Speaker Recognition

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Engineering

by

Shubham Ghosalkar

Rohan Jagiasi

Punit Kulal

under the guidance of

Mrs. Asha Bharambe

V.E.S.

Since 1962

Department of Information Technology

Vivekanand Education Society's Institute of Technology

2017-18

Abstract

Speaker Recognition is the ability of the system to recognise the speaker from the set of speaker samples available in the system. It is of 2 types, one which uses a keyword, called as text dependent systems, and another, which can recognizes voice in any language/text, also called as text independent speaker recognition. A text independent, language independent speaker recognition system is implemented using Dense & Convolutional Neural Networks.

Speaker Recognition has found several applications in upcoming electronic products like personal/home assistants, telephone banking and biometric identification. In this project, we explore a system which uses MFCC along with DNN and CNN as the model for building a speaker recognition system.

Keywords: speaker recognition, feature extraction, mel frequency cepstral coefficient, acoustic features, vector quantization, gaussian mixture model, deep neural networks

Table of Contents

Chapter 1

Introduction

1.1 Introduction

The speech signal contains many levels of information. Primarily a message is conveyed via the spoken words. At other levels, speech conveys the information about the language being spoken, the emotion, gender, and the identity of the speaker. The automatic recognition of speaker and speech recognition are very closely related. While speech recognition sets its goals at recognizing the spoken words in speech, the aim of automatic speaker recognition is to identity the speaker by extraction, characterization and recognition of the information contained in the speech signal. Automatic Speaker Recognition is a biometric which is used in telephone banking, home security etc. It is also used in smart devices in order to give a more personalized experience to the user.

1.2 Problem Statement

To build a system which takes voice samples as input and recognizes the identity of the speaker provided that the speaker's voice sample is already available to the system.

1.3 Objectives

- To build a model from the extracted features of voice samples.
- To recognize the identity of the speaker, given a voice sample having the same utterance as the one learnt by the system.

Chapter 2

Literature Survey

We studied multiple papers which built speaker recognition systems using different approaches like Hidden Markov models, Decision Trees, etc. Out of these multiple approaches, Vector Quantization, Gaussian Mixture Model and Neural Networks are known for their higher accuracy and efficient processing.

2.1 Literature/Techniques Studied

Analog audio signals can't be directly used to build a model for speaker recognition. A process called feature extraction(dimensionality reduction) is used to obtain a vector which can be further processed to build a model. There are various feature extraction methods such as MFCC, LPC, PLP etc. of which we have studied MFCC in detail.

- Feature Extraction using MFCC
    - Features are informative and nonredundant data, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.
    - Mel-frequency cepstral coefficients(MFCCs) are the state of the art technique in extracting features from a voice sample.
    - MFCCs use concepts such as mel scale, mel filterbank, fourier transform, discrete cosine transform, periodogram estimate for extracting the feature.

- Mel Scale

  The Mel scale was named and invented by Stevens Volkman and Newman. It is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The name mel comes from the word melody to indicate that the scale is based on pitch comparisons. It is a logarithmic scale similar to the way the human ear perceives sound.

$$f_{mel} = 2595 \cdot \log_{10}(1 + f/700)$$

where f is the linear frequency.

There are many approaches/techniques to build a model based on the features extracted from the audio signal. We have studied three such techniques which are mentioned below:

- Vector Quantization[3]
  - Vector Quantization is a technique in which a we build an n-dimensional vector from the features of the voice sample.
  - Thus, each voice sample is present in the n-dimensional vector space.
  - The process of recognition takes place by finding the closest point to the new input voice sample to be recognized from the n-dimensional space.
  - The speaker corresponding to the closest point is returned as the output.

- Gaussian Mixture Model[4]
  - Gaussian Mixture Model is a probabilistic model for representing normally distributed subpopulations within an overall population.
  - GMM makes use of an iterative algorithm called Expectation Maximization.
  - First we calculate the expectation of the component assignments for each data point given the model parameters.
  - Then these calculated expectations values are maximized with respect to the model parameters.
  - The speaker corresponding to the maximized value is returned as the output.

