to speaker identification, where the threshold can be determined from the rate of its decrease with increase in train data and vice versa.

Due to normalization, the threshold value lies within 0-1 but the metric for thresholding was Euclidian distance while for Random forest it is probability which is produced by the algorithm tends to be more intuitive, simple and lesser prone to error, yet an experiment was conducted in order to observe the opens set accuracies for both of these methods with manual thresholding.

The results for GMM-UBM is also tabulated below.

Table 4.7 Open set speaker identification with Random Forest and NED and GMM-UBM

S.NO	Name of Dataset	Number of Train Voices	Number of Test Voices	Classifier Used	Number of Correct Identifications		
					TOTAL	CLOSED	OPEN
1	Uyghur Male Dataset	50	100	Random Forest	84	39	45
2	Uyghur male dataset	50	100	Normalized Euclidian Distance	73	35	38
3	Uyghur Male Dataset	50	100	GMM-UBM	70	35	35
4	Uyghur Female dataset	50	100	Random Forest	83	40	43
5	Uyghur Female Dataset	50	100	Normalized Euclidian Distance	80	40	40
6	Uyghur Female dataset	50	100	GMM-UBM	73	35	38
7	Librispeech Recordings	25	50	Random Forest	44	21	23
8	Librispeech Recordings	25	50	Normalized Euclidian Distance	42	20	22
9	Librispeech Recordings	25	50	GMM-UBM	40	19	21

From the table above, it can be observed that Random forest has the best accuracy over the other two methods along with other advantages mentioned earlier. Hence the final feature chosen for the final system design is MFCC and the final classifier chosen is Random forest classifier with 10000 trees. Number of trees can be increased but that will take more computational time.

4.2.3 Parameters to Design A System Which Predicts Threshold automatically

The parameters for the design of this system are number of train voices, number of test voices, closed set mean and open set mean. The reasons are discussed in previous chapter.
The sample train dataset used to train the system is given below. Any other dataset can be used provided the four input variables are independent variables and threshold is the dependent variable.

Table 4.8 Training parameters for Random-forest regression

Sl.no	Number of Train data	Number of test data	Mean of closed set probabilities	Mean of open set probabilities	Manual Threshold
1	5	10	30	34.6	28
2	10	20	27.73	23.35	22
3	20	40	22.5	18	16
4	30	60	19.5	15	13
5	34	50	19.4	16.3	13
6	15	30	30	24	21
7	30	50	19.5	16	14
8	35	70	19	14.5	12
9	20	36	22.5	18.5	16
10	5	8	34.6	31	29
11	55	100	18.78	14.06	11
12	40	80	20	14.35	11
13	45	90	19	13.7	12
15	45	80	19	14.35	12

15 sets of information are fed to the system in order to predict future threshold values based on the input variables provided in future. The threshold is observed to be lesser than mean of open set probabilities. After feeding this information when a new set of voices needs to be tested, the number of train data and test data is fed into the system. A subsystem calculates the closed set and open set mean. The four independent variables fed to the threshold predicting system yields the result. The method used to predict threshold is Random forest regression with multiple independent variable.

The chart below tabulates the four-independent variable, predicted threshold and actual threshold. i.e. if the thresholding had been done manually, then the manual threshold selected is regarded as actual threshold.
The result obtained with Uyghur female dataset is tabulated below.

<u>Experiment 1:</u>

Table 4.9 Predicted threshold vs Manual Threshold

Sl.no	Number of Train data	Number of test data	Mean of closed set probabilities	Mean of open set probabilities	Predicted Threshold	Manual Threshold
1	20	36	22.5	18.5	16.0533	16
2	45	80	19	14.35	11.9493	12
3	80	100	18.7	16.3	11.801	11
4	10	20	27.73	23.35	24.1334	22
5	40	80	20	14.35	12.3855	11

It can be observed that the threshold predicted and the manual threshold is quite close to each other. Hence the accuracy of the system will not be largely affected. The final step is to observe the accuracy obtained with manual thresholding vs predicted thresholding.

Experiment 2:

Table 4.10 Accuracy of Open set identification with predicted threshold vs Manual Threshold

S.NO	Name of Dataset	Number of Train Voices	Number of Test Voices	Threshold Used	Number of Correct Identifications		
					TOTAL	CLOSED	OPEN
1	Uyghur Male Dataset	25	50	Manual	43	20	23
2	Uyghur male dataset	50	100	Manual	84	39	45
3	Uyghur Male Dataset	25	50	Predicted	41	20	21
4	Uyghur Male dataset	50	100	Predicted	86	38	48
5	Uyghur Female Dataset	25	50	Manual	41	20	21
6	Uyghur Female dataset	50	100	Manual	83	40	43
7	Uyghur Female Dataset	25	50	Predicted	40	20	20
8	Uyghur Female dataset	50	100	Predicted	82	39	43
9	Librispeech Recording	15	30	Manual	25	12	13
10.	Librispeech Recording	25	50	Manual	44	21	23
11.	Librispeech Recording	15	30	Predicted	25	11	14
12.	Librispeech Recording	25	50	Predicted	42	17	23

After selecting the final feature as MFCC and final classifier as Random forest the Random forest regression model is used to predict threshold automatically undoing human intervention. The accuracy of predicted threshold vs manual threshold is observed for three datasets with two test cases for each.

It can be observed that the accuracy obtained with predicted threshold is not compromised. At it is test proofed on all three datasets hence it can be confirmed that the design is robust and effective.

4.3 Chapter Summary

The experimental results and LDA plots are observed for different cases of musical instrument classification and it is concluded that CC is the best feature for classification of musical instrument and best results are obtained with ANN and K-NN. For final system design ANN is chosen as better tuning an improve the result. So, CC and ANN can classify these ten musical instruments belonging to five different family well enough.
26 out of 30 samples belonging to 10 classes are well classified, while 14 out of 15 samples belonging to 5 classes are well classified under two varied conditions with different samples in train and test set in each case. The number of trees selected for Random Forest was 10000 in all cases and the number of neighbours of K-NN was between 1 to 5 with metric as Euclidian distance in all cases. The SVM used here was linear and for ANN four hidden layers were used apart from input and output with Rectifier function at input and hidden layers while applied SoftMax function at output layer which can be vaguely termed as categorical version of sigmoid function. The loss function was cross entropy and the optimizer was Adam optimizer. The loss in case of CC and MFCC for three cases was in order of e^{-6}.

In the second case of human voice identification it can be concluded that the MFCC is the best feature for open set text independent speaker identification with Random forest classifier and Random forest regression model for threshold prediction. The accuracy obtained was highest compared to GMM-UBM, SVM, K-NN, ANN and Normalized Euclidian distance. The closed set accuracy for CC was very good but accuracy got reduced steeply during open set identification. For Uyghur male dataset closed set accuracy was 90 while open set accuracy for 50 train voices vs 100 test voices was 86 with predicted threshold which actually exceeded the accuracy with manual threshold. For Uyghur female dataset closed set accuracy was 91 while open set accuracy for 50 train voices vs 100 test voices was 82. with predicted threshold. For Librispeech recordings dataset closed set accuracy was 43 out of 50 while open set accuracy for 25 train voices vs 50 test voices was 41 with predicted threshold which is 82 percent. However, the closed set accuracy is 64 percent. Increasing the number of Random forest trees will lead to the betterment of accuracy. The next chapter is devoted to the analysis of above results.

So, the proposed methods which is CC as features or acoustic vectors and ANN for classification in first case or musical instrument classification case and MFCC as features with Random forest as classifier and Random forest regressor with multiple inputs for automatic threshold prediction, proves its ground in this chapter with the observed experimental results and plots.

.

5.ANALYSIS

In this section basically, the results from the previous chapter is discussed and conclusion is drawn

5.1. Analysis of Musical Instrument Classification

As three experiments were conducted in this case so results from all the three experiments will be graphically discussed. After that an overall conclusion will be drawn.

5.1.1. Analysis from experiment 1

The first experiment was carried out with Drum, Flute. Trumpet. Violin. Piano which belongs to five different families of musical instrument mentioned earlier. For each instrument or class 15 samples were taken during training and 3 samples were taken for testing so a total of 75 samples were taken during train phase and 15 samples were used in test phase to check the accuracy and robustness of the system. The samples belonged to random pitch and dynamics. The experiment was carried out with six features and four classifiers as specified in the previous section. The features are MFCC, CC, LPC coefficients, Centroid, HPCP and Pitch salience. The classifiers are K-NN, ANN, Random Forest and SVM. The result obtained showed maximum accuracy with CC. The result is graphically plotted below.

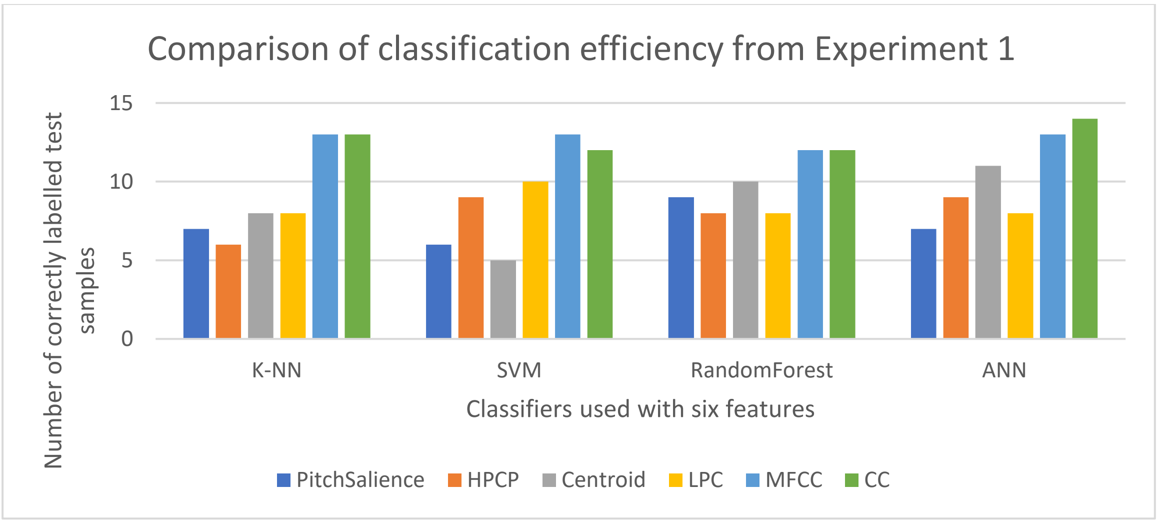

Figure 5.1: Comparison of results obtained at experiment 1

As observed from this plot the maximum accuracy is obtained for CC with ANN. The highest number of correctly labelled test sample is 14 with CC, while it is 13 for MFCC. The other features have much lesser accuracy.
The results show that the accuracy is quite low for Pitch salience peaks, HPCP, LPC coefficients and Centroid. It can be observed that maximum accuracy is obtained against ANN followed by KNN for CC and MFCC. Almost all samples were correctly classified with these two features. In this experiment CC yields maximum accuracy which is followed by MFCC, Centroid, LPC and HPCP and Pitch salience. The accuracy for the HPCP and Pitch salience

are almost same. They obtained maximum accuracy with Random forest while LPC coefficients gained maximum accuracy with SVM and Centroid with ANN. The ANN if better tuned can lead to some improvement.

Since the accuracy obtained for MFCC and CC are very close hence the LDA plots are used to decide the best classifier to observe the degree of uniqueness imparted by each during training phase.

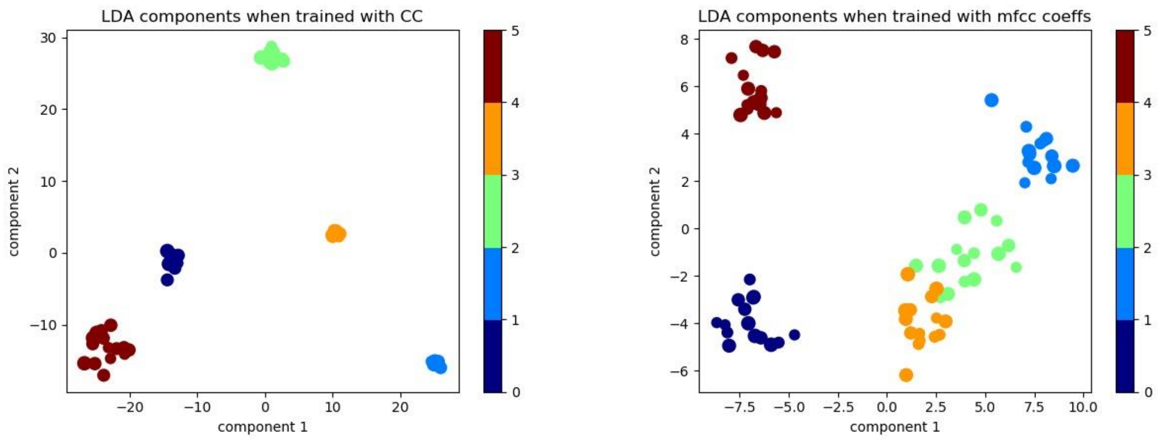

Figure 5.2: LDA plots of CC and MFCC

The above plot clearly shows that the cluster formed by CC are tighter and more far apart from each other than MFCC. There is some overlapping in case of MFCC but there is no overlapping in case of CC. So, from this experiment that CC is the best feature for classification of said five musical instruments under the condition of random pitch and dynamics is concluded.

5.1.2. Analysis from experiment 2

The second experiment was carried out with Drum, Flute. Trumpet. Violin. Piano belonging to five different families of musical instrument. For each instrument or class 15 samples were taken during training and 3 samples were taken for testing so a total of 75 samples were taken during train phase and 15 samples were used in test phase to check the accuracy and robustness of the system. Each musical instrument had random dynamics but a fixed pitch.
Drum- Unpitched
Flute – B4 & B5
Trumpet-C5&C6
Violin- A3& A4
Piano-D4
The pitch information of each instrument is different.

The experiment was carried out with six features and four classifiers as specified in the previous section. The features are MFCC, CC, LPC or LPC coefficients, Centroid, HPCP and Pitch salience. The classifiers are K-NN, ANN, Random Forest and SVM. The result obtained showed maximum accuracy with CC. The result is graphically plotted below.

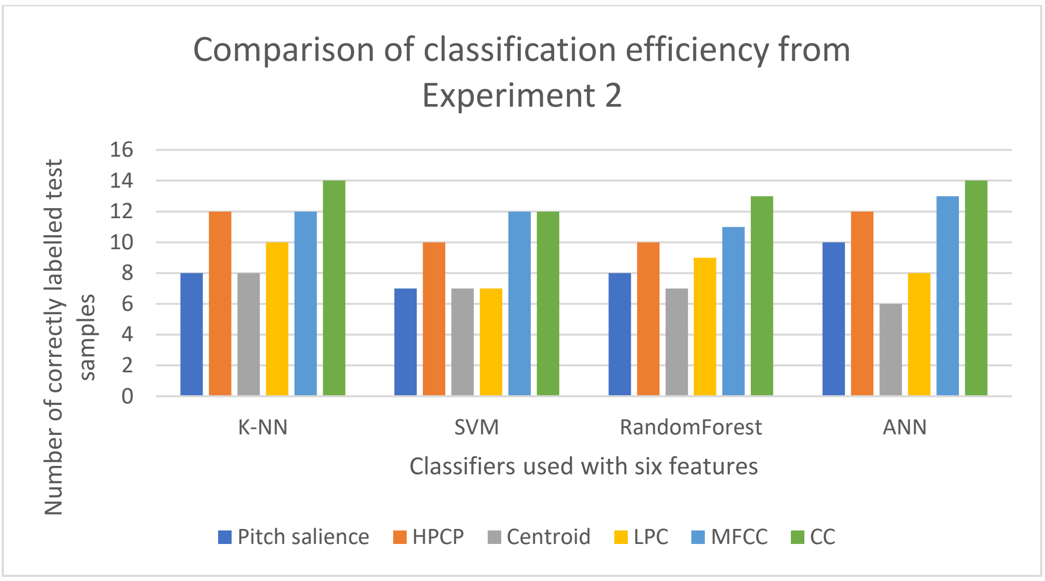

Figure 5.3: Comparison of results obtained at experiment 2

As observed from this plot the maximum accuracy is obtained for CC with ANN. The highest Number of correctly labelled test sample is 14 with CC, while it is 13 for MFCC. The other features have much lesser accuracy.

The results show that the accuracy is quite low for Pitch salience, LPC and Centroid while it is quite good for MFCC, HPCP and CC. It can be observed that maximum accuracy is obtained against ANN followed by K-NN for CC and MFCC. In this experiment CC yields maximum accuracy which is followed by MFCC, HPCP, LPC, Pitch salience and Centroid. The accuracy of HPCP has risen quite steep compared to previous case. As the pitch information was different hence this feature which brings out the dominant pitch classes present in musical instruments. CC and MFCC dominantly gave very good accuracies. In this case also the accuracy of CC was more than MFCC. Accuracy has dropped for Centroid. Pitch salience shows little potential in classification under varied pitch conditions also LPC accuracy has been constant.

Since the accuracy obtained for MFCC and CC are very close hence the LDA plots are used to decide the best classifier to observe the degree of uniqueness imparted by each during training phase.

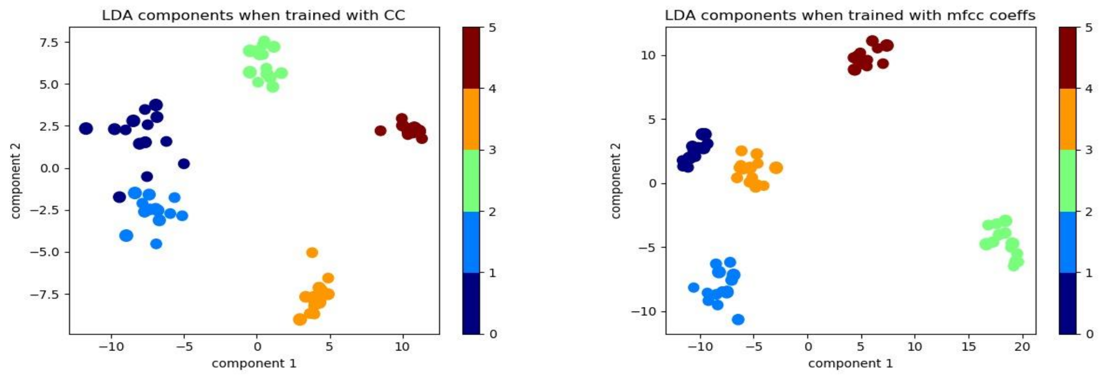

Figure 5.4: LDA plots of CC and MFCC

The above plot clearly shows that the cluster formed by CC are tight and far apart from each other and so does MFCC. In this experiment due to varying pitch characteristics the classification is even better This is due to the fundamental nature of MFCC capturing the frequency information from spectrum along with Timbre by Short time power spectrum analysis. It has potential for even better accuracy. In order to come to a conclusion of best feature in next experiment number of instrument has been increased to ten in order to observe differences.

5.1.3. Analysis from experiment 3

The third experiment is carried out with ten instruments which are Organ, French-horn, Cello Clarinet, Tambourine Drum Flute, Trumpet, Violin, Piano. For each instrument or class 15 samples were taken during training and 3 samples were taken for testing so a total of 150 samples were taken during train phase and 30 samples were used in test phase to check the accuracy and robustness of the system. The experiment was carried out with six features and four classifiers as specified in the previous section. The features are MFCC, CC, LPC, Centroid, HPCP and Pitch salience. The classifiers are K-NN, ANN, Random Forest and SVM. The pitch and dynamics are random. The result obtained showed maximum accuracy with CC. The result is graphically plotted below.

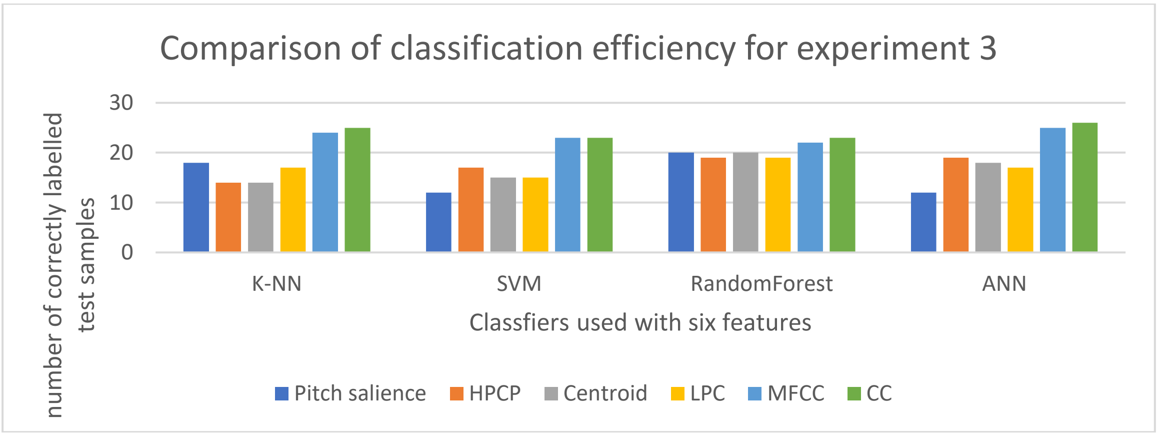

Figure 5.5: Comparison of results from experiment 3.

As observed from this plot the maximum accuracy is obtained for CC with ANN. The highest Number of correctly labelled test sample is 27 with CC, while it is 25 for MFCC. The other features have much lesser accuracy.

This shows that the accuracy was maximum in case of CC which is followed by MFCC when compared with ANN and KNN. These two features have highest general accuracy compared to others. The accuracy for Centroid and Pitch Salience is more than HPCP and LPC by one vote. It proves the robustness of CC and MFCC again but CC has maintained maximum accuracy throughout along with least overlapping and best clustering features until now. Since the accuracy obtained for MFCC and CC are very close hence the LDA plots are used to decide the best classifier to observe the degree of uniqueness imparted by each during training phase.

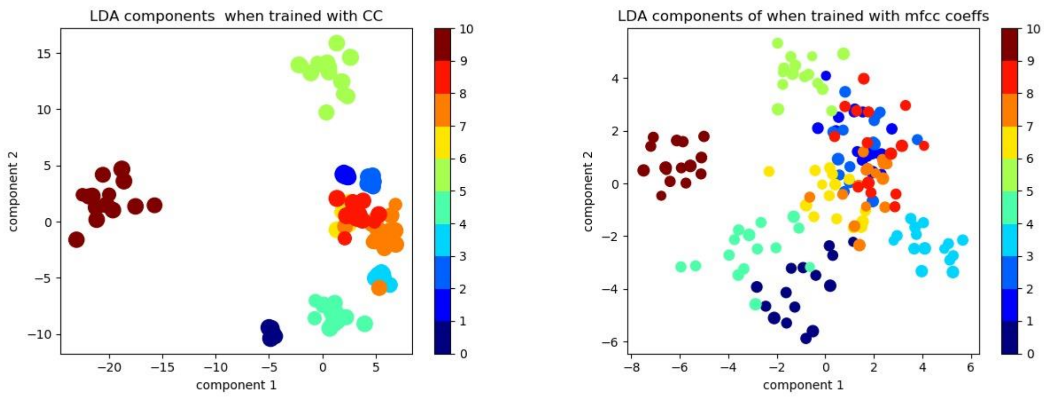

Figure 5.6: LDA plots of CC and MFCC

Some overlapping is observed for MFCC and the cluster boundary is not well defined so there are chances of false mismatch. Classes 9,1,2,8 are overlapping quite a lot and hence with increase in number of instruments accuracy will fall much steeply.
Clearly CC is the best feature as the overlapping is much less compared to MFCC and the clusters are tight and well defined. Only class 9 and 8 has some overlapping otherwise other classes are well distinguished. So, the accuracy will fall much less steeply in compared to MFCC and it has potential for better results.

5.1.4. Overall Analysis

The overall accuracy obtained with all 60 test samples of three experiments (as in all cases test samples were different) for MFCC and CC is given graphically below.

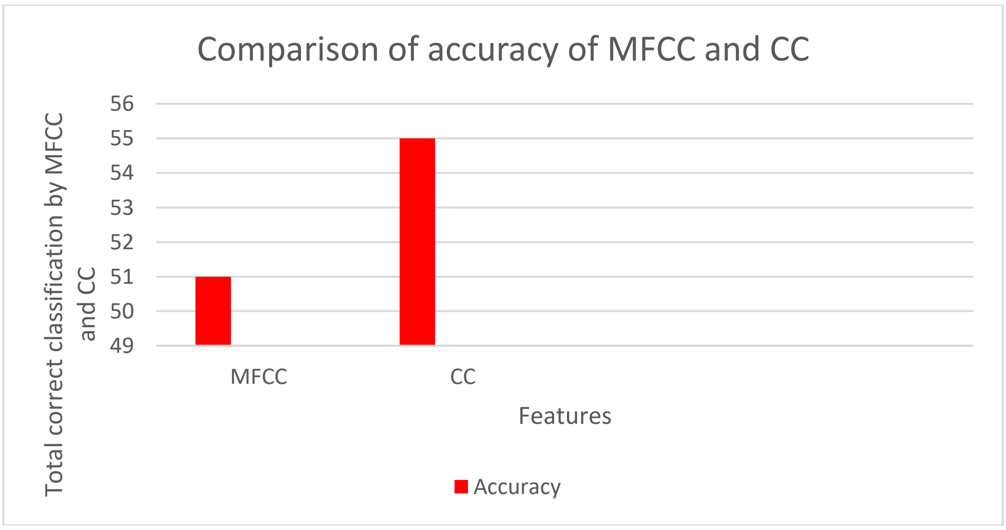

Figure 5.7: Comparison of accuracy of MFCC and CC

Hence the accuracy is given by

$$\text{Accuracy} = \frac{Number\ of\ correctly\ classified\ test\ samples}{Total\ number\ of\ test\ samples} \times 100 \tag{5.1}$$

Hence in first and second case, overall accuracy is 93% while in third case 27 out of 30 samples are correctly classified, hence overall accuracy is 90%. For MFCC, in first two cases overall accuracy is 86% as 13 out of 15 test samples are correctly classified while in third case it is 83% as 25 out of 30 samples were correctly classified by it. The accuracy of proposed approach is larger than all other combinations of MFCC.

Overall the system is experimented with (70+75+150) i.e. 295 train samples and (30+15+15) 60 test samples. The overall accuracy summing the correctly labelled test samples from all three experiments is obtained and it can be observed that 55 out of 60 test samples in three different scenarios was classified correctly by CC giving an accuracy of 91.6% while 51 out of 60 test samples in three different scenarios was classified correctly by MFCC giving an accuracy of 85%.

5.2. Analysis of OSTI-SI

As four experiments were conducted in this case so results from all the three experiments will be graphically discussed. After that an overall conclusion will be drawn.

5.2.1. Analysis of Experiment 1.

Here closed set speaker identification test has been conducted on two features with five classifiers. This test is done in order to observe the efficiency of MFCC and CC with speaker recognition provided no test case is unknown.
Three graphs are plotted to observe the accuracy for each of three datasets with two features i.e. MFCC and CC with five classifiers which are ANN, SVM, KNN, Random forest and Normalized Euclidian Distance. The accuracy obtained with Uyghur male dataset is given below

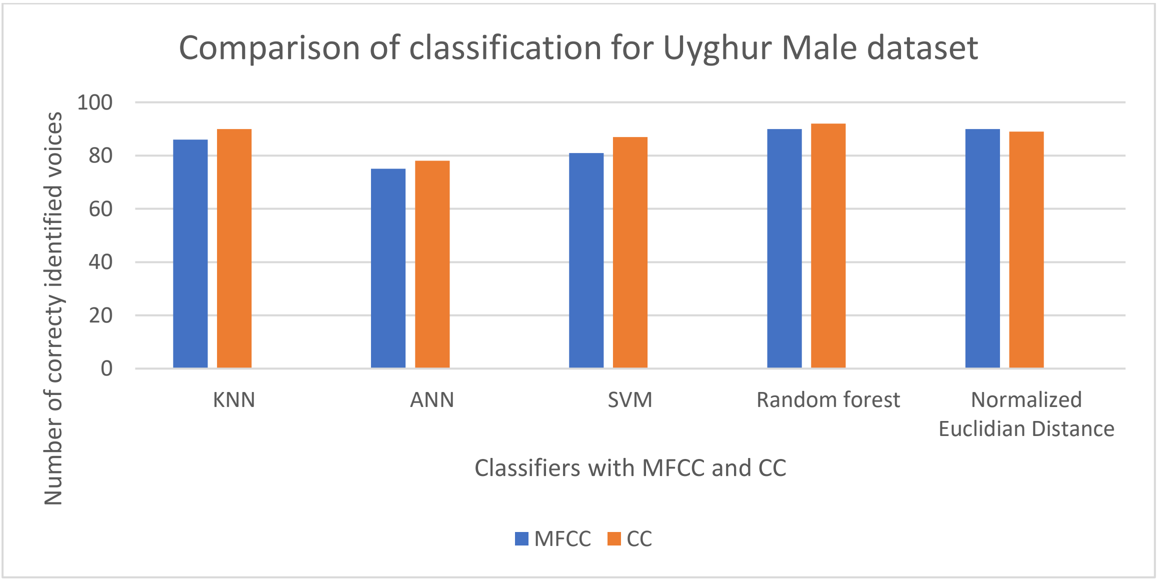

Figure 5.8: Comparison of classification for Uyghur Male dataset

The maximum accuracy is obtained for CC is 92 while with MFCC it is 90 out of 100 voices. The maximum accuracy as observed is obtained from Random forest classifier but all other classifiers worked pretty well making very less difference with maximum accuracy however ANN is not well tuned hence it gives the lowest accuracy in this case. The normalized Euclidian distance also makes very good accuracy.

The accuracy obtained with Uyghur female dataset is given below

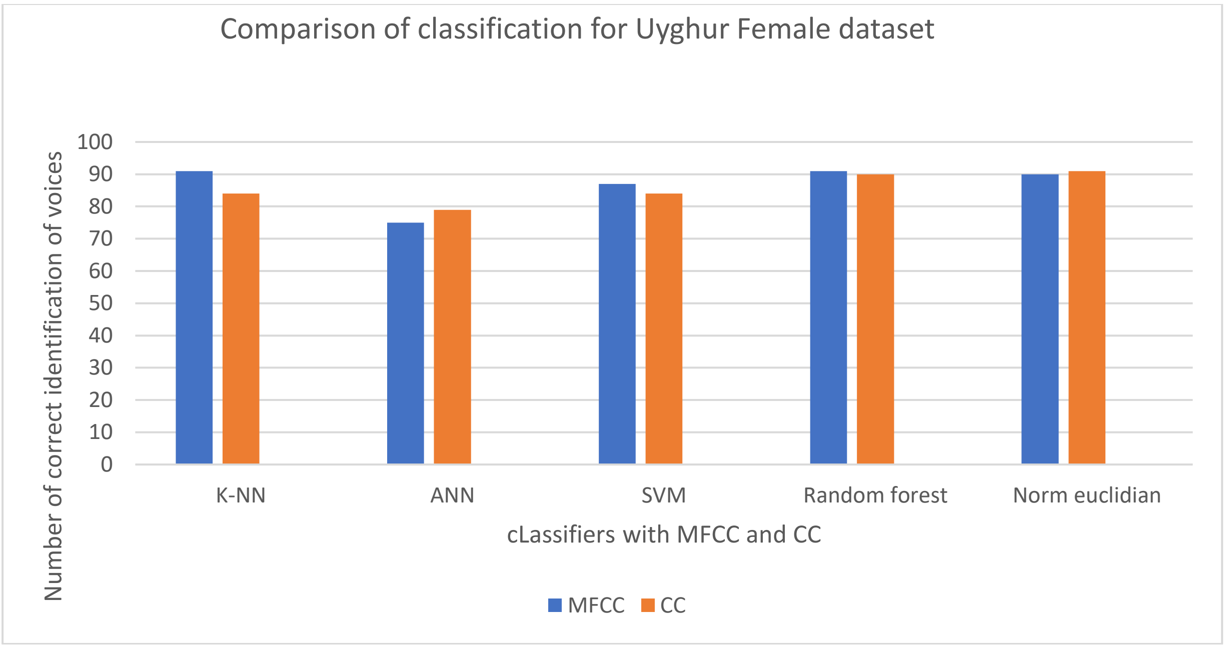

Figure 5.9: Comparison of classification for Uyghur Female dataset

The maximum accuracy is obtained for CC is 91 while with MFCC it is also 91 out of 100 voices. The maximum accuracy as observed is obtained from Random forest classifier for MFCC and normalized Euclidian distance in case of CC but all other classifiers worked pretty well making very less difference with maximum accuracy however ANN is not well tuned hence it gives the lowest accuracy in this case.
The accuracy obtained for Librispeech recording is given below.

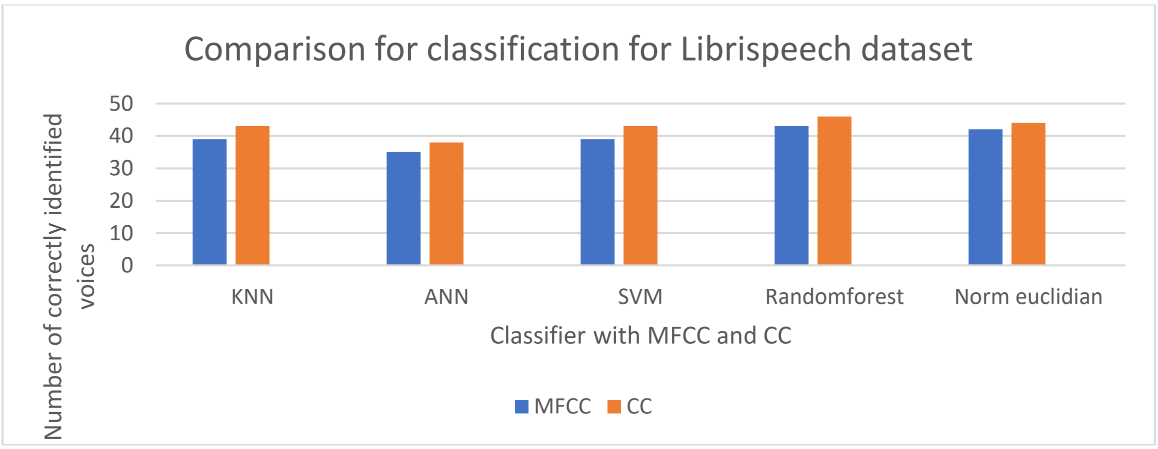

Figure 5.10: Comparison for classification for Librispeech dataset

The maximum accuracy is obtained for CC is 46 while with MFCC it is also 43 out of 100 voices. The maximum accuracy as observed is obtained from Random forest classifier for both features but all other classifiers worked pretty well specially KNN making very less difference with maximum accuracy however ANN is not well tuned hence it gives the lowest accuracy in this case.

5.2.2. Analysis of Experiment 2.

Since both the features produced very good accuracy hence to choose one for the final design of system open set identification test with manual thresholding is performed for both of them. This is also observed for three datasets separately. But graphically plotting the overall accuracy obtained in three cases we get here a graph as follows,

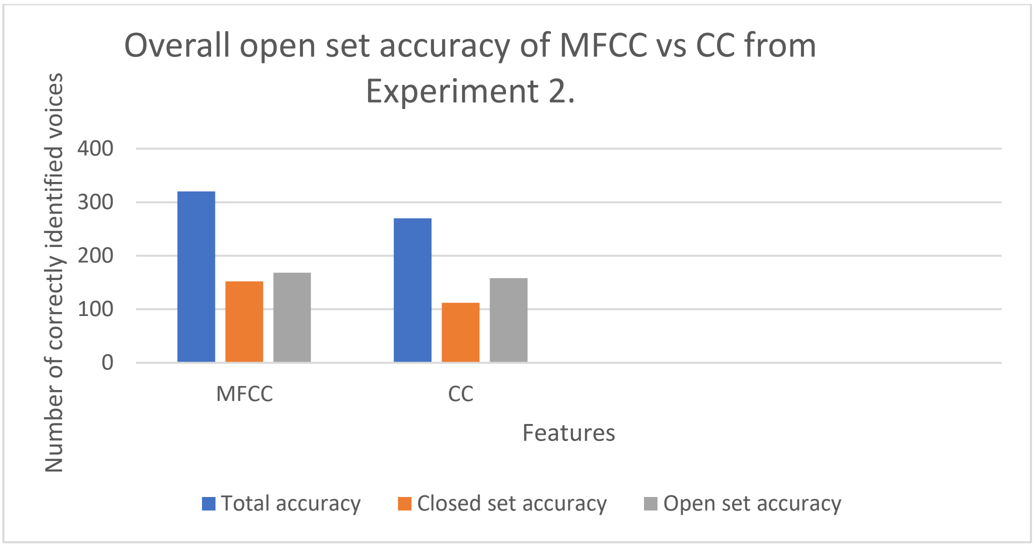

Figure 5.11: Overall open set accuracy of MFCC vs CC from Experiment 2.

The result is the overall accuracy obtained after summing up the results from experiment 2 for each case of MFCC and CC. So out of 380 voices in each case around 320 voices were correctly identified by MFCC with number of true acceptance being 152 out of 190 and true rejection being 168 out of 190 while for CC out of 380 voices in each case around 270 voices were correctly identified by CC with number of true acceptance being 112 out of 190 and true rejection being 158 out of 190.So the result showed that MFCC has around 84% overall accuracy and 80 closed set accuracy and 83%open set accuracy but CC has 71% closed set accuracy with 41% closed set accuracy and 58% open set accuracy which is comparatively much lower than MFCC hence this feature is discarded for open set identification and MFCC is chosen for final system design.
The confusion matrix for MFCC is as follows.

TABLE 5.1: Tabulation of Results of Experiment 2 for MFCC

		Predicted	
		Registered Speaker	Unknown voice
Actual	Registered speaker	152	38
	Unknown voice	22	168

The confusion matrix for CC is as follows.

TABLE 5.2: Tabulation of Results of Experiment 2 for CC

		Predicted	
		Registered Speaker	Unknown voice
Actual	Registered speaker	112	78
	Unknown voice	32	158

5.2.3. Analysis of Experiment 3.

Hence MFCC is chosen as final feature. It is needed to choose a final classifier for the system. Although Random forest has several advantages over Normalized Euclidian distance as it provides a better metric useful for threshold prediction as discussed before yet, the open set accuracy with manual thresholding is checked with both these classifiers. A comparison has been drawn with the conventional method called GMM-UBM to observe the improvement. The plot is the overall accuracy obtained after summing up the results from experiment 3 for each case of Normalized Euclidian distance, GMM-UBM and Random forest classifier.