- Neural Network[5]
  - Neural nets take a feature vector as input. MFCC feature vector is used.
  - A multi layered network is used and the next node’s calculation depends on the weightage of each element in the vector.
  - Various activation functions like tanh, relu, etc
  - Softmax is used in the last layer of the neural net.

2.2 Papers

1. Lior Uzan and Lior Wolf in their paper [1] compared the I-Vector method, which is currently the leading method in voice recognition, to a novel deep neural network solution. Deep CNN is used to train spectrogram images obtained from raw voice samples. The experiments show that CNN outperforms the I-Vector based system in a significant margin. This could be due to the fact that I-Vectors are usually beneficial on systems with a larger number of speakers, or the relatively short length of utterances (2.29 seconds on average).
2. Roger Achkar, Mustafa El-Halabi, Elie Bassil, Rayan Fakhro and Marny Khalil in their paper [2] explained how neural networks can be used for speaker recognition. The neural network described in this paper is a Multilayer Perceptron (MLP). The MLP that is used in this system is made up of 6 input neurons, 1 hidden layer consisting of 4 neurons, and 2 output neurons. It uses Linear Prediction in order to extract formant frequencies which is used as the input vector for the net. The paper guarantees an accuracy for upto 6% for 4 speakers but the accuracy may decrease as more speakers are added. The paper focuses on a robust Speaker Recognition security system for handheld devices.
3. Geeta Nijhawan and Dr. M.K Soni in the paper [8] described their approach towards speaker recognition using MFCC for conversion of the voice sample into the feature vector. Preprocessing uses a method called as Spectral Subtraction. MFCC and LPC are used and the features are fed into 2 models, a radial basis neural net and a back propagation neural net. Around 80% of accuracy is obtained. It emphasizes on text independent speaker recognition.
4. Douglas A. Reynolds in his paper [4] presented and evaluated identification and verification systems for text-independent speaker recognition based on Gaussian mixture speaker models. The paper explained the general approach to use this probabilistic model. It achieves upto 96% accuracy for 46 speakers. The dataset used is telephone conversation database with 8kHz as sampling rate.

Chapter 3

Analysis

This chapter provides details about the structure of the project, software requirements, dataset and libraries used.

3.1 System analysis:

The project is divided into 3 main sections,

- Feature Extraction
  Voice samples need to be converted to numbers. Extracting features from voice samples is the first step in speaker recognition

- Training
  This stage involves feeding the extracted features into the neural net. This is where the speaker gets enrolled in the model.

- Testing
  In this stage, a recorded test sample goes through the feature extraction step. The features extracted are then tested on the trained net and the result is obtained.

3.2 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the users. The use case diagram for our system is given below.