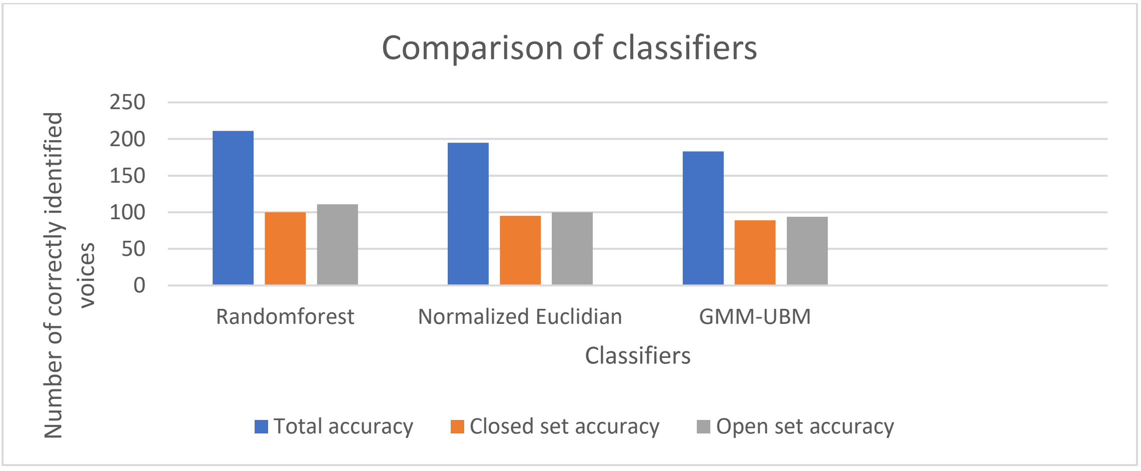

Figure 5.12: Comparison of classifiers

It can be observed that maximum accuracy has been obtained with Random forest classifier. The confusion matrix for each case is given below.
The confusion matrix for Random forest is given by,

TABLE 5.3: Tabulation of Results of Experiment 3 for Random Forest

		Predicted	
		Registered Speaker	Unknown voice
Actual	Registered speaker	100	25
	Unknown voice	14	111

So, the number of correct predictions by Random forest is 211 out of 250 with 84% accuracy and closed set accuracy being 100 out of 125 with 80% accuracy and open set accuracy being 111 out of 125 with 88% accuracy.
The confusion matrix for Normalized Euclidian distance is given by,

TABLE 5.4: Tabulation of Results of Experiment 3 for Normalized Euclidian distance

		Predicted	
		Registered Speaker	Unknown voice
Actual	Registered speaker	95	30
	Unknown voice	25	100

So, the number of correct predictions by Random forest is 195 out of 250 with 78% accuracy and closed set accuracy being 95 out of 125 with 76% accuracy and open set accuracy being 100 out of 125 with 80% accuracy.
It is less than Random forest.

The confusion matrix for GMM-UBM is given by,

TABLE 5.5: Tabulation of Results of Experiment 3 for GMM-UBM

		Predicted	
		Registered Speaker	Unknown voice
Actual	Registered speaker	89	36
	Unknown voice	31	94

So, the number of correct predictions by Random forest is 183 out of 250 with 73% accuracy and closed set accuracy being 89 out of 125 with 71% accuracy and open set accuracy being 94 out of 125 with 75% accuracy. It is less than Random forest and Normalized Euclidian distance. Actually, the latter was proposed by us during the first phase of our research. Also, it provides a better metric for classification than Normalized Euclidian distance.

So, the final classifier is chosen as Random forest classifier with 10000 Decision trees.

5.2.4. Analysis of Experiment 1 from 4.2.3.

The threshold is predicted by Random forest regression algorithm by the four independent variables as discussed before. Here the difference between manual threshold and predicted threshold value is observed graphically.

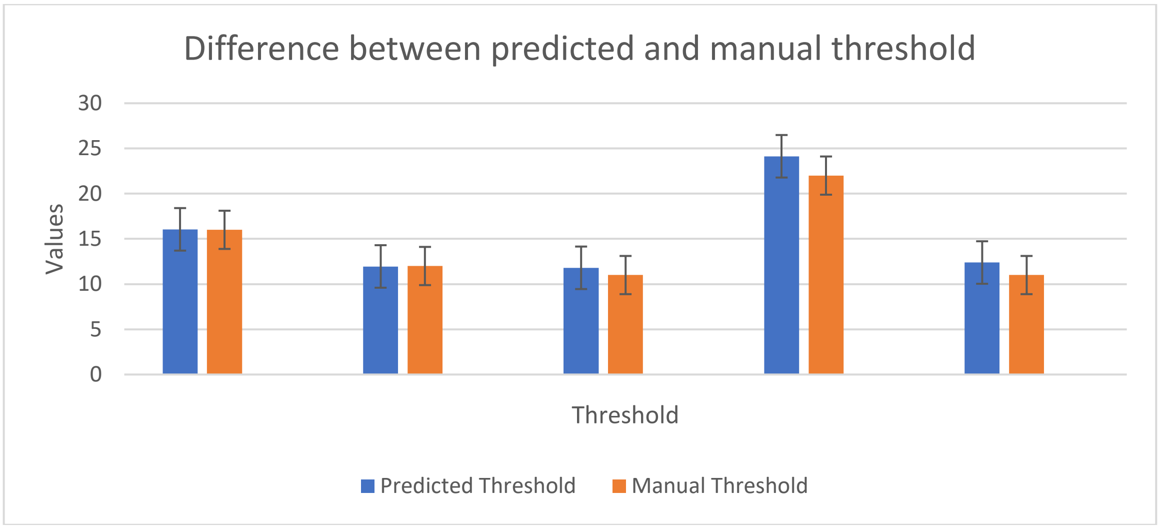

Figure 5.13: Difference between predicted and manual threshold

It can be observed that the error between manual and predicted threshold is very less. Hence it is certain that it will hamper the accuracy less.

5.2.5. Analysis of Experiment 2 from 4.2.3.

However, to be sure an experiment is performed to observe the closed set and open set accuracy with predicted and manual threshold. The plot is the overall accuracy obtained after summing up the results from experiment 4 for each case of predicted threshold and manual threshold.

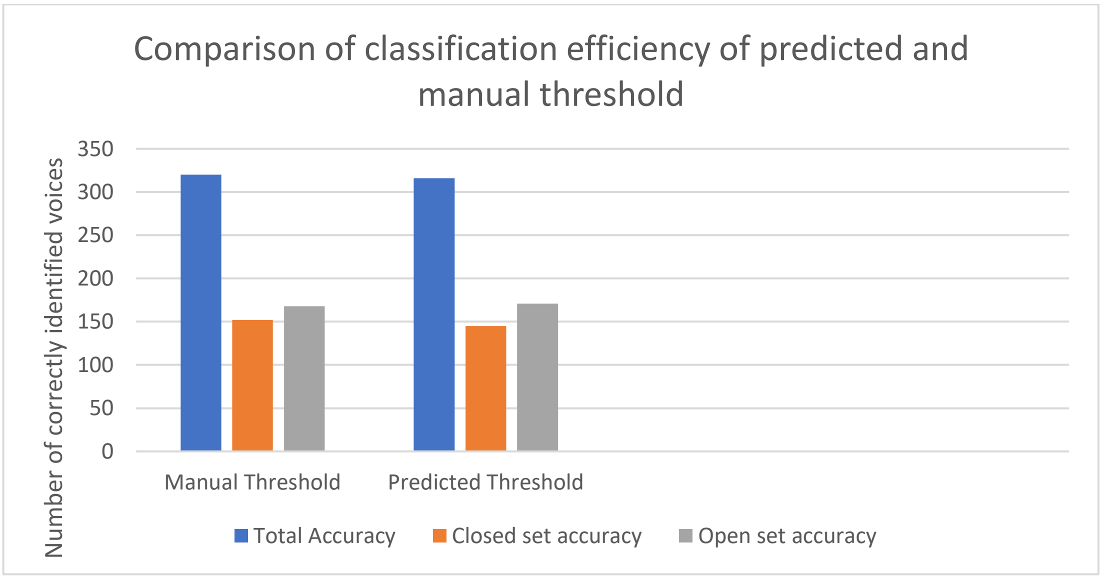

Figure 5.14: Comparison of classification efficiency of predicted and manual threshold

The total accuracy obtained for manual threshold and the total accuracy obtained by predicted threshold has very less difference.

The confusion matrix of open set text independent speaker identification with manual threshold is given below.

TABLE 5.6: Tabulation of Results of Experiment for manual threshold of OSTI-SI

		Predicted	
		Registered Speaker	Unknown voice
Actual	Registered speaker	152	38
	Unknown voice	22	168

So out of 380 voices in each case around 320 voices were correctly identified with manual threshold with number of true acceptance being 152 out of 190 and true rejection being 168 out of 190.So the result showed that this system has around 84% overall accuracy and 80% closed set accuracy and 83%open set accuracy. The confusion matrix of open set text independent speaker identification with predicted threshold is given below.

TABLE 5.7: Tabulation of Results of Experiment for predicted threshold of OSTI-SI

		Predicted	
		Registered Speaker	Unknown voice
Actual	Registered speaker	145	45
	Unknown voice	19	171

So out of 380 voices in each case around 315 voices were correctly identified with predicted threshold with number of true acceptance being 145 out of 190 and true rejection being 171 out of 190.So the result showed that this system has around 83% overall accuracy and 77% closed set accuracy and 90%open set accuracy. The accuracy overall has been down by 1%. The trade-off between false acceptance and false rejection can be done by selecting likewise threshold during train set creation.

5.2.6. Overall Analysis.

The overall analysis is drawn between GMM-UBM, Normalized Euclidian distance with manual thresholding and Random forest classification with automatically predicted threshold by the system for 50 train data vs 100 test data case of each of two dataset of Uyghur male and Uyghur female and 50 train data vs 100 test data case for Librispeech recording is given below. The total accuracy comparison for all three methods with three datasets is given below.

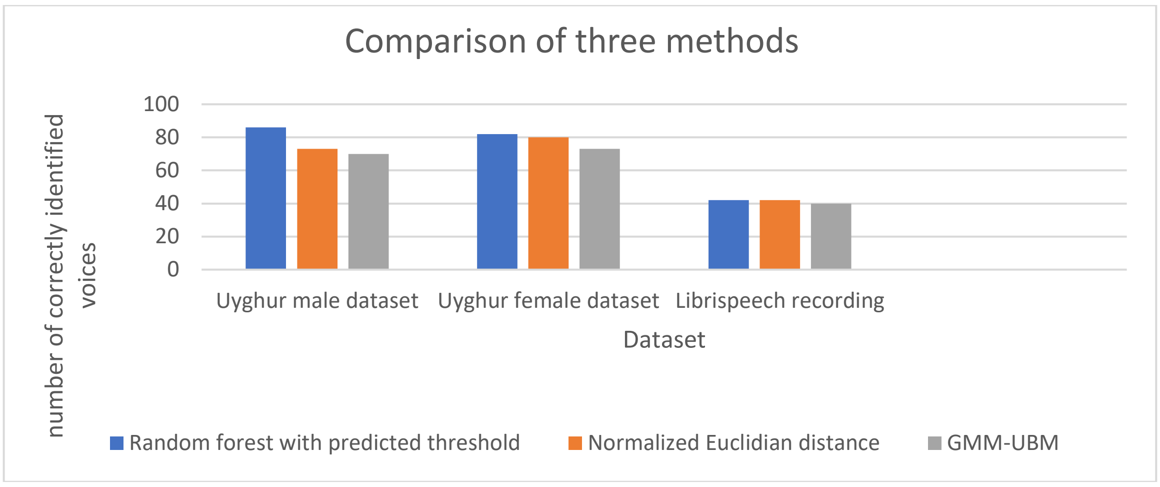

Figure 5.15: Comparison of three methods

It can be observed than GMM-UBM has the least accuracy. The accuracy is more for Random forest classification with automatically predicted threshold in first two cases and in third case they are equal. Hence the method competes well against those designed with manual threshold it is hence proved here. It is better than conventional GMM-UBM and also previously proposed Normalized Euclidian distance with manual thresholding.

5.3. Contributions and Improvements over earlier approaches

The goal for research is to improve on existing works and contribute something that is new and can be implemented in real world applications. In this work too, improvement on existing methods has been made, new ideas have been proposed which will ease the process and make real time applications convenient.
The contribution made in musical instrument classification are described below with graphs and brief explanations.

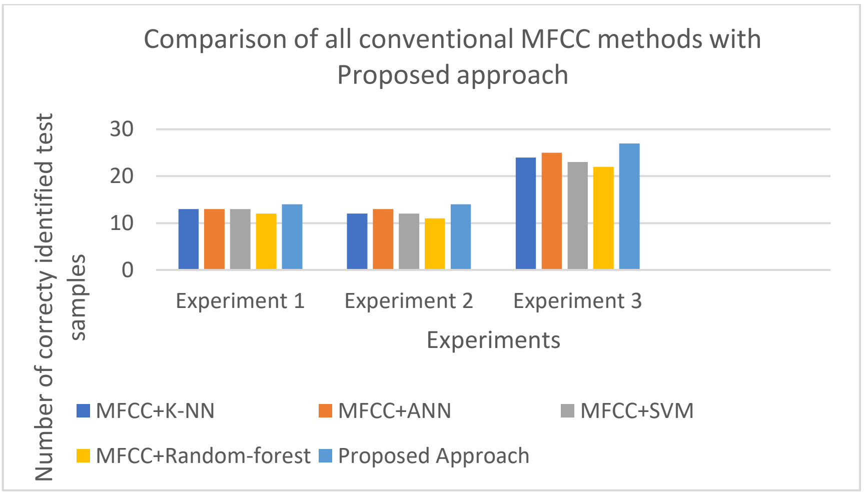

Figure 5.16: Comparison of conventional MFCC approaches with Proposed approach

1. The graph above shows how the proposed approach which uses CC as feature with 40 coefficients and ANN as classifier gives us an accuracy which surpasses all conventional approaches made with MFCC which is one of the most pre-dominant and popularly used

feature in musical instrument classification nowadays as can be observed from the previous works [4], [23], [6], [37].These the better accuracy along with robustness is evident from multiple experimentations which uses different data with variety pitch and dynamics each time.

2. The approach is better than [2] as although the accuracy obtained with it was great but as CNN was used, it would require large amount of train and test data and would fail in situations where data is limited. Since availability of data still now is less, hence the approach faces limitations which is not the case with our proposed approach. It can provide very good accuracy (90%-93%)with less data and the computational speed is much lower than CNN based approaches.
3. The method used in [4] is MFCC + SVM, while in [6] it is MFCC + K-NN. In [23] they used MFCC and timbral audio descriptors that includes spectral centroids and in [37] they used MFCC +ANN. It can be observed from fig (5.16) that the proposed approach gave us better accuracy than all others.

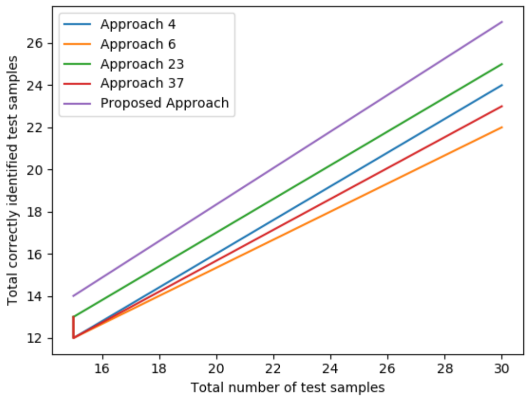

Figure5.17: Comparison of earlier approaches with proposed approach

4. The above figure exhibits the number of test samples correctly predicted by other methods and by our proposed approach. It is found to be maximum hence it has definitely increased the accuracy.
5. The different deviation observed with different approaches and proposed approach is given below. The classification effectiveness of different approaches vs proposed approach for experiment 3 with 30 test samples and 150 train samples given below shows how the predicted class deviates from actual class in different situations. The deviation for proposed approach is least.

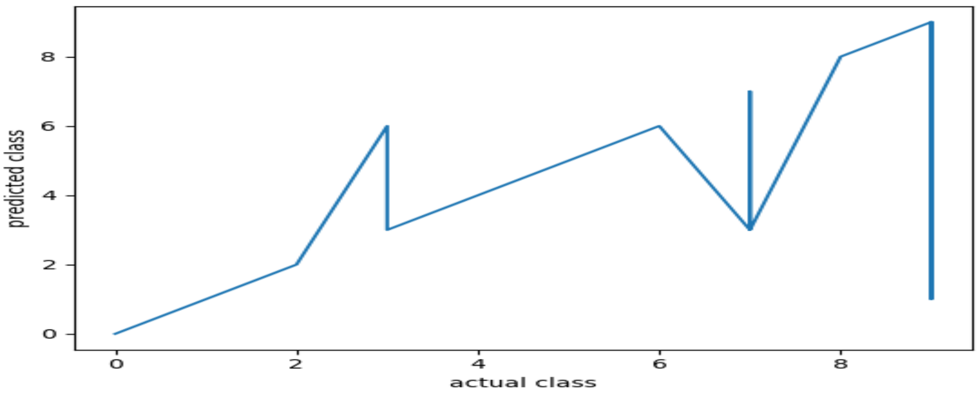

Figure 5.18: Classification effectiveness of proposed approach for experiment 3

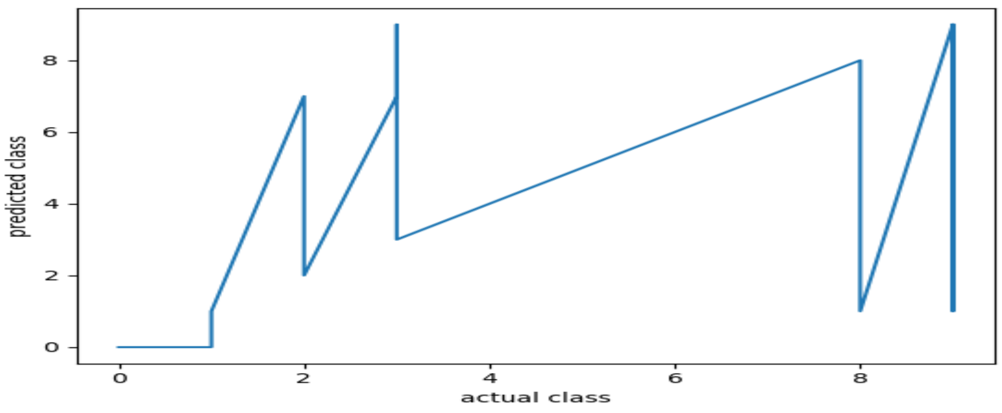

Figure 5.19: Classification effectiveness of approach [6] for experiment 3

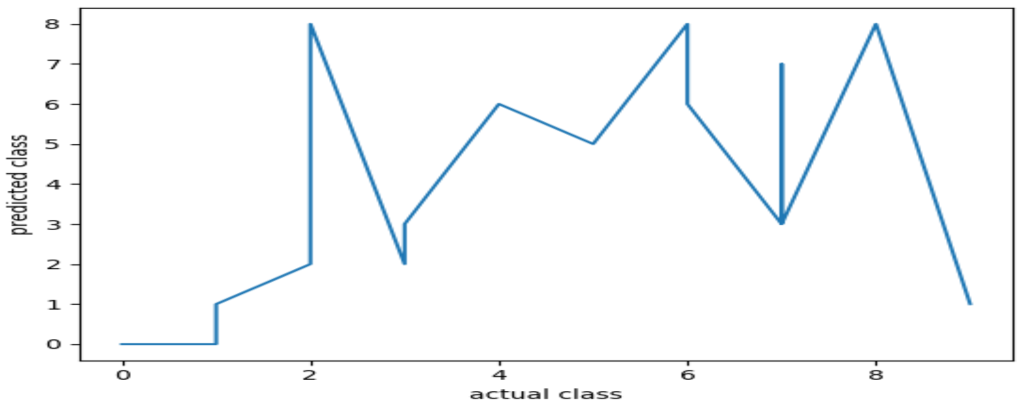

Figure 5.20: Classification effectiveness of approach [4] for experiment 3

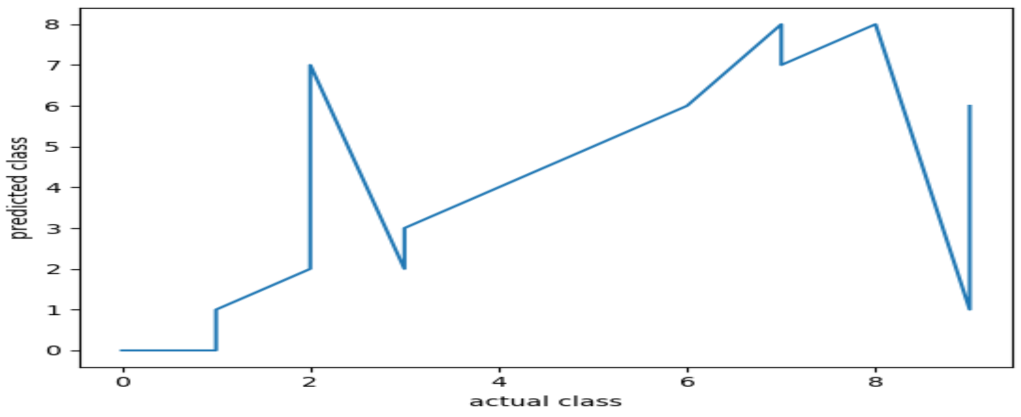

Figure 5.21: Classification effectiveness of approach [23] for experiment 3

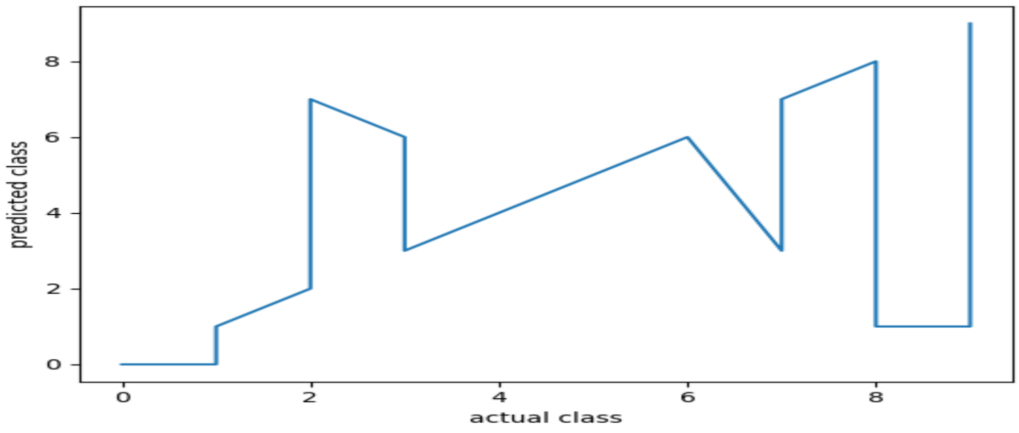

Figure 5.22: Classification effectiveness of approach [37] for experiment 3

6. The decrease of accuracy with increase in number of classes in less in case of CC than MFCC as the overall the system is experimented with (70+75+150) i.e. 295 train samples and (30+15+15) 60 test samples. The overall accuracy summing the correctly labelled test samples from all three experiments obtained is 55 out of 60 test samples in three different scenarios by CC giving an accuracy of 91.6% while 51 out of 60 test samples in three different scenarios was classified correctly by MFCC giving an accuracy of 85%. The drop of accuracy is very less compared to MFCC.
7. The CC is actually better than other features like LPC coefficients, HPCP, MFCC, Pitch salience and Centroid.
8. From the LDA plots it is evident that the train samples were well trained with CC and the feature was able to distinguish or separate then better than any other features bringing their uniqueness.
9. The rate of decrease of accuracy with increase in number of musical instruments is expected to decrease less steeply than that of MFCC as less overlapping is observed even with increase in instruments in third experiment.
10. In this study a comparative analysis of six different kind of features with four powerful machine learning algorithms is drawn.

The contribution and improvement over earlier approaches achieved for OSTI-SI i.e. open set text independent speaker identification are described with plots and explanations below.

1. The closed set accuracy obtained was best with CC and Random forest classifier but as it was not useful in open set identification hence that is discarded and MFCC being considered as the best feature for human voice classification of both closed set and open set, a comparative analysis is drawn for MFCC vs classifiers proposed earlier and classifier proposed in this study which is Random Forest for all three datasets. Approach [11] uses Euclidian distance while approach [9], [10], [14], [15], [16] uses GMM approach with MFCC. The proposed approach for closed set is MFCC and Random forest classifier with 10000 decision trees.

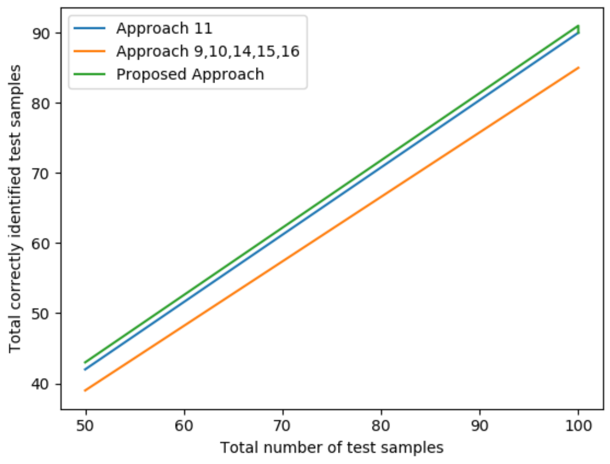

Figure5.23: Comparison of earlier approaches with proposed approach

2. For open set classification MFCC is found to work better than CC and the best classifier chosen was Random Forest for this purpose for it not only provided a better intuitive metric which is probability of likeliness for threshold selection but also has better accuracy compared to normalised Euclidian distance proposed by us in [38] and conventional GMM-UBM and other approaches. Normalised Euclidian distance has better accuracy and advantages over others and has been discussed in [38]. The figure 5.12 actually draws the conclusion and since approaches [12], [13], [17] was based on GMM-UBM and [38] was based on normalised Euclidian hence the proposed approach is better than all these is henceforth proved.
3. The inclusion of predictive threshold is new and the most innovative approach taken in this study as it not only enhances reproducibility, reliability, cost effectiveness undoing complexity, human interference but also does not compromise the accuracy achieved much. The drop of accuracy is very less as observed from figure 5.14 and 5.15 suggests that the accuracy achieved is actually much better than conventional approaches mentioned earlier. The earlier approaches of [12], [13], [17] and [38] did not have the privilege of predicted threshold and since working with very large datasets with no automated threshold generating system is practically impossible, hence this new inclusion enhances and simplifies the process of OSTI-SI.

Hence, the proposed approach for this case is MFCC as feature with Random forest classifier and Random forest regressor with multiple input for automatic threshold prediction has the above-mentioned advantages

5.4. Chapter Summary

The conclusion drawn from analysis of musical instrument classification are as follows

1) CC is the best classifier for musical instrument classification amongst LPC coefficients, HPCP, MFCC, Pitch salience and Centroid as observed from experimental results and LDA plots.
2) The classifier for CC is chosen as ANN although K-NN gave very good result as better tuning of ANN can improve accuracy and best accuracy is obtained with it.

3) From the LDA plots it is evident that the train samples were well trained with CC and the feature was able to distinguish or separate then better than any other features bringing their uniqueness.
4) The rate of decrease of accuracy with increase in number of musical instruments is expected to decrease less steeply than that of MFCC as less overlapping is observed even with increase in instruments in third experiment.
5) The proposed approach is CC as feature with 40 coefficients and ANN as classifier

The conclusion drawn from overall analysis of human voice classification are as follows.

1) Both CC and MFCC are good for closed set speaker identification. Accuracy of CC is more in this case.
2) MFCC is the best feature for open set text independent speaker identification as it provides maximum accuracy and threshold selection for it is easier as the likeliness probability which adds up to one in random forest classification technique follows a pattern and is discriminable rather than CC.
3) The best method of classification is Random forest classification for open set identification as although for closed set normalized Euclidian distance gives good accuracy but accuracy is less than random forest in open set. Moreover, random forest provides a better metric for classification, which is probability of likeliness between each test data with all train classes and also is much faster, flexible and with increase in number of trees up to 1 lakh accuracy further increases. But this is avoided as computational time in that case required would be more.
4) After choosing classifier the model needed to be designed. Random forest regression model with four independent variables is able to predict future threshold values based on the change of likeliness of probability with inclusion of unknown classes given number of known and unknown classes.
5) Random forest regression model is fit for scattered multidimensional data and hence is chosen for this problem.
6) The predicted threshold and manual threshold does not differ much.
7) The drop of accuracy with predicted threshold is negligible compared to the ease of computation, user-friendliness, robustness reliability obtained. It is only 1% for 380 voices in total.
8) The proposed approach for this case is MFCC as feature with Random forest classifier and Random forest regressor with multiple input for automatic threshold prediction.

6.CONCLUSIONS AND FUTURE SCOPES

The two most important and popular field of research in the domain of sound recognition are dealt with, in this study. After observing the experimentations and in-depth analysis following conclusions can be drawn.

6.1 Conclusions

1. This study deals with musical instrument classification which can also be said as musical instrument identification as the details of each musical instrument can be obtained from the system and since it classifies many test samples at a time and classifies them to different groups ,hence it is a special case of identification .The open set text independent speaker identification is the process of identifying several human speakers based on their voices and categorizing them according to the sound sample provided at train set. This is also a kind of classification, but since classes are not as pre-defined as musical instruments hence identification is best named.
2. The features of musical instruments sounds can be categorised into different groups based on the characteristics they deal with.
3. In this study, some of the most prevalent and important features of musical instruments are dealt with and all of them identifies and classifies musical instrument sounds but accuracy varies.
4. The accuracy although is maximum for classification related task, with spectral features but the pitch related features can nevertheless classify sounds.
5. The accuracy of HPCP increases when pitch information for each class is different as it helps the feature to capture pitch class information and since that is different, it hence categorizes sounds based on that and hence the accuracy increases. It can be used in situations where two variants of musical instruments are needed to be distinguished.
6. The pitch salience has the potential for giving better accuracy in second experiment of musical instrument classification when pitch information for each class is different as observed from LDA plots, but it did not do very well with this dataset.
7. The centroids carry brightness information and that definitely changes from one sound to another. So, there is no guarantee that two samples will have same or similar brightness hence this feature is non-robust. It acts well in some situations fails in other and hence non-reliable.
8. The LPC coefficients are good but not better than MFCC or CC as the accuracy received with it is much lower compared to others. It acts better than pitch related features though and is more robust than them.
9. The LPC coefficients are spectrum related. It actually predicts the spectrum so the error in prediction reflects itself in loss in accuracy.
10. MFCCs are very good and stable classifier for distinguishing sounds and they are robust too but our aim was to come up with something better than MFCC as the accuracy of MFCC falls with increase in number of classes and since the number of classes in musical instruments are huge, hence this accuracy loss will seriously hamper real time applications output and producibility.
11. CC is the best classifier for musical instrument classification amongst LPC, HPCP, MFCC, Pitch salience and Centroid as observed from experimental results and LDA plots.
12. CC is the best that is featured from its accuracy profile and LDA plots.

13. The number of coefficients used for CC is 40 as it optimizes the system in terms of accuracy, reproducibility and reliability.
14. The classifier for CC is chosen as ANN although K-NN gave very good result as better tuning of ANN can improve accuracy and best accuracy is obtained with it. In cases where tuning needs to be avoided for generality and less complexity-NN is a good alternative.
15. From the LDA plots it is evident that the train samples were well trained with CC and the feature was able to distinguish or separate then better than any other features bringing their uniqueness. It even surpassed MFCC, the commonly and most widely used feature.
16. The rate of decrease of accuracy with increase in number of musical instruments is expected to decrease less steeply than that of MFCC as less overlapping is observed even with increase in instruments in third experiment and this very characteristic will increase the producibility and accuracy of real time applications designed for this field.
17. Both CC and MFCC are good for closed set speaker identification. Accuracy of CC is more in this case. The CC features are not only good for instrumental sound, but also good for speaker recognition is proved in this.
18. MFCC is the best feature for open set text independent speaker identification as it provides maximum accuracy and threshold selection for it is easier as the likeliness probability which adds up to one in random forest classification technique follows a pattern and is discriminable rather than CC.
19. The best method of classification is Random forest classification for open set identification as although for closed set normalized Euclidian distance gives good accuracy but accuracy is less than random forest in open set. Moreover, random forest provides a better metric for classification, which is probability of likeliness between each test data with all train classes and also is much faster, flexible and with increase in number of trees up to 1 lakh accuracy further increases. But this is avoided as computational time in that case required would be more. The addition in number of trees doesn't imply overfitting.
20. Overfitting is avoided and that is evident from the accuracy obtained with three different sets of voice data. If there had been overfitting of train data, accuracy would had been much lower.
21. After choosing classifier the model needed to be designed. Random forest regression model with four independent variables is able to predict future threshold values based on the change of likeliness of probability with inclusion of unknown classes given number of known and unknown classes.
22. Random forest regression model is fit for scattered multidimensional data and hence is chosen for this problem.
23. The predicted threshold and manual threshold does not differ much.
24. The drop of accuracy with predicted threshold is negligible compared to the ease of computation, user-friendliness, robustness reliability obtained. It is only 1% for 380 voices in total.
25. It was seen that the proposed approach for musical instrument classification gave better accuracy than conventional methods with MFCC used in [4], [23], [6] and [37]. It even had advantages over [2] and since these works are very recent, hence the proposed approach seems to be an improvement to this field of study.
26. The proposed approach for open set text independent speaker identification with adaptive threshold is very innovative as the concept of adaptive threshold is very new.
27. The proposed approach not only worked better than the pre-dominant conventional GMM-UBM based approach but also it has added the new mechanism of predicting threshold which undoes human interruption and thereby speeding up the process, making it error free and hassle free, undoing manual labour and hence increasing cost efficient. This concept is totally new int his field of study and is very useful. It also gave better accuracy for closed

set and open set speaker identification better than earlier approaches [9], [10], [11], [12], [13], [14], [15], [16], [17], [31]

28. Hence for musical instrument classification, proposed approach is CC (with 40 coefficients) and ANN while for OSTI-SI with automatic predicted threshold MFCC+ Random-Forest classification + Random-Forest regression is proposed.

Hence the overall conclusion and contributions are summarized above. The next discussion is on future scope of study, which will enhance the proposed approaches mentioned earlier.

6.2 Future Scopes

The future scope related to this study is discussed below.

1. For musical instrument classification only, solo recordings are dealt with. In future the classification of instruments in polyphonic audio signals will be considered.
2. The above mentioned is the most challenging subclass of musical instrument identification task.
3. Genre classification can also be dealt with. It is not very popular zone but a very useful one.
4. Sounds from noisy environment can be used in future studies and classifying them correctly will give birth to several important real time applications related to this field.
5. There is significant decrease of accuracy with increase of number of classes of musical instruments in case of MFCC. The decrease is less steep in CC but not completely eliminated and hence it is needed to bring it down to negligible amount for the number of musical instrument present is huge and for applications like musical instrument information retrieval, the number of classes will be very large and to correctly distinguish them, error should be further reduced.
6. More studies on musical instruments will aid in this process.
7. Studies must include instruments that are not very popular or regional. An attempt to categorize Indian musical instruments had been made but as the data was very limited hence the accuracy achieved although was great but robustness was doubtful and test or result with so less amount of data is not suitable for research conclusions.
8. The dataset for musical instruments available online is limited. Measures can be taken to increase them specifically for classification related tasks not only for audio processing task.
9. For OSTI-SI, the test should be performed on many more datasets to observe its robustness. In this study total of 500 sound samples are dealt with for dataset for research purpose are limited and costly.
10. Test must be performed on noisy data to evaluate the performance of the system.
11. In future, it is desired to collect human voice samples at different health and mental conditions and to create more robust and more efficient system.
12. The adaptive threshold approach proposed should be able to work in other classification scenarios like image classification or video classification with some changes in parameters but that should be tested and then decided.
13. In future, a real time application based on proposed approaches is to be built.
14. More researches on human sound processing and human sound production along with musical instrument sound production will aid in these kinds of studies.

Hence the above points summarize the future scope of this study. This study is hence concluded and two most popular and important research domain in sound recognition has been dealt with and improved in this study as can be observed but there is scope for further improvement as discussed above.

REFERENCES

1. Bormane, D. S. and Dusane, M. “A Novel Techniques for Classification of Musical Instruments,” Journal of Information and Knowledge Management vol.3, no. 10, pp.1–7 ,2013
2. Park, T. and Lee, T. “Musical instrument sound classification with deep Convolutional Neural Network using Feature Fusion approach”. Electronics and Telecommunications Research Institute (ETRI), Republic of Korea,2015. (unpublished)
3. T. A. Anderson, "Musical instrument classification utilizing a neural network," 2017 12th International Conference on Computer Science and Education (ICCSE*)*, Houston, 2017, pp. 163-166.
4. M. S. Nagawade and V. R. Ratnaparkhe, "Musical instrument identification using MFCC," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2017, pp. 2198-2202.
5. Dattatraya Kuralkar, Saurabh Deshmukh, “Study of Audio Descriptors for Specific Musical Instrument Identification in North Indian Classical Music”, International Journal of Science and Research (IJSR), vol 4, no. 12, pp 1518-1520, 2015.
6. R. S. Kothe, D. G. Bhalke and P. P. Gutal, "Musical instrument recognition using k-nearest neighbour and Support Vector Machine," 2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT), Pune, 2016, pp. 308-313.
7. S. D. Patil and P. S. Sanjekar, "Musical instrument identification using SVM, MLP& AdaBoost with formal concept analysis," 2017 1st International Conference on Intelligent Systems and Information Management (ICISIM), Aurangabad, 2017, pp. 105-109.
8. M. Abulaish, “Ontology Engineering for Imprecise Knowledge Management*”*. Saarbrucken, Germany: Lambert Academic, 2008. (unpublished)

9. F. Y. Leu and G. L. Lin, "An MFCC-Based Speaker Identification System," IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), Taipei, 2017, pp. 1055-1062.
10. N. M. AboElenein, K. M. Amin, M. Ibrahim and M. M. Hadhoud, "Improved text-independent speaker identification system for real time applications," Fourth International. Japan-Egypt Conference on Electronics, Communications and Computers (JEC-ECC*)*, Cairo, 2016, pp. 58-62
11. AK. Singh, R. Singh and A. Dwivedi, "Mel frequency cepstral coefficients-based text independent Automatic Speaker Recognition using MATLAB," International Conference on Reliability Optimization and Information Technology (ICROIT), Faridabad, 2014, pp. 524-527.
12. R. A. Sadewa, T. A. B. Wirayuda and S. Saadeh, "Speaker recognition implementation for authentication using filtered MFCC — VQ and a thresholding method," 3rd International Conference on Information and Communication Technology (ICOICT), Nusa Dua, 2015, pp. 261-265.
13. R. Karadaghi, H. Hertlein and A. Ariyaeeinia, "Effectiveness in open-set speaker identification," 2014 International Carnahan Conference on Security Technology (ICCST), Rome, 2014, pp. 1-6
14. F. Y. Leu and G. L. Lin, "An MFCC-Based Speaker Identification System," IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), Taipei, 2017, pp. 1055-1062.
15. D. A. Reynolds, R. C. Rose, “Robust text-independent speaker identification using Gaussian mixture speaker models,” IEEE Transactions on, Speech and Audio Processing, vol. 3, no. 1, pp. 72-83, 1995.
16. D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, “Speaker Verification Using Adapted Gaussian Mixture Models,” IEEE transactions on Digital Signal Processing, vol. 10, no. 1–3, pp. 19-41, 2000.
17. A.Brew and P. Cunningham, "Combining Cohort and UBM Models in Open Set Speaker Identification," Seventh International Workshop on Content-Based Multimedia Indexing, Chania, 2009, pp. 62-67.
18. F. Răstoceanu and M. Lazăr, "Score fusion methods for text-independent speaker verification applications*,"* 6th Conference on Speech Technology and Human-Computer Dialogue (SpeD), Brasov, 2011, pp. 1-6.
19. F. H. Foomany and K. Umapathy, "Classification of music instruments using wavelet-based time-scale features," 2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), San Jose, CA, 2013, pp. 1-4.
20. Sumit Kumar Banchhor and Arif Khan, “Musical Instrument Recognition using Zero Crossing Rate and Short-time Energy”, International Journal of Applied Information Systems (IJAIS)vol.1, no.3, pp-16-19,2012
21. M. Erdal Ozbek, Nalan Ozkurt and F. Acar Savaci,” Wavelet ridges for musical instrument classification”, Journal of Intelligent Information Systems, vol.38, no. 9, pp-241–256,2012