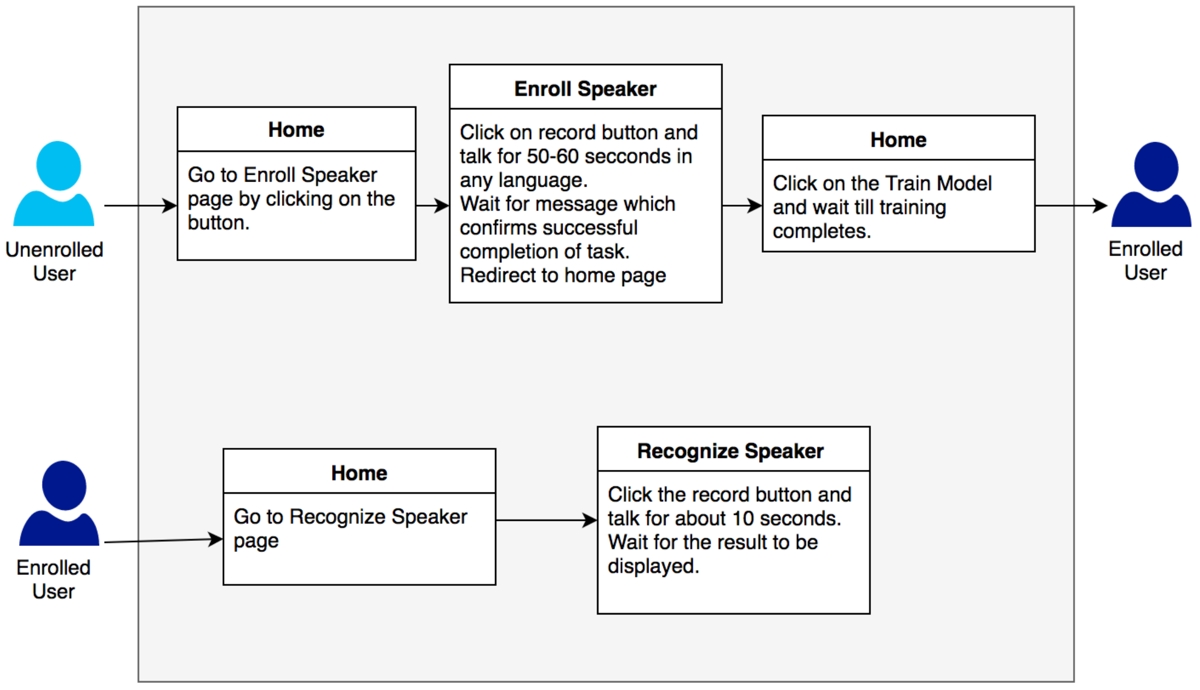

Unenrolled User
Home
Go to Enroll Speaker page by clicking on the button.
Enroll Speaker
Click on record button and talk for 50-60 seconds in any language.
Wait for message which confirms successful completion of task.
Redirect to home page
Home
Click on the Train Model and wait till training completes.
Enrolled User
Enrolled User
Home
Go to Recognize Speaker page
Recognize Speaker
Click the record button and talk for about 10 seconds.
Wait for the result to be displayed.

Figure 3.1 - Use Case Diagram

3.3 Software Requirements

Anaconda

Anaconda is a free and open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system conda.

It includes libraries like iPython notebook, Matplotlib, Numpy and other essential libraries used for machine learning.

Python 3 was used as the programming platform.

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

It allows easy and fast prototyping (through user friendliness, modularity, and extensibility). It supports both convolutional networks and recurrent networks, as well as combinations of the two. Keras can run seamlessly on CPU and GPU.

Tensorflow

Tensorflow was used as backend for Keras. TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open source license on November 9, 2015

Python Speech Features

Python Speech Features is an open source library hosted on Github which is used to extract features from voice samples. MFCC and Filterbank energies can be easily extracted from a wav file, ie, an uncompressed audio file using this library. The task of converting the voice signal to mel scale, by the steps of framing and windowing, taking log of filterbank energies, discrete cosine transform, is done by this library

SoX

SoX, which stands for Sound eXchange, is often called as the Swiss Army knife of audio manipulation. SoX is a command-line audio processing tool, particularly suited to making quick, simple edits and to batch processing. SoX reads and writes audio files in most popular formats and can optionally apply effects to them. It can combine multiple input sources, synthesise audio, and, on many systems, act as a general purpose audio player or a multi-track audio recorder. It also has limited ability to split the input into multiple output files. It has been used in order to pre-process raw audio samples recorded.

Audacity

Audacity is a free open source digital audio editor and recording computer software application. Audacity can be used for post-processing of all types of audio, including podcasts by adding effects such as normalization, trimming, and fading in and out. It is the GUI equivalent of SoX which is useful in order to visualize voice graphs and fine tune the settings as required.

Flask

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It is BSD licensed. Flask is called a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Flask is used in the project to make a Web UI and connect it to the Tensorflow Backend.

Dataset Description

THUGY20 is an open Uyghur speech database published by Center for Speech and Language Technology (CSLT) at Tsinghua University, Signal and Information Processing Lab at Xinjiang University, and the AI cloud research center (AICRC). It involves the full set of speech and language resources required to establish an Uyghur speech recognition system and an Uyghur speaker recognition system. Voices of upto 200 speakers, 100 men and 100 women, are available in the dataset. Train samples are over a minute and test samples are between 8-10 seconds in length.