22. S. Essid, G. Richard and B. David, "Instrument recognition in polyphonic music based on automatic taxonomies,"IEEE Transactions on Audio, Speech, and Language Processing, vol. 14, no. 1, pp. 68-80, Jan. 2006.
23. Priyanka S. Jadhav, “Classification of Musical Instruments sounds by Using MFCC and Timbral Audio Descriptors “International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), vol: 3, no. 7, pp-5001 – 5006,2015
24. M. Muller, D. P. W. Ellis, A. Klapuri and G. Richard, "Signal Processing for Music Analysis," in IEEE Journal of Selected Topics in Signal Processing, vol. 5, no. 6, pp. 1088-1110, Oct. 2011.
25. James Bergstra, Norman Casagrande, Bertrand, “Aggregate Features and AdaBoost for Music Classification,” University of Montreal(unpublished)
26. Keith D. Martin and Youngmoo E. Kim,” 2pMU9.Musical instrument identification: A pattern-recognition approach” 136th meeting of the Acoustical Society of America, America,1998, pp-1-12.
27. A. Rosenberg and F. Soong, "Evaluation of a vector quantization talker recognition system in text independent and text dependent modes," ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing, Tokyo,1986, pp. 873-876.
28. R. C. Rose, E. M. Hofstetter and D. A. Reynolds, "Integrated models of signal and background with application to speaker identification in noise," in IEEE Transactions on Speech and Audio Processing, vol. 2, no. 2, pp. 245-257, Apr 1994.
29. J S Pan, Thesis” Improved Algorithms for VQ Codeword Search, Codebook Design and Codebook Index Assignment”, University of Edinburgh (1996). (unpublished)
30. Jialong He, Li Liu and G. Palm, "A discriminative training algorithm for VQ-based speaker identification," IEEE Transactions on Speech and Audio Processing, vol. 7, no. 3, pp. 353-356, May 1999.
31. S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 28, no. 4, pp. 357-366, Aug 1980.
32. Steve Young, Julian Odell, Dave Ollason, ValtchoValtchev, Phil Woodland (1995). The HTK Book, version 2.1, Department of Engineering, Cambridge University, UK.
33. Philharmonia Orchestra Sound Samples (http://www.philharmonia.co.uk/explore/sound_samples)
34. Freesound (https://freesound.org/)
35. A.Rozi, Dong Wang, Zhiyong Zhang and T. F. Zheng, "An open/free database and Benchmark for Uyghur speaker recognition,” International Conference of Oriental COCOSDA held jointly with conference. on Asian Spoken Language Research and Evaluation, Shanghai, 2015, pp. 81-85.
36. V. Panayotov, G. Chen, D. Povey and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audiobooks," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, 2015, pp. 5206-5210.
37. Mukherjee H., Obaidullah S.M., Phadikar S., Roy K. (2017) “SMIL-A Musical Instrument Identification System.” Computational Intelligence, Communications, and Business Analytics: First International Conference CICBA 2017(Springer). Kalyani,2017, pp-129-140.
38. S. Chakraborty and R. Parekh, "An improved approach to open set text-independent speaker identification (OSTI-SI)," 2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, 2017, pp. 51-56.

APPENDIX: Code for the Implementation

Code for cepstral coefficient calculation

cepsanal2.py

```
"""
Created on Sat Apr 28 21:26:23 2018

@author: Shrutisarika
"""

# calculate filterbank features. Provides e.g. fbank and mfcc features for use in ASR applications
# Author: James Lyons 2012
from __future__ import division
import numpy
from python_speech_features import sigproc
from scipy.fftpack import dct
#from scipy.signal import hamming

def ceps(signal,samplerate=16000,winlen=0.025,winstep=0.01,numcep=40,
         nfilt=40,nfft=512,lowfreq=0,highfreq=None,preemph=0.97,ceplifter=22,appendEnergy=True,
         winfunc= numpy.hamming):
    """Compute MFCC features from an audio signal.

    :param signal: the audio signal from which to compute features. Should be an N*1 array
    :param samplerate: the samplerate of the signal we are working with.
    :param winlen: the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
    :param winstep: the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
    :param numcep: the number of cepstrum to return, default 13
    :param nfilt: the number of filters in the filterbank, default 26.
    :param nfft: the FFT size. Default is 512.
    :param lowfreq: lowest band edge of mel filters. In Hz, default is 0.
    :param highfreq: highest band edge of mel filters. In Hz, default is samplerate/2
    :param preemph: apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
    :param ceplifter: apply a lifter to final cepstral coefficients. 0 is no lifter. Default is 22.
    :param appendEnergy: if this is true, the zeroth cepstral coefficient is replaced with the log of the total frame
energy.
    :param winfunc: the analysis window to apply to each frame. By default no window is applied. You can use
numpy window functions here e.g. winfunc=numpy.hamming
    :returns: A numpy array of size (NUMFRAMES by numcep) containing features. Each row holds 1 feature
vector.
    """
    feat,energy = fbank(signal,samplerate,winlen,winstep,nfilt,nfft,lowfreq,highfreq,preemph,winfunc)
    feat = numpy.log(feat)
    feat = dct(feat, type=2, axis=1, norm='ortho')[:,:numcep]
    feat = lifter(feat,ceplifter)
    if appendEnergy: feat[:,0] = numpy.log(energy) # replace first cepstral coefficient with log of frame energy
    return feat

def fbank(signal,samplerate=16000,winlen=0.025,winstep=0.01,
          nfilt=40,nfft=512,lowfreq=0,highfreq=None,preemph=0.97,
          winfunc= numpy.hamming):
    """Compute Mel-filterbank energy features from an audio signal.

    :param signal: the audio signal from which to compute features. Should be an N*1 array
    :param samplerate: the samplerate of the signal we are working with.
    :param winlen: the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
    :param winstep: the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
    :param nfilt: the number of filters in the filterbank, default 26.
    :param nfft: the FFT size. Default is 512.
    :param lowfreq: lowest band edge of mel filters. In Hz, default is 0.
    :param highfreq: highest band edge of mel filters. In Hz, default is samplerate/2
    :param preemph: apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
    :param winfunc: the analysis window to apply to each frame. By default no window is applied. You can use
numpy window functions here e.g. winfunc=numpy.hamming
    :returns: 2 values. The first is a numpy array of size (NUMFRAMES by nfilt) containing features. Each row
holds 1 feature vector. The
        second return value is the energy in each frame (total energy, unwindowed)
    """
    highfreq= highfreq or samplerate/2
    signal = sigproc.preemphasis(signal,preemph)
    frames = sigproc.framesig(signal, winlen*samplerate, winstep*samplerate, winfunc)
    pspec = sigproc.powspec(frames,nfft)
    energy = numpy.sum(pspec,1) # this stores the total energy in each frame
    energy = numpy.where(energy == 0,numpy.finfo(float).eps,energy) # if energy is zero, we get problems with
log

    #fb = get_filterbanks(nfilt,nfft,samplerate,lowfreq,highfreq)
    feat = pspec # compute the filterbank energies
    feat = numpy.where(feat == 0,numpy.finfo(float).eps,feat) # if feat is zero, we get problems with log

    return feat,energy

def logfbank(signal,samplerate=16000,winlen=0.025,winstep=0.01,
          nfilt=40,nfft=512,lowfreq=0,highfreq=None,preemph=0.97):
    """Compute log Mel-filterbank energy features from an audio signal.

    :param signal: the audio signal from which to compute features. Should be an N*1 array
    :param samplerate: the samplerate of the signal we are working with.
    :param winlen: the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
    :param winstep: the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
    :param nfilt: the number of filters in the filterbank, default 26.
    :param nfft: the FFT size. Default is 512.
    :param lowfreq: lowest band edge of mel filters. In Hz, default is 0.
    :param highfreq: highest band edge of mel filters. In Hz, default is samplerate/2
    :param preemph: apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
    :returns: A numpy array of size (NUMFRAMES by nfilt) containing features. Each row holds 1 feature vector.
    """
    feat,energy = fbank(signal,samplerate,winlen,winstep,nfilt,nfft,lowfreq,highfreq,preemph)
    return numpy.log(feat)

def ssc(signal,samplerate=16000,winlen=0.025,winstep=0.01,
        nfilt=40,nfft=512,lowfreq=0,highfreq=None,preemph=0.97,
        winfunc= numpy.hamming):
    """Compute Spectral Subband Centroid features from an audio signal.

    :param signal: the audio signal from which to compute features. Should be an N*1 array
    :param samplerate: the samplerate of the signal we are working with.
    :param winlen: the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
    :param winstep: the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
    :param nfilt: the number of filters in the filterbank, default 26.
    :param nfft: the FFT size. Default is 512.
    :param lowfreq: lowest band edge of mel filters. In Hz, default is 0.
    :param highfreq: highest band edge of mel filters. In Hz, default is samplerate/2
    :param preemph: apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
    :param winfunc: the analysis window to apply to each frame. By default no window is applied. You can use
numpy window functions here e.g. winfunc=numpy.hamming
```

```
    :returns: A numpy array of size (NUMFRAMES by nfilt) containing features. Each row holds 1 feature vector.
    """
    highfreq= highfreq or samplerate/2
    signal = sigproc.preemphasis(signal,preemph)
    frames = sigproc.framesig(signal, winlen*samplerate, winstep*samplerate, winfunc)
    pspec = sigproc.powspec(frames,nfft)
    pspec = numpy.where(pspec == 0,numpy.finfo(float).eps,pspec) # if things are all zeros we get problems

    fb = get_filterbanks(nfilt,nfft,samplerate,lowfreq,highfreq)
    feat = pspec # compute the filterbank energies
    R = numpy.tile(numpy.linspace(1,samplerate/2,numpy.size(pspec,1)),(numpy.size(pspec,0),1))

    return (pspec*R) / feat

def lifter(cepstra, L=22):
    """Apply a cepstral lifter the the matrix of cepstra. This has the effect of increasing the
    magnitude of the high frequency DCT coeffs.

    :param cepstra: the matrix of mel-cepstra, will be numframes * numcep in size.
    :param L: the liftering coefficient to use. Default is 22. L <= 0 disables lifter.
    """
    if L > 0:
        nframes,ncoeff = numpy.shape(cepstra)
        n = numpy.arange(ncoeff)
        lift = 1 + (L/2.)*numpy.sin(numpy.pi*n/L)
        return lift*cepstra
    else:
        # values of L <= 0, do nothing
        return cepstra



def delta(feat, N):
    """Compute delta features from a feature vector sequence.

    :param feat: A numpy array of size (NUMFRAMES by number of features) containing features. Each row holds
1 feature vector.
    :param N: For each frame, calculate delta features based on preceding and following N frames
    :returns: A numpy array of size (NUMFRAMES by number of features) containing delta features. Each row
holds 1 delta feature vector.
    """
    if N < 1:
        raise ValueError('N must be an integer >= 1')
    NUMFRAMES = len(feat)
    denominator = 2 * sum([i**2 for i in range(1, N+1)])
    delta_feat = numpy.empty_like(feat)
    padded = numpy.pad(feat, ((N, N), (0, 0)), mode='edge')   # padded version of feat
    for t in range(NUMFRAMES):
        delta_feat[t] = numpy.dot(numpy.arange(-N, N+1), padded[t : t+2*N+1]) / denominator   # [t : t+2*N+1] ==
[(N+t)-N : (N+t)+N+1]
    return delta_feat
```

Code for implementing CC in train set

trainceps2.py

```
from __future__ import division
import numpy as np
from scipy.io.wavfile import read
from LBG import lbg
#from mel_coefficients import mfcc

import matplotlib.pyplot as plt
import os
from cepsanal2 import ceps

def training(nfiltbank,dir):
    nSpeaker = 150
    nCentroid = 64
    codebooks_ceps = np.empty((nSpeaker,nfiltbank,nCentroid))
    #codebooks_lpc = np.empty((nSpeaker, orderLPC, nCentroid))
    directory = os.getcwd() + dir;
    fname = str()

    for i in range(nSpeaker):
        fname = '/s' + str(i+1) + '.wav'
        print('Now speaker ', str(i+1), 'features are being trained' )
        (fs,s) = read(directory + fname)
        #mel_coeff = mfcc(s, fs, nfiltbank)
        #print(mel_coeff)

        mel_coefs = np.transpose(ceps(s,fs))
        #lpc_coeff = lpc(s, fs, orderLPC)
        codebooks_ceps[i,:,:] = lbg(mel_coefs, nCentroid)
       # codebooks_lpc[i,:,:] = lbg(lpc_coeff, nCentroid)


    #plotting 5th and 6th dimension MFCC features on a 2D plane
    #comment lines 54 to 71 if you don't want to see codebook
    return (codebooks_ceps)
```

Code for implementing CC with neural networks for 5 train classes

annceps-40-5.py

```
"""
Created on Sat Apr 28 23:03:32 2018

@author: Shrutisarika
"""

from __future__ import division
import numpy as np
from scipy.io.wavfile import read
from LBG import EUDistance,lbg
#from mel_coefficients import mfcc
import matplotlib.pyplot as plt
from trainceps2 import training
import os
from python_speech_features import mfcc

from cepsanal2 import ceps
```

```
#from mel_coefficients import mfcc



from matplotlib import offsetbox
from sklearn import (manifold, datasets, decomposition, ensemble,
                     discriminant_analysis, random_projection)
print(__doc__)


import matplotlib as mpl
nSpeaker = 70

nfiltbank = 40
def mfcc_generator(nfiltbank,dirt):
    coef_list = []
    nSpeaker = 5
    nCentroid = 64
    mfcc_gen = np.empty((nSpeaker,nfiltbank,nCentroid))
    #codebooks_lpc = np.empty((nSpeaker, orderLPC, nCentroid))
    directory = os.getcwd() + dirt;
    fname = str()

    for i in range(nSpeaker):
        fname = '/s' + str(i+1) + '.wav'
        print('Now speaker ', str(i+1), 'features are being trained' )
        (fs,s) = read(directory + fname)



        mel_coefs = np.transpose(ceps(s,fs))
        coef_list.append(mel_coefs)
        #lpc_coeff = lpc(s, fs, orderLPC)
        mfcc_gen[i,:,:] = lbg(mel_coefs, nCentroid)
    return (mfcc_gen)

A = 0
list1 = []
list11 = []
list2 = []
(codebooks_mfcc) = training(nfiltbank,'/muss3')
for i in (codebooks_mfcc):
    for j in i:
        A = np.asarray(i).reshape(-1)
    list1.append(A)

k = np.array(list1)
#X = np.append(values = k,arr = np.zeros((4,1280)).astype(int),axis = 0)

(codebooks_test) = mfcc_generator(nfiltbank,'/testddf')
for i1 in (codebooks_test):
    for j1 in i1:
        A1 = np.asarray(i1).reshape(-1)
    list11.append(A1)

k1 = np.array(list11)
outer = []
X_train = np.array(k)
#y = np.array(list(range(nSpeaker)))
```

```
#y = np.append(arr = y,values=np.zeros(25))
X_test = np.array(k1)

for i in range(0,5):
   for j in range(0,14):
       q = i
       outer.append(q)
y = np.array(outer)
"""
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
"""
from sklearn import metrics
#CREATE YOUR CLASSIFIER HERE
# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

# Initialising the ANN
classifier = Sequential()
from keras.utils import to_categorical
#y_binary = to_categorical(y)

# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 570, init = 'uniform', activation = 'relu', input_dim = 2560))

# Adding the second hidden layer
classifier.add(Dense(output_dim = 570, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 570, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 570, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 570, init = 'uniform', activation = 'relu'))
#classifier.add(Dense(output_dim = 588, init = 'uniform', activation = 'relu'))
# Adding the output layer
classifier.add(Dense(output_dim = 70, init = 'uniform', activation = 'softmax'))

# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
onehot = keras.utils.to_categorical(y, num_classes=70)
#model.fit(X_train, onehot, epochs=100, batch_size=1000)
# Fitting the ANN to the Training set
classifier.fit(X_train, onehot, batch_size = 60 , nb_epoch = 100)
# Generate dummy data

#548,50 for 5
# Train the model, iterating on the data in batches of 32 samples
#model.fit(data, one_hot_labels, epochs=10, batch_size=32)
#model.fit(X_train,y, batch_size = 32, nb_epoch = 10)

# Part 3 - Making the predictions and evaluating the model

# Predicting the Test set results
#y_pred = classifier.predict(X_test)
#y_pred = (y_pred > 0.5)


testspeaker = 5
y_pred = classifier.predict_classes(X_test)
```

```
y_predprob = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
"""
cm = confusion_matrix(y,y_pred)
from sklearn import metrics
accurary = metrics.accuracy_score(y,y_pred)
"""
sumok = 0
closedsetcorrectness = 0
sumnotok = 0
y_percent = (y_predprob*100)
y_ted = np.array(list(range(testspeaker)))

listnew = []
y_new = list(y_percent)
nn = 0
fuc = np.append(arr = y_ted,values= y_pred)
fuclist = list(fuc)
for ff in fuclist:

    nn = ff
    for jj in y_new[nn]:
        if jj == max(y_new[nn]):

            maxi = jj
            listnew.append(maxi)
            print('Likelihood is '+str(maxi))
            if maxi>50:
                print(str(ff)+' of test set identifies itself with '+str(fuclist[ff+testspeaker])+' of train set')
                sumok += 1
                if str(ff) == str(fuclist[ff+testspeaker]):
                    closedsetcorrectness+=1

                #print('probably correct')
            else:
                print('No match')
                sumnotok +=1


    if ff == (testspeaker-1):

        print('done')
        break
print(str(closedsetcorrectness)+' correctly predicted and '+str(sumok-closedsetcorrectness)+' wrongly predicted as
FA')
dd = np.array(listnew)
med = np.median(dd)


"""
# Applying LDA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components = 2)
X_lda = lda.fit_transform(X_train, y)
X_testlda = lda.transform(X_test)
##############################################################################
"""
print("Computing Linear Discriminant Analysis projection")
X2 = X_train.copy()
X2.flat[::X_train.shape[1] + 1] += 0.01  # Make X invertible
```

```
X_lda = discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_transform(X2, y)

###########################################################################################
N = 5
cmap = plt.cm.jet
# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]
# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

bounds = np.linspace(0,N,N+1)
norm = mpl.colors.BoundaryNorm(bounds, cmap.N)

print("Computing LDA projection")


scat = plt.scatter(X_lda[:, 0], X_lda[:, 1],c=y,s=np.random.randint(50,100,N),cmap=cmap,norm=norm)
# create the colorbar
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.title('LDA components when trained with CC')
cb = plt.colorbar(scat, spacing='proportional',ticks=bounds)

plt.show()
```

<u>Code for implementing CC with neural networks for 10 train classes</u>
annceps-40.py

```
from __future__ import division
import numpy as np
from scipy.io.wavfile import read
from LBG import EUDistance,lbg
#from mel_coefficients import mfcc
import matplotlib.pyplot as plt
from trainceps2 import training
import os
from python_speech_features import mfcc

from cepsanal2 import ceps

#from mel_coefficients import mfcc



from matplotlib import offsetbox
from sklearn import (manifold, datasets, decomposition, ensemble,
                     discriminant_analysis, random_projection)
print(__doc__)


import matplotlib as mpl
nSpeaker = 150

nfiltbank = 40
def mfcc_generator(nfiltbank,dirt):
    coef_list = []
    nSpeaker = 10
    nCentroid = 64
    mfcc_gen = np.empty((nSpeaker,nfiltbank,nCentroid))
    #codebooks_lpc = np.empty((nSpeaker, orderLPC, nCentroid))
    directory = os.getcwd() + dirt;
    fname = str()

    for i in range(nSpeaker):
        fname = '/s' + str(i+1) + '.wav'
        print('Now speaker ', str(i+1), 'features are being trained' )
        (fs,s) = read(directory + fname)



        mel_coefs = np.transpose(ceps(s,fs))
        coef_list.append(mel_coefs)
        #lpc_coeff = lpc(s, fs, orderLPC)
        mfcc_gen[i,:,:] = lbg(mel_coefs, nCentroid)
    return (mfcc_gen)

A = 0
list1 = []
list11 = []
list2 = []
(codebooks_mfcc) = training(nfiltbank,'/muss3')
for i in (codebooks_mfcc):
    for j in i:
        A = np.asarray(i).reshape(-1)
    list1.append(A)

k = np.array(list1)
#X = np.append(values = k,arr = np.zeros((4,1280)).astype(int),axis = 0)

(codebooks_test) = mfcc_generator(nfiltbank,'/testmuss3')
for i1 in (codebooks_test):
    for j1 in i1:
        A1 = np.asarray(i1).reshape(-1)
    list11.append(A1)

k1 = np.array(list11)






outer = []
X_train = np.array(k)
#y = np.array(list(range(nSpeaker)))
#y = np.append(arr = y,values=np.zeros(25))
X_test = np.array(k1)

for i in range(0,10):
    for j in range(0,15):
        q = i
        outer.append(q)
y = np.array(outer)
"""
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
"""
from sklearn import metrics
#CREATE YOUR CLASSIFIER HERE
# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

# Initialising the ANN
classifier = Sequential()
from keras.utils import to_categorical
#y_binary = to_categorical(y)

# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 600, init = 'uniform', activation = 'relu', input_dim = 2560))

# Adding the second hidden layer
classifier.add(Dense(output_dim = 600, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 600, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 600, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 600, init = 'uniform', activation = 'relu'))
#classifier.add(Dense(output_dim = 670, init = 'uniform', activation = 'relu'))
#classifier.add(Dense(output_dim = 588, init = 'uniform', activation = 'relu'))
# Adding the output layer
classifier.add(Dense(output_dim = 150, init = 'uniform', activation = 'softmax'))

# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
onehot = keras.utils.to_categorical(y, num_classes=150)
#model.fit(X_train, onehot, epochs=100, batch_size=1000)
# Fitting the ANN to the Training set
classifier.fit(X_train, onehot, batch_size = 50, nb_epoch = 100)
# Generate dummy data

#548,50 for 5
# Train the model, iterating on the data in batches of 32 samples
#model.fit(data, one_hot_labels, epochs=10, batch_size=32)
#model.fit(X_train,y, batch_size = 32, nb_epoch = 10)

# Part 3 - Making the predictions and evaluating the model

# Predicting the Test set results
#y_pred = classifier.predict(X_test)
#y_pred = (y_pred > 0.5)


testspeaker = 10
y_pred = classifier.predict_classes(X_test)
y_predprob = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
"""
cm = confusion_matrix(y,y_pred)
from sklearn import metrics
accurary = metrics.accuracy_score(y,y_pred)
"""
sumok = 0
closedsetcorrectness = 0
sumnotok = 0
y_percent = (y_predprob*100)
y_ted = np.array(list(range(testspeaker)))
```

```
listnew = []
y_new = list(y_percent)
nn = 0
fuc = np.append(arr = y_ted,values= y_pred)
fuclist = list(fuc)
for ff in fuclist:

    nn = ff
    for jj in y_new[nn]:
        if jj == max(y_new[nn]):

            maxi = jj
            listnew.append(maxi)
            print('Likelihood is '+str(maxi))
            if maxi>50:
                print(str(ff)+' of test set identifies itself with '+str(fuclist[ff+testspeaker])+' of train set')
                sumok += 1
                if str(ff) == str(fuclist[ff+testspeaker]):
                    closedsetcorrectness+=1

                #print('probably correct')
            else:
                print('No match')
                sumnotok +=1


    if ff == (testspeaker-1):

        print('done')
        break
print(str(closedsetcorrectness)+' correctly predicted and '+str(sumok-closedsetcorrectness)+' wrongly predicted as 
FA')
dd = np.array(listnew)
med = np.median(dd)

"""
# Applying LDA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components = 2)
X_lda = lda.fit_transform(X_train, y)
X_testlda = lda.transform(X_test)
"""
##############################################################################
print("Computing Linear Discriminant Analysis projection")
X2 = X_train.copy()
X2.flat[::X_train.shape[1] + 1] += 0.01  # Make X invertible

X_lda = discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_transform(X2, y)

#######################################################################################
N = 10
cmap = plt.cm.jet
# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]
# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

bounds = np.linspace(0,N,N+1)
norm = mpl.colors.BoundaryNorm(bounds, cmap.N)
```

```
print("Computing LDA projection")


scat = plt.scatter(X_lda[:, 0], X_lda[:, 1],c=y,s=np.random.randint(100,200,N),cmap=cmap,norm=norm)
# create the colorbar
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.title('LDA components  when trained with CC')
cb = plt.colorbar(scat, spacing='proportional',ticks=bounds)

plt.show()
outer1w = []
for iw in range(0,10):
   for jw in range(0,1):
       qw = iw
       outer1w.append(qw)
y1 = np.array(outer1w)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y1,y_pred)
```

<u>Code for implementing MFCC</u>

<u>base.py</u>

```
# calculate filterbank features. Provides e.g. fbank and mfcc features for use in ASR applications
# Author: James Lyons 2012
from __future__ import division
import numpy
from python_speech_features import sigproc
from scipy.fftpack import dct
#from scipy.signal import hamming

def mfcc(signal,samplerate=16000,winlen=0.025,winstep=0.01,numcep=20,
        nfilt=20,nfft=512,lowfreq=0,highfreq=None,preemph=0.97,ceplifter=22,appendEnergy=True,
        winfunc= numpy.hamming):
    """Compute MFCC features from an audio signal.

    :param signal: the audio signal from which to compute features. Should be an N*1 array
    :param samplerate: the samplerate of the signal we are working with.
    :param winlen: the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
    :param winstep: the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
    :param numcep: the number of cepstrum to return, default 13
    :param nfilt: the number of filters in the filterbank, default 26.
    :param nfft: the FFT size. Default is 512.
    :param lowfreq: lowest band edge of mel filters. In Hz, default is 0.
    :param highfreq: highest band edge of mel filters. In Hz, default is samplerate/2
    :param preemph: apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
    :param ceplifter: apply a lifter to final cepstral coefficients. 0 is no lifter. Default is 22.
    :param appendEnergy: if this is true, the zeroth cepstral coefficient is replaced with the log of the total frame
energy.
    :param winfunc: the analysis window to apply to each frame. By default no window is applied. You can use
numpy window functions here e.g. winfunc=numpy.hamming
    :returns: A numpy array of size (NUMFRAMES by numcep) containing features. Each row holds 1 feature
vector.
    """
    feat,energy = fbank(signal,samplerate,winlen,winstep,nfilt,nfft,lowfreq,highfreq,preemph,winfunc)
    feat = numpy.log(feat)
    feat = dct(feat, type=2, axis=1, norm='ortho')[:,:numcep]
    feat = lifter(feat,ceplifter)
    if appendEnergy: feat[:,0] = numpy.log(energy) # replace first cepstral coefficient with log of frame energy
    return feat

def fbank(signal,samplerate=16000,winlen=0.025,winstep=0.01,
          nfilt=20,nfft=512,lowfreq=0,highfreq=None,preemph=0.97,
          winfunc= numpy.hamming):
    """Compute Mel-filterbank energy features from an audio signal.

    :param signal: the audio signal from which to compute features. Should be an N*1 array
    :param samplerate: the samplerate of the signal we are working with.
    :param winlen: the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
    :param winstep: the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
    :param nfilt: the number of filters in the filterbank, default 26.
    :param nfft: the FFT size. Default is 512.
    :param lowfreq: lowest band edge of mel filters. In Hz, default is 0.
    :param highfreq: highest band edge of mel filters. In Hz, default is samplerate/2
    :param preemph: apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
    :param winfunc: the analysis window to apply to each frame. By default no window is applied. You can use
numpy window functions here e.g. winfunc=numpy.hamming
    :returns: 2 values. The first is a numpy array of size (NUMFRAMES by nfilt) containing features. Each row
holds 1 feature vector. The
        second return value is the energy in each frame (total energy, unwindowed)
    """
    highfreq= highfreq or samplerate/2
    signal = sigproc.preemphasis(signal,preemph)
    frames = sigproc.framesig(signal, winlen*samplerate, winstep*samplerate, winfunc)
    pspec = sigproc.powspec(frames,nfft)
    energy = numpy.sum(pspec,1) # this stores the total energy in each frame
    energy = numpy.where(energy == 0,numpy.finfo(float).eps,energy) # if energy is zero, we get problems with
log

    fb = get_filterbanks(nfilt,nfft,samplerate,lowfreq,highfreq)
    feat = numpy.dot(pspec,fb.T) # compute the filterbank energies
    feat = numpy.where(feat == 0,numpy.finfo(float).eps,feat) # if feat is zero, we get problems with log

    return feat,energy

def logfbank(signal,samplerate=16000,winlen=0.025,winstep=0.01,
          nfilt=20,nfft=512,lowfreq=0,highfreq=None,preemph=0.97):
    """Compute log Mel-filterbank energy features from an audio signal.

    :param signal: the audio signal from which to compute features. Should be an N*1 array
    :param samplerate: the samplerate of the signal we are working with.
    :param winlen: the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
    :param winstep: the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
    :param nfilt: the number of filters in the filterbank, default 26.
    :param nfft: the FFT size. Default is 512.
    :param lowfreq: lowest band edge of mel filters. In Hz, default is 0.
    :param highfreq: highest band edge of mel filters. In Hz, default is samplerate/2
    :param preemph: apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
    :returns: A numpy array of size (NUMFRAMES by nfilt) containing features. Each row holds 1 feature vector.
    """
    feat,energy = fbank(signal,samplerate,winlen,winstep,nfilt,nfft,lowfreq,highfreq,preemph)
    return numpy.log(feat)

def ssc(signal,samplerate=16000,winlen=0.025,winstep=0.01,
        nfilt=20,nfft=512,lowfreq=0,highfreq=None,preemph=0.97,
        winfunc= numpy.hamming):
```

```
    """Compute Spectral Subband Centroid features from an audio signal.

    :param signal: the audio signal from which to compute features. Should be an N*1 array
    :param samplerate: the samplerate of the signal we are working with.
    :param winlen: the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
    :param winstep: the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
    :param nfilt: the number of filters in the filterbank, default 26.
    :param nfft: the FFT size. Default is 512.
    :param lowfreq: lowest band edge of mel filters. In Hz, default is 0.
    :param highfreq: highest band edge of mel filters. In Hz, default is samplerate/2
    :param preemph: apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
    :param winfunc: the analysis window to apply to each frame. By default no window is applied. You can use
numpy window functions here e.g. winfunc=numpy.hamming
    :returns: A numpy array of size (NUMFRAMES by nfilt) containing features. Each row holds 1 feature vector.
    """
    highfreq= highfreq or samplerate/2
    signal = sigproc.preemphasis(signal,preemph)
    frames = sigproc.framesig(signal, winlen*samplerate, winstep*samplerate, winfunc)
    pspec = sigproc.powspec(frames,nfft)
    pspec = numpy.where(pspec == 0,numpy.finfo(float).eps,pspec) # if things are all zeros we get problems

    fb = get_filterbanks(nfilt,nfft,samplerate,lowfreq,highfreq)
    feat = numpy.dot(pspec,fb.T) # compute the filterbank energies
    R = numpy.tile(numpy.linspace(1,samplerate/2,numpy.size(pspec,1)),(numpy.size(pspec,0),1))

    return numpy.dot(pspec*R,fb.T) / feat

def hz2mel(hz):
    """Convert a value in Hertz to Mels

    :param hz: a value in Hz. This can also be a numpy array, conversion proceeds element-wise.
    :returns: a value in Mels. If an array was passed in, an identical sized array is returned.
    """
    return 2595 * numpy.log10(1+hz/700.)

def mel2hz(mel):
    """Convert a value in Mels to Hertz

    :param mel: a value in Mels. This can also be a numpy array, conversion proceeds element-wise.
    :returns: a value in Hertz. If an array was passed in, an identical sized array is returned.
    """
    return 700*(10**(mel/2595.0)-1)

def get_filterbanks(nfilt=20,nfft=512,samplerate=16000,lowfreq=0,highfreq=None):
    """Compute a Mel-filterbank. The filters are stored in the rows, the columns correspond
    to fft bins. The filters are returned as an array of size nfilt * (nfft/2 + 1)

    :param nfilt: the number of filters in the filterbank, default 20.
    :param nfft: the FFT size. Default is 512.
    :param samplerate: the samplerate of the signal we are working with. Affects mel spacing.
    :param lowfreq: lowest band edge of mel filters, default 0 Hz
    :param highfreq: highest band edge of mel filters, default samplerate/2
    :returns: A numpy array of size nfilt * (nfft/2 + 1) containing filterbank. Each row holds 1 filter.
    """
    highfreq= highfreq or samplerate/2
    assert highfreq <= samplerate/2, "highfreq is greater than samplerate/2"

    # compute points evenly spaced in mels
    lowmel = hz2mel(lowfreq)
    highmel = hz2mel(highfreq)
    melpoints = numpy.linspace(lowmel,highmel,nfilt+2)
    # our points are in Hz, but we use fft bins, so we have to convert
    #  from Hz to fft bin number
    bin = numpy.floor((nfft+1)*mel2hz(melpoints)/samplerate)

    fbank = numpy.zeros([nfilt,nfft//2+1])
    for j in range(0,nfilt):
        for i in range(int(bin[j]), int(bin[j+1])):
            fbank[j,i] = (i - bin[j]) / (bin[j+1]-bin[j])
        for i in range(int(bin[j+1]), int(bin[j+2])):
            fbank[j,i] = (bin[j+2]-i) / (bin[j+2]-bin[j+1])
    return fbank

def lifter(cepstra, L=22):
    """Apply a cepstral lifter the the matrix of cepstra. This has the effect of increasing the
    magnitude of the high frequency DCT coeffs.

    :param cepstra: the matrix of mel-cepstra, will be numframes * numcep in size.
    :param L: the liftering coefficient to use. Default is 22. L <= 0 disables lifter.
    """
    if L > 0:
        nframes,ncoeff = numpy.shape(cepstra)
        n = numpy.arange(ncoeff)
        lift = 1 + (L/2.)*numpy.sin(numpy.pi*n/L)
        return lift*cepstra
    else:
        # values of L <= 0, do nothing
        return cepstra

def delta(feat, N):
    """Compute delta features from a feature vector sequence.

    :param feat: A numpy array of size (NUMFRAMES by number of features) containing features. Each row holds
1 feature vector.
    :param N: For each frame, calculate delta features based on preceding and following N frames
    :returns: A numpy array of size (NUMFRAMES by number of features) containing delta features. Each row
holds 1 delta feature vector.
    """
    if N < 1:
        raise ValueError('N must be an integer >= 1')
    NUMFRAMES = len(feat)
    denominator = 2 * sum([i**2 for i in range(1, N+1)])
    delta_feat = numpy.empty_like(feat)
    padded = numpy.pad(feat, ((N, N), (0, 0)), mode='edge')   # padded version of feat
    for t in range(NUMFRAMES):
        delta_feat[t] = numpy.dot(numpy.arange(-N, N+1), padded[t : t+2*N+1]) / denominator   # [t : t+2*N+1] ==
[(N+t)-N : (N+t)+N+1]
    return delta_feat
```

Code for implementing MFCC in train set.

train2

```
from __future__ import division
import numpy as np
from scipy.io.wavfile import read
from LBG import lbg
#from mel_coefficients import mfcc

import matplotlib.pyplot as plt
import os
from python_speech_features import mfcc

def training(nfiltbank,dirt):
    coef_list = []
    nSpeaker = 100
    nCentroid = 64
    codebooks_mfcc = np.empty((nSpeaker,nfiltbank,nCentroid))
    #codebooks_lpc = np.empty((nSpeaker, orderLPC, nCentroid))
    directory = os.getcwd() + dirt;
    fname = str()

    for i in range(nSpeaker):
        fname = '/s' + str(i+1) + '.wav'
        print('Now speaker ', str(i+1), 'features are being trained' )
        (fs,s) = read(directory + fname)
        #mel_coeff = mfcc(s, fs, nfiltbank)
        #print(mel_coeff)

        mel_coefs = np.transpose(mfcc(s,fs))
        coef_list.append(mel_coefs)
        #lpc_coeff = lpc(s, fs, orderLPC)
        codebooks_mfcc[i,:,:] = lbg(mel_coefs, nCentroid)

    #plt.figure(nSpeaker + 1)
    #print(coef_list)

    #c1 = plt.scatter(coef_list[0,6,:],coef_list[0,4,:],s = 100, color = 'red', marker = '+')
    #c2 = plt.scatter(codebooks_mfcc[1,6,:], codebooks_mfcc[1,4,:], s = 500, color = 'blue',cmap='viridis', marker
= 'o')
    #c3 = plt.scatter(codebooks_mfcc[2,6,:], codebooks_mfcc[2,4,:], s = 100, color = 'grey', marker = '+')
    #c4 = plt.scatter(codebooks_mfcc[3,6,:], codebooks_mfcc[3,4,:], s = 100, color = 'yellow', marker = '+')
    #c5 = plt.scatter(codebooks_mfcc[4,6,:], codebooks_mfcc[4,4,:],s = 100, color = 'violet', marker = '+')
    #c6 = plt.scatter(codebooks_mfcc[5,6,:], codebooks_mfcc[5,4,:], s = 100, color = 'pink', marker = '+')
    #c7 = plt.scatter(codebooks_mfcc[6,6,:], codebooks_mfcc[6,4,:], s = 100, color = 'orange', marker = '+')
    #c8 = plt.scatter(codebooks_mfcc[7,6,:], codebooks_mfcc[7,4,:], s = 100, color = 'grey', marker = '+')
    #plt.grid()
    #plt.legend((c1, c2,c3,c4), ('Sp1 centroids', 'Sp2 centroids','Sp3 centroids', 'Sp4 centroids'), scatterpoints = 1,
loc = 'upper left')
    #plt.show()
    return (codebooks_mfcc)
```

Code for implementing Random forest classifier on OSTI-SI along with calculation of closed set and open set mean

opensetwithadaptivethres.py

```
"""
Created on Mon Apr 23 17:44:19 2018

@author: Shrutisarika
"""

from __future__ import division
```

```
import numpy as np
from scipy.io.wavfile import read
from LBG import EUDistance,lbg
#from mel_coefficients import mfcc
import matplotlib.pyplot as plt
from train2 import training
import os
from python_speech_features import mfcc
import pandas as pd
nSpeaker = 100

nfiltbank = 20
def mfcc_generator(nfiltbank,dirt):
    coef_list = []
    nSpeaker = 100
    nCentroid = 64
    mfcc_gen = np.empty((nSpeaker,nfiltbank,nCentroid))
    #codebooks_lpc = np.empty((nSpeaker, orderLPC, nCentroid))
    directory = os.getcwd() + dirt;
    fname = str()

    for i in range(nSpeaker):
        fname = '/s' + str(i+1) + '.wav'
        print('Now speaker ', str(i+1), 'features are being trained' )
        (fs,s) = read(directory + fname)
        #mel_coeff = mfcc(s, fs, nfiltbank)
        #print(mel_coeff)

        mel_coefs = np.transpose(mfcc(s,fs))
        coef_list.append(mel_coefs)
        #lpc_coeff = lpc(s, fs, orderLPC)
        mfcc_gen[i,:,:] = lbg(mel_coefs, nCentroid)
    return (mfcc_gen)
def mfcc_generatortrain(nfiltbank,dirt):
    coef_list = []
    nSpeaker = 100
    nCentroid = 64
    mfcc_gentrain = np.empty((nSpeaker,nfiltbank,nCentroid))
    #codebooks_lpc = np.empty((nSpeaker, orderLPC, nCentroid))
    directory = os.getcwd() + dirt;
    fname = str()

    for i in range(nSpeaker):
        fname = '/s' + str(i+1) + '.wav'
        print('Now speaker ', str(i+1), 'features are being trained' )
        (fs,s) = read(directory + fname)
        #mel_coeff = mfcc(s, fs, nfiltbank)
        #print(mel_coeff)

        mel_coefs = np.transpose(mfcc(s,fs))
        coef_list.append(mel_coefs)
        #lpc_coeff = lpc(s, fs, orderLPC)
        mfcc_gentrain[i,:,:] = lbg(mel_coefs, nCentroid)
    return (mfcc_gentrain)

A = 0
list1 = []
list11 = []
list2 = []
(codebooks_mfcc) = training(nfiltbank,'/trainrev')
```

```
for i in (codebooks_mfcc):
    for j in i:
        A = np.asarray(i).reshape(-1)
    list1.append(A)

k = np.array(list1)
#X = np.append(values = k,arr = np.zeros((4,1280)).astype(int),axis = 0)

(codebooks_test) = mfcc_generator(nfiltbank,'/testrev')
for i1 in (codebooks_test):
    for j1 in i1:
        A1 = np.asarray(i1).reshape(-1)
    list11.append(A1)

k1 = np.array(list11)
listd = []
(codebooks_train) = mfcc_generatortrain(nfiltbank,'/testrev')
for i11 in (codebooks_train):
    for j11 in i11:
        A11 = np.asarray(i11).reshape(-1)
    listd.append(A11)

ktrain = np.array(listd)


trainspeaker = 100
testspeaker = 100


X_train = np.array(k)

y = np.array(list(range(nSpeaker)))
y_ted = np.array(list(range(testspeaker)))
#y = np.append(arr = y,values=np.zeros(25))
X_test = np.array(k1)
ktrainarr = np.array(ktrain)
# Feature Scaling

from sklearn import metrics
#CREATE YOUR CLASSIFIER HERE
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10000, criterion = 'entropy', random_state = 0)

classifier.fit(X_train,y)

y_pred = classifier.predict(X_test)
#accurary = metrics.accuracy_score(y,y_pred)

y_predprob = classifier.predict_proba(X_test)
######################################################################
classifiertrain = RandomForestClassifier(n_estimators = 10000, criterion = 'entropy', random_state = 0)

classifiertrain.fit(X_train,y)

y_predtrain = classifier.predict(ktrainarr)
#accurary = metrics.accuracy_score(y,y_pred)

y_predprobtrain = classifier.predict_proba(ktrainarr)
###############################################################################
```

```
from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y,y_pred)
sumok = 0
closedsetcorrectness = 0
sumnotok = 0
y_percent = (y_predprob*100)
y_ptrain = (y_predprobtrain*100)
"""
for ii in y_percent:
    for jj in ii:
        if jj == max(ii):
            maxi = jj
            if maxi>15:
                sumok+=1
                print('may match')

            else:
                sumnotok+=1
                print('no match')


y_new = []
"""
listnew = []
y_new = list(y_percent)
nn = 0
fuc = np.append(arr = y_ted,values= y_pred)
fuclist = list(fuc)
for ff in fuclist:

    nn = ff
    for jj in y_new[nn]:
        if jj == max(y_new[nn]):

            maxi = jj
            listnew.append(maxi)
            print('Likelihood is '+str(maxi))
            if maxi>0:
                print('Speaker '+str(ff)+' of test set identifies itself with '+str(fuclist[ff+testspeaker])+' of train set')
                sumok += 1
                if str(ff) == str(fuclist[ff+testspeaker]):
                    closedsetcorrectness+=1

                #print('probably correct')
            else:
                print('Speaker '+str(ff)+'is unknown')
                sumnotok +=1


    if ff == (testspeaker-1):

        print('done')
        break
print(str(closedsetcorrectness)+' correctly predicted and '+str(sumok-closedsetcorrectness)+' wrongly predicted as
FA')
dd = np.array(listnew)
meanwithtest = np.mean(dd)

listnewtrain = []
y_newtrain = list(y_ptrain)
```

```
nn1 = 0
fuc1 = np.array(y_predtrain)
fuclist1 = list(fuc1)
for ff1 in fuclist1:

    nn1 = ff1
    for jj1 in y_newtrain[nn1]:
        if jj1 == max(y_newtrain[nn1]):

            maxim = jj1
            listnewtrain.append(maxim)
dd1 = np.array(listnewtrain)
meanwithtrain = np.mean(dd1)

a = trainspeaker
b = testspeaker
c = meanwithtrain
d = meanwithtest
listee = [a,b,c,d]
arraa = np.array(listee)
""
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y,y_pred)


from sklearn import metrics
accurary = metrics.accuracy_score(y,y_pred)

plt.plot(y_predprob)
plt.xlabel('class index')
plt.ylabel('Probability of Membership')
plt.show()
plt.savefig('probamale')
plt.plot(y,y_pred)
plt.scatter(y,y_pred)
plt.xlabel('actual class')
plt.ylabel('predicted class')
plt.show()
plt.savefig('maleacc')
“””
```

Code for implementing adaptive threshold .Through this code the threshold is predicted.

adaptivthreshold.py

```
"""
Created on Mon Apr 23 17:28:32 2018

@author: Shrutisarika
""
from matplotlib import offsetbox
import matplotlib as mpl
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#from opensetwithadaptivethres import arraa
# Importing the dataset
dataset = pd.read_csv('uyghurfemale.csv')
X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, 4].values
# Encoding categorical data
"""
# Encoding the Independent Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 3] = labelencoder_X.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()
#Avoiding the dummy variable trap
X = X[:,1:]
"""

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 0)

#will be creating random forest regressor later here
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10000,random_state = 0)
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)

#visualizing polynomial regression and applying some effects ie lowering the inc steps for better prediction

#visualizing polynomial regression results

a = (arraa)[np.newaxis]
print (a)

y_predtest = regressor.predict(a)

#plotting the predicted values train set
#from mpl_toolkits.mplot3d import Axes3D
xs1 = X_train[:,0]
xs2 = X_train[:,2]
xs3 = X_train[:,3]
plt.scatter(xs1,y_train,color = 'red')
plt.scatter(xs2,y_train,color = 'green')
plt.scatter(xs3,y_train,color = 'yellow')



#plotting the ind and dep variable,scattered
plt.plot(X_train,regressor.predict(X_train),color = 'blue')#plotting x_train vs predicted y_train values by regressor
#plt.plot(X_test,regressor.predict(X_test),color = 'green')
plt.title('Salary vs Experience(Training Set)')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.show()

"""
#plotting the predicted values test set

plt.scatter(X_test,y_test,color = 'orange')#plotting test values(dep and ind)
plt.plot(X_train,regressor.predict(X_train),color = 'green')#the regreesion model is the same like that of train set
plt.title('Salary vs Experience(Test Set)')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.show()
```

```
"""
from sklearn import (manifold, datasets, decomposition, ensemble,
                     discriminant_analysis, random_projection)
plt.plot(y_test,y_pred)
print("Computing Linear Discriminant Analysis projection")
X2 = X_train.copy()
X2.flat[::X_train.shape[1] + 1] += 0.01  # Make X invertible

X_lda = discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_transform(X2, y_train)

##########################################################################################
N = 2
cmap = plt.cm.jet
# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]
# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

bounds = np.linspace(0,N,N+1)
norm = mpl.colors.BoundaryNorm(bounds, cmap.N)

print("Computing LDA projection")


scat = plt.scatter(X_lda[:, 0], X_lda[:, 1],c=y_train,s=np.random.randint(50,100,N),cmap=cmap,norm=norm)
# create the colorbar
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.title('Plot of LDA components of musical instruments when trained with cepstral coeffs')
cb = plt.colorbar(scat, spacing='proportional',ticks=bounds)

plt.show()
```

PUBLICATIONS

1. S. Chakraborty and R. Parekh, "An improved approach to open set text-independent speaker identification (OSTI-SI)," *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, Kolkata, 2017, pp. 51-56.

An Improved Approach to Open Set Text-Independent Speaker Identification (OSTI-SI)

ShrutiSarika Chakraborty
School of Education Technology
Jadavpur University
Kolkata, India
shrutisarikachakraborty@gmail.com

Ranjan Parekh
School of Education Technology
Jadavpur University
Kolkata, India
rparekh@school.jdvu.ac.in

***Abstract*—This paper focuses on open set text independent speaker identification which is one of the most challenging subclass of Speaker recognition. The initial stage is similar to closed set speaker identification, where the distortion for each test voice against all train voices are determined. The distortions after normalization is set as decision criteria which eases the process of thresholding. The threshold variation which is mostly independent of dataset but dependent on the size of train data set and its values are quite similar for three datasets. The identification rate with balanced False Acceptance Rate(FAR) and False Rejection Rate(FRR) is 73-86%.**

Keywords--- Open set text-independent speaker identification; GMM-UBM (Gaussian mixture model-Universal Background model); Threshold; Vector Quantization (VQ);

I. INTRODUCTION

Speaker Recognition is the process of recognizing individual based on his/her voice. It can be classified into two types, speaker verification and speaker identification. Speaker verification is the process of verifying a speaker's claimed identity based on his/her already registered voice whereas speaker identification involves identifying whether a speaker's voice matches or not with any member of several registered voices [2]. Speaker verification is therefore a one to one matching process whereas speaker identification typically involves performing one to many matches. Both of these can either be text-dependent or text-independent. In the former case, a fixed and pre-defined text string is provided to the speaker based on which the voice patterns are compared, while in the latter case the text is arbitrary and typically unknown [3]. Speaker identification can again be of two types: open-set and closed-set. In the closed-set case, it is assumed that the test voice pattern is already present in the database and simply needs to be identified. In the open-set case, it is not known from beforehand whether the test voice pattern is actually present in the database or not [1]. Open-set matching is therefore more challenging [4] as it not only involves a comparison technique but also requires appropriate thresholds to prevent false matching of new voices with existing voices. Typical applications of closed-set speaker verification include voice based authentication systems while open-set speaker identification is required in surveillance and criminal investigations e.g. ransom callers.

The focus of this paper is to improve on existing techniques of open-set text-independent speaker identification (OSTI-SI). There are two main challenges to deal with. First, since because of text independence voice patterns do not have any text references for comparisons, this makes each voice pattern quite arbitrary which makes their comparisons difficult. Even the same speaker speaking different words can sound different and present radically changed voice waveforms. Secondly, because of the open-set database, it is not known whether the test voice pattern is actually present or not and can result in false matches. This requires determining a threshold for decision making, which is not fixed and will change depending on the database. A OSTI-SI based system will therefore need to be robust enough for comparing arbitrary voice patterns with each other and also incorporate adaptive thresholds that change depending on the database. The paper is organized as follows: section 2 provides a literature survey of existing approaches, section 3 outlines the proposed methodology, section 4 provides experimentations and results for testing the proposed system, section 5 analyses the system vis-à-vis other contemporary approaches, section 6 brings up the final conclusions with future scopes.

II. LITERATURE SURVEY

The past two decades have witnessed researches focusing on speaker recognition and its development. However, at present speech recognition technology is more implemented than speaker recognition technology [11]. Speaker identification is accomplished by proper feature extraction, choosing suitable feature for the purpose followed by feature matching [3]. Feature extraction deals with converting the speech data into acoustic vectors to facilitate feature matching. MFCCs are widely utilized as features in speech processing tasks like language identification, speaker identification, emotion recognition for it serves the purpose quite efficiently. [2] Feature matching employs quantization techniques like Vector Quantization (VQ) and/or speaker modeling techniques like GMM to achieve desired classification of test data [11]. In recent years, studies indicate that the energy distribution of human voice signals follows a Gaussian Model which is why GMM is more dominant in the field of speaker identification. For the past twenty years, GMM-UBM is identified as one of the major approaches, in field of speaker identification.

978-1-5386-1931-5/17/$31.00 ©2017 IEEE

[16,17]. The feature matching procedure is accompanied with the calculation of distance between the test speaker's acoustic model with all registered speakers' acoustic models. The test speech is then mapped with that registered speaker with which it processes minimum distance [2, 7]. Recent researches focuses on creating a thresholding method effective for OSTI-SI. The proposed method was quite efficient to eliminate all untrained or unknown data but not very accurate to authenticate known data [5]. OSTI-SI has also been tried upon cohort model and UBM based approach. It has been observed that when UBM based approach when unified with cohort model in a projective framework, improves accuracy [12]. There are other alternative approaches for open set speaker identification such as GMM-UBM supported by score normalization and i-vector method. A comparative study on the effectiveness of two methods had been made [1]. The test conducted on a subset of NIST-SRE 2008 database containing 400 registered speakers and 200 out of set speakers. It had been observed that when the test was conducted on clean data 39.5% accuracy was obtained for GMM-UBM while 42.5% for GMM-UBM with TZ(test normalization, zero normalization) norm and 49.5% with i-vector method. The i-vector method [15] has relation to the GMM-UBM technique as a single i-vector is said to be the consolidated representation of an adapted GMM. The other subset of speaker recognition which is speaker verification has gained prominence recently. Speaker identification suffers low accuracy as the increase in population of registered speakers increases, the variability and the chance of false acceptance of unknown speaker or false rejection of registered speaker increases [4]. It also becomes increasingly difficult to tune the threshold in such cases. Speaker verification is a simpler method involving binary classification of determining whether the speaker's claimed voice, matches with the already registered voice of that speaker in the database [15]. Actually, it is the task of deciding, if the given speech utterance is provided by the hypothesized speaker S or not [19]. The binary classifier can be formulated as follows, where T(x) is denoted as the test ratio (for speaker verification systems using GMM is the likelihood ratio) and η is the threshold value [19].

H0: x is from the hypothesized speaker.

H1: x is not from the hypothesized speaker.

Then the decision in an optimal manner is:

$$T(x) = \frac{f(H0|x)}{f(H1|x)} \geq \eta, accept, T(x) = \frac{f(H0|x)}{f(H1|x)} \leq \eta, reject$$

III. Proposed Methodology

It has been observed that OSTI-SI lacks implementation in day-to-day applications due to problems such as low accuracy and indefinite thresholding method, which is to be employed irrespective of dataset to obtain a reliable result. Also, it is quite difficult to define a threshold which will result in equal error rate. The other problem lies with the choice of decision to be used. Due to intra-person variability of speech more problems arises [4]. The more populated the registered dataset is, the more prone to error it is. This study focuses on building an improved method of OSTI-SI. The method consists of two stages. In the first stage, the MFCC features are extracted from the speech samples and then they are vector-quantized based on LBG (Linde-Buzo-Gray) algorithm. Then the Euclidean distance is measured for all test data against all trained samples. The train sample with which there is minimum Euclidean distance is considered as the best match for that particular test data. The Euclidean distances are normalized to obtain the data in range of 0-1. In the second stage, the normalized minimum Euclidean distance or the distortion of each test data with respect to its best matched train data is considered as the decision criteria. Then a threshold is determined based on observations which balances the FAR and FRR. The threshold tends to decrease with increase in population. The same method with the same threshold with only slight changes, for respective length of population has been compared against other datasets. The accuracy ranges between 73 – 86 % for all datasets. The block diagram of the process is given below in Fig.1.

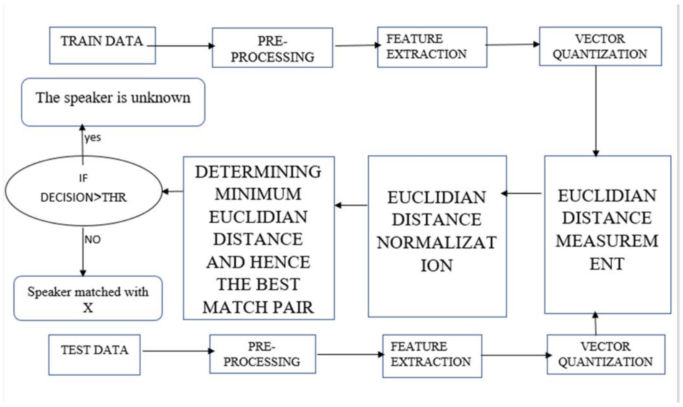

Fig.1 . Block diagram

A. *Pre-Processing.*

All voice samples are sampled at 16000Hz and the voices from different sessions of the same speaker have been merged to produce samples of length 60-150 sec. It is quite certain that lengthier the train data is, better is the accuracy. Also, clean noise free data are used for this paper.

B. *Feature Extraction*

For the purpose of speaker recognition, the most dominating features are mel frequency cepstral co-efficient [2]. MFCC involves sensitivities of human perception with respect to frequencies under consideration. The process of extracting MFCC features in given below in Fig.2.

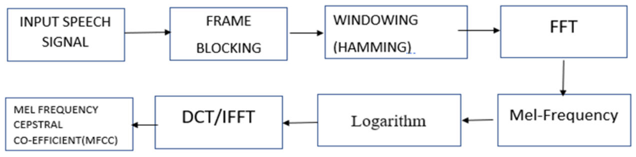

Fig.2 . MFCC

C. Framing

Entire speech sample is segmented into small frames before further operations. The length generally varies between 20 to 30 ms [7]. Adjacent frames are separated by M $(M < N)$ in a voice signal which is segmented into frames of N samples. Regular values for $N$and $M$are $N = 256, M = 100$.

D. Windowing

After frame blocking, each frame is windowed with a Hamming window in order to taper the first and last points of the frames to reduce signal discontinuities. In a typical case, the signal in a frame is denoted by (n), where $n = \{0,\ldots,N-1\}$, and the signal after windowing is given by $s(n) * t(n)$, where $t(n)$ is the representation of Hamming window defined by (1) [3]

$$t(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \;\; ; 0 \le n \le N-1 \qquad (1)$$

E. Discrete Fourier Transform

The Fast Fourier Transform (DFT) transforms each frame of Nsamples from the time domain to the frequency domain. The DFT operation is defined by the following:

$$X_k = \sum_{i=0}^{N-1} x_i . e^{-j.\frac{2\pi ki}{N}} \qquad (2)$$

F. Mel Frequency Wrapping

Human perception of the contents of frequency of sound for speech signals does not follow a linear scale. This fact has been proven from psychological studies. For each tone with an actual frequency f measured in Hz, a subjective pitch is measured on a scale called the 'mel' scale. This *mel-frequency* scale follows a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz [11] as given by the following formula:

$$M = 2595.\log_{10}\left(1 + \frac{f}{700}\right) \qquad (3)$$

The number of *mel* spectrum coefficients, *K*, is generally chosen to be 20. This filter bank is applied in the frequency domain, which is equivalent to applying the triangle-shape windows to the spectrum [3]. To smooth the magnitude spectrum and to minimize the size of the features are two major reasons of using triangular bandpass filters [3].

G. Discrete Cosine Transform

It is now required to transform the log Mel spectrum into time domain for which inverse Discrete Cosine Transform (DCT) is used [7]. The result of the transformation gives us Mel Frequency Cepstral Coefficient (MFCC) which represents acoustic vectors. So each input utterance is transformed into a sequence of the acoustic vector. Forward DCT is defined by the following where $\propto$ is a constant dependent on N

$$X_k = \propto . \sum_{i=0}^{N-1} x_i . \cos\left\{\frac{(2i+1)\pi k}{2N}\right\} \qquad (4)$$

H. Vector Quantization

Vector Quantization (VQ) is a data reduction method. It is useful for reduction of dimension so as to reduce redundancy of data. It may be regarded as a process by which vectors from a large vector space is consolidated into the limited number of regions present in that space [3]. The centre of each region so obtained (known as clusters) is called a codeword. A codebook is the collection of all codewords of the vectors. VQ technique is implemented through LBG algorithm. Feature vectors of both train and test data extracted from MFCC is applied to VQ [5].

I. Classification

Distortion measure is the parameter which represents the similarity between the input feature vectors and a codebook. The smaller the distortion, the higher the similarity. The distortion is shown below [5].

$$D_Q(I,V) = \sum_{t=1}^{T} \min d(i_t, v_k) \;\; ; \; 1 \le k \le K \qquad (5)$$

Here D_Q is the distortion of the input feature vectors I, which consist of T vectors, to the codebook V, which consist of k centroids, i_t is the tth input vector and $d$is the Euclidean distance. For two n-dimensional vectors $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$ Euclidean distance is defined as:

$$d(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \qquad (6)$$

The sum of the Euclidean distances between each feature vector to the nearest centroid in a codebook is the distortion lying between a codebook and a group of feature vectors [5]. The Euclidean distance is measured for each test sample against all train samples to obtain distortion. The Euclidean distances obtained for a certain test sample when compared against all train samples is normalized. Normalization sets all Euclidean distances within the range of 0-1. This simplifies the thresholding process. Let $a = \{a_1, a_2, \ldots, a_n\}$ denote the Euclidean distances of a certain test sample against n train data. Then Normalized Euclidean Distance is given by:

$$D = \frac{a_1 + a_2 + \cdots + a_n}{\sqrt{\sum_{i=1}^{n}(a_i)^2}} \; ; 1 \le i \le n \qquad (7)$$

Minimum Euclidean distance is the distortion which defines the similarity measure. The train data against which the test sample processes minimum Euclidean distance in a set of n train samples is the best match for that test sample. In closed set speaker identification, the test sample is matched against that train data. But in open set the scenario is different. The decision criteria is the minimum normalized Euclidian distance. If it is greater than the threshold the voice is rejected as unknown else it is matched with its best match.

$$\begin{aligned} D(X,S_i) > \sigma_i &\rightarrow reject \\ D(X,S_i) < \sigma_i &\rightarrow accept \end{aligned} \qquad (8)$$

Here S_i is the set of train data/registered speakers stored in the database and X is the current test sample, D denotes normalized Euclidian distances, σ_i are the threshold values chosen to obtain equal error rate. The error which arises in first stage of OSTI-SI is the error of mismatch. The cause of it can be many but not decision or threshold. So, this error is overlooked in this paper. The second error is False acceptance error(FA), where an unknown voice is mistakenly matched against a train data. The third error is False Rejection error(FR), where the known voice is mistakenly rejected as unknown. These errors rise either due to decision or threshold. In order to gain an equal error rate, where the number of false acceptance rate (FAR) and false rejection rate (FRR) error is balanced the threshold needs to be correct.

TABLE I. FAR AND FRR

		Predicted	
		Registered Speaker	Unknown voice
Actual	Registered speaker	No error	FR
	Unknown voice	FA	No error

FAR is directly proportional to the decision threshold σ_i , while FRR is inversely proportional to the same. FAR and FRR when plotted against the decision threshold the point of intersection of these two curves is defined as the Equal Error Rate (EER). The FAR and FRR are equal at EER [6]. It has been observed that the threshold values adopted by trial and error method to obtain EER is more or less same for all the datasets. The threshold is studied with gradual increase in length of train and test data and it decreases with increase in population which is quite obvious. The determination of threshold is simplified by normalization of Euclidian distances which otherwise would had been too difficult to study.

This approach is independent of external variations, which may affect accuracy negatively at times. In GMM-UBM,UBM is a model trained from voices of non-target speakers using expectation maximization (EM) may hamper system accuracy. But that is the inevitable part of the approach. Proper choice of UBM set is necessary to get optimum accuracy. The choice of UBM set is decided by trial and error method which is again time-consuming. Cohort model fails for speakers sounding similar to each other. This method, with proper threshold is free from such errors. The results obtained from Uyghur dataset, where the similarity of voices of different persons is very high, demonstrates that. It also balances FAR and FRR unlike methods that gives unbalanced FAR and FRR, the latter being very high, compared to former. It is required to make further studies on the thresholding method for this method, so that an adaptive threshold can be employed to overcome difficulties in thresholding.

IV. Experimentations and Results

Two datasets used to test the performance and efficacy of the proposed approach. The first is the Uyghur dataset. It contains the voice of 200 speakers equally divided into male and female class. It is a subset of THYUG-20 SRE [9]. It contains 3003 recorded voices of 200 people. The environment is noise free. For each speaker, his/her train data is formed by merging his/her voice from multiple trials producing a speech signal with the length varying between 60-150 sec. The test data is kept constant to 10 sec. The male and female datasets are further subdivided into two to conduct the experiment. The number of train data at each trial is kept exactly half of test data to test open set accuracy. The number of train data and test data is gradually increased as observed in Figures 3,4,5 to note the gradual change in threshold. The second dataset is the Librispeech dataset formed from Librispeech recordings [10]. All samples are clean data i.e. without noise. The train dataset contains 25 speech signals while the number of test data is 50. The train data are of 60-150 sec in length while test data is of 8-10 sec in length. After normalization, the distortions lie between 0-1. For simplicity threshold has been multiplied with 100 so as to ease the process of observation. In case of GMM-UBM, the UBM model used here is another set of Librispeech recordings totally different from the previous one. It also consists of a few custom recordings recorded in noise free environment. A total of 50 speech signals are used for training the UBM model and each has a duration of 30-60 sec.

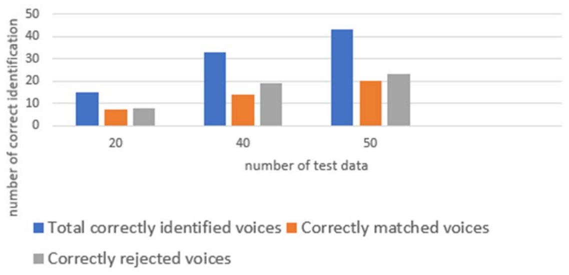

Fig. 3. Open set identification of Librispeech dataset

Fig 3 plots number of correctly identified voices of Librispeech recordings with 20, 40 and 50 test data respectively. Fig 4 and 5 plots open set identification of Uyghur male and female dataset with 40, 60, 80 and 100 test data respectively. Each voice is sampled to 16000Hz and all voices used here are clean data with no environmental noise. Figures below contain the result obtained by applying proposed method on the above-mentioned datasets where the number of voices at each trial increases gradually.

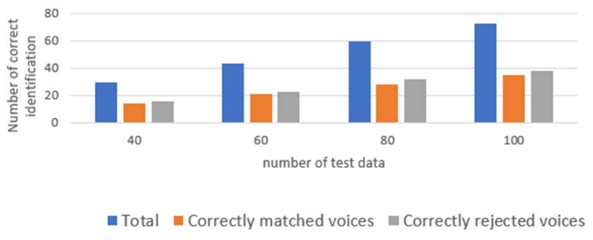

Fig. 4. Open set identification of Uyghur male dataset

In each case number of train data is exactly half of test data. The threshold value decreases with increase in number of train data as the denominator increases (equation 7) the normalized distortion falls and this explains the change in threshold value. Value of thresholds for different datasets do not differ much.

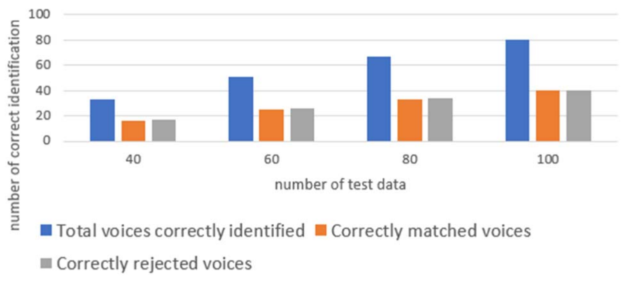

Fig. 5. Open set identification of Uyghur female dataset

It can be observed from the figures 3,4,5 that the identification rate of the system is in between (73-86)%. A balance is achieved between FAR and FRR. It can be seen from Fig 6 that the thresholds used in all three different datasets are more or less equal. It gradually decreases with increase in number of train data and change of threshold follows a similar pattern.

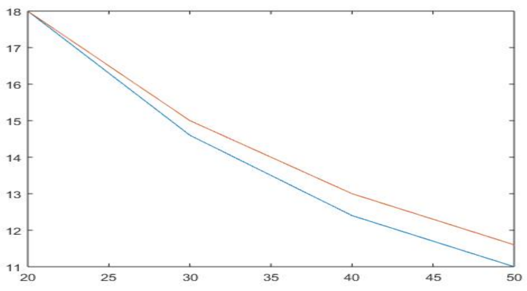

Fig. 6. Variation of threshold values with number of train data for Uyghur female and male dataset

For Uyghur, male dataset the closed set speaker identification is 93/100 test voices. The number of correct identification of unknown voices is 38/50 (76%), while the correct identification of known voices and matching it to its best match is 35/50 (70%). The overall accuracy is 73%. The increase in number of voices increases the variability and hence accuracy falls. For Uyghur, female dataset the closed set speaker identification is 91/100. The number of correct identification of unknown voices is 40/50 (80%) while the correct identification of known voices is 40/50 (80%). The overall open set identification is 80%. The threshold used is 11.6 while for male dataset it is 11.0. The threshold is not subject to huge changes in order to get enough accuracy. For Librispeech recordings the closed set speaker identification is 44 out of 50 voices. The overall correct identification is 43 (86%) while the correct identification of known voices is 20/25 (80%). The number of unknown voices correctly rejected is 23/25 (92%).It is more due to less number of data. In all these cases half of test samples are unknown and half of them are known. Hence the train set contains half the number of voice samples present in test data set in all cases.

V. ANALYSIS

In closed set identification, it is assumed that the test voice will belong to one of the registered voices and it is matched to its best match. There is no provision of decision or threshold to reject an imposter voice. In OSTI-SI, determination of correct decision and a good threshold is prioritized. Hence closed set identification technique needs further improvement for open set identification. Recent researches focusing on creating a thresholding method was efficient to nullify FA but suffered from a considerable amount of FR [5]. A threshold for each codebook was trained after the generation of the codebook of certain person or speaker. The threshold which consisted of minimum and maximum distortion value, was obtained by authenticating several voices of the same speaker [5]. In the verification phase, the resulting lowest distortion is validated [2, 7]. But it is desired to obtain a balance between FA and FR, so the need for improved method arises. Although cohort model performs well in situation where unknown voices belongs to casual imposters but cohort based speaker model becomes more vulnerable to attack by speakers sounding more similar to registered speakers [13]. The speaker verification is efficient for 1:1 match. This idea was extended into GMM-UBM where GMM is trained for each train samples in the training set. A speaker-independent model, or UBM, is trained from the out-of-set speakers using the EM algorithm [12]. Every speaker has a model which is represented with certain parameters using a GMM.A score that decides whether a given utterance $Q = \{q_1;\ q_2;\ \dots, q_n\}$ originates with speaker 'a' using the mean log-likelihood 1/n log(P(Q|θa)) where N is the number of models in the mixture, $\theta a = \{\mu a, \sum_{a}, \alpha_a\}$.The decision function for a UBM is given as follows, where T_θ is the threshold [12].

$$\frac{1}{n}\log[P(Q|\theta a)] - \frac{1}{n}\log[P(Q|\theta_{UBM})] > T_\theta \qquad (9)$$

The decision adopted for this paper is far simpler considered to GMM-UBM or i-vector approach. The method used for speaker classification is based on Bayes Classifier. Bayes decision rule is the optimal decision rule [18].

$$P(w|x) = \frac{P(x|w).P(w)}{P(x)} \qquad (10)$$

In words, posterior = (likelihood × prior) / evidence. To get maximum accuracy in classification, the posterior probability should be maximum [18] and so the distortion must be minimum. Hence, we are employing this criteria as the decision criteria unlike GMM-UBM where we take the likelihood ratio as the deciding factor. In this study, comparison of proposed method against GMM-UBM has been done, where GMM-UBM classifier is given by,

$$\log[P(X|\theta)] - \log[P(X|\theta_{UBM})] > T_\theta \qquad (11)$$

Also in case of GMM-UBM an additional set of Universal background model is needed, consisting of voices of unknown speakers to be compared against the registered data set. The change in UBM dataset also changes the accuracy for the test set. Hence for a particular dataset a particular UBM is needed for optimum accuracy. Fig.7 indicates comparison between GMM-UBM and the proposed method. The threshold value is inversely proportional to number of train data in the database.

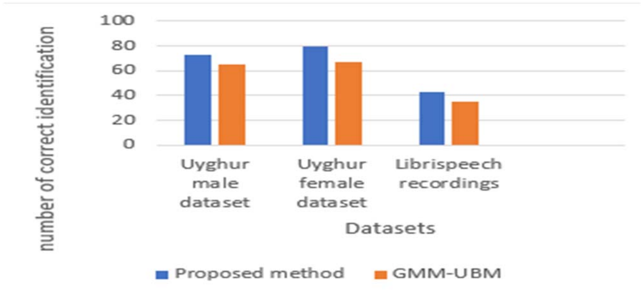

Fig. 7. Comparisons

Hence this study provides an improved approach for which (i) The accuracy of proposed method is more than GMM-UBM method which is one of the most dominating method [1] in the field of OSTI-SI. (ii) Unlike GMM-UBM it is independent of UBM model. For GMM-UBM,we had to choose an appropiate UBM for better accuracy (equation 11). (iii) The method provides a simpler approach to speaker identification, where the threshold can be determined from the rate of its decrease with increase in train data and vice versa. Due to normalization, the threshold value lies within 0-1 and it's value for different datasets are quite similar to each other.(iv)It works well for similar sounding,but different voices arising from different sources unlike cohort models.

VI. Conclusions and Future Scopes

OSTI-SI is the most challenging subset of Speaker identification. The paper proposes a method better and faster than the predominant GMM-UBM technique. It is not dependent on external factors like UBM as in case of GMM-UBM whose accuracy differs with different UBM .It produces good result with similar sounding but different voices, unlike cohort model. The thresholding process is also simpler due to normalization of the distortion, which is treated as decision. The threshold variation is inversely proportional to the size of train data-set.The FAR and FRR is also balanced. It is needed to create a adaptive threshold. Also, it is needed to apply revised method in order to increase the accuracy of the process.

References

[1] R. Karadaghi, H. Hertlein and A. Ariyaeeinia, "Effectiveness in open-set speaker identification," 2014 International Carnahan Conference on Security Technology (ICCST), Rome, 2014, pp. 1-6

[2] A. K. Singh, R. Singh and A. Dwivedi, "Mel frequency cepstral coefficients based text independent Automatic Speaker Recognition using matlab," International Conference on Reliability Optimization and Information Technology (ICROIT), Faridabad, 2014, pp. 524-527.

[3] N. M. AboElenein, K. M. Amin, M. Ibrahim and M. M. Hadhoud, "Improved text-independent speaker identification system for real time applications," Fourth Int. Japan-Egypt Conference on Electronics, Communications and Computers (JEC-ECC), Cairo, 2016, pp. 58-62

[4] A. M. Ariyaeeinia, J. Fortuna, P. Sivakumaran and A. Malegaonkar, "Verification effectiveness in open-set speaker identification," in IEE Proceedings - Vision, Image and Signal Processing, vol. 153, no. 5, pp. 618-624, Oct. 2006.

[5] R. A. Sadewa, T. A. B. Wirayuda and S. Sa'adah, "Speaker recognition implementation for authentication using filtered MFCC — VQ and a thresholding method," 3rd International Conference on Information and Communication Technology (ICoICT), Nusa Dua, 2015, pp. 261-265.

[6] H. B. Kekre and V. Kulkarni, "Closed set and open set Speaker Identification using amplitude distribution of different Transforms," 2013 International Conference on Advances in Technology and Engineering (ICATE), Mumbai, 2013, pp. 1-8.

[7] F. K. Soong, A. E. Rosenberg, B. H. Juang and L. R. Rabiner, "Report: A vector quantization approach to speaker recognition," in AT&T Technical Journal, vol. 66, no. 2, pp. 14-26, March-April 1987

[8] S. Dey and K. Kashyap, "A dynamic-threshold approach to text-dependent speaker recognition using principles of immune system," IEEE India Conf (INDICON), New Delhi, 2015, pp. 1-6.

[9] A. Rozi, Dong Wang, Zhiyong Zhang and T. F. Zheng, "An open/free database and Benchmark for Uyghur speaker recognition," Int Conf. Oriental COCOSDA held jointly with conf. on Asian Spoken Language Research and Evaluation, Shanghai, 2015, pp. 81-85.

[10] V. Panayotov, G. Chen, D. Povey and S. Khudanpur, "Librispeech: An ASR corpusbased on public domainaudiobooks," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, 2015, pp. 5206-5210.

[11] F. Y. Leu and G. L. Lin, "An MFCC-Based Speaker Identification System," IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), Taipei, 2017, pp. 1055-1062.

[12] A. Brew and P. Cunningham, "Combining Cohort and UBM Models in Open Set Speaker Identification," Seventh International Workshop on Content-Based Multimedia Indexing, Chania, 2009, pp. 62-67.

[13] V. Prakash and J. H. L. Hansen, "In-Set/Out-of-Set Speaker Recognition Under Sparse Enrollment," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 15, no. 7, pp. 2044-2052, Sept. 2007.

[14] I. Magrin-Chagnolleau, F. Bimbot, and R. IRISA. Indexing telephone conversations by speakers using time frequency principal component analysis.. IEEE Int. Conf. on Multimedia and Expo, 2000.

[15] N. Dehak, P. Kenny, R. Dehak et al., "Front-End Factor Analysis for Speaker Verification," IEEE Transactions on Audio, Speech, and Language Processing,,vol. 19, no. 4, pp. 788-798, 2011.

[16] D. A. Reynolds, R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," IEEE Transactions on,Speech and Audio Processing, vol. 3, no. 1, pp. 72-83, 1995.

[17] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models," Digital Signal Processing, vol. 10, no. 1–3, pp. 19-41, 2000.

[18] R. Duda, P. Hart,D. Stork. Pattern Classification,2nd edition, Wiley, New York

[19] F. Răstoceanu and M. Lazăr, "Score fusion methods for text-independent speaker verification applications," 6th Conference on Speech Technology and Human-Computer Dialogue (SpeD), Brasov, 2011, pp. 1-6.