Chapter 4

System Design

This chapter explains the design of the project, with the basic architecture, detailed flow design and the use case diagram.

4.1 Architecture

It shows the various components or modules in the project and how they interact with each other.

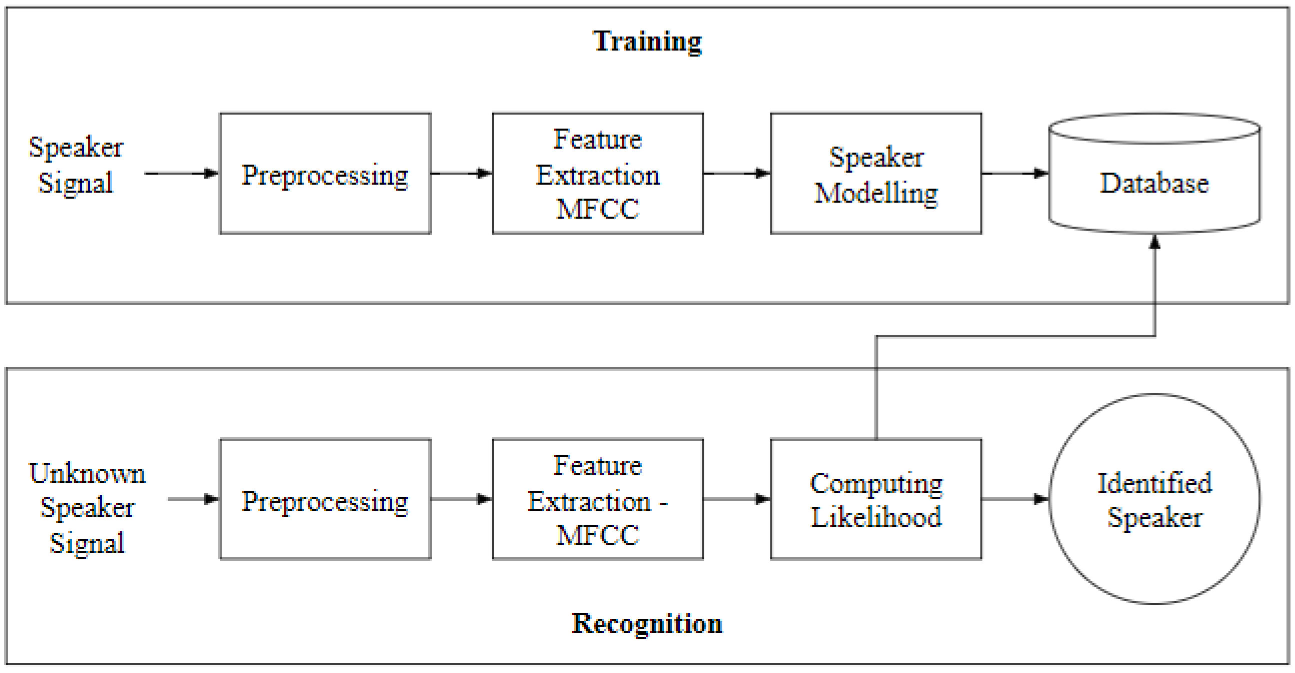


Figure 4.1

Figure 4.1 describes the architecture of the system. The components of the system are as follows:

- Preprocessing block: In this block, the voice samples are preprocessed to minimize/filter the noise content and eliminate the silent parts in the signal.
- Feature Extraction block: This block is responsible for extracting MFCCs. It then reduces the dimension of these vectors and passes it to the Speaker Modelling block.
- Speaker modelling block: This block take MFCCs as input and builds a model.
- Computing Likelihood: This block is present only in recognition phase. It searches for a match in the trained model to identify the speaker and returns it as output.

4.2 State Transition Diagram and Explanation

It shows the state of the raw audio file as it undergoes preprocessing and is recognised by the system.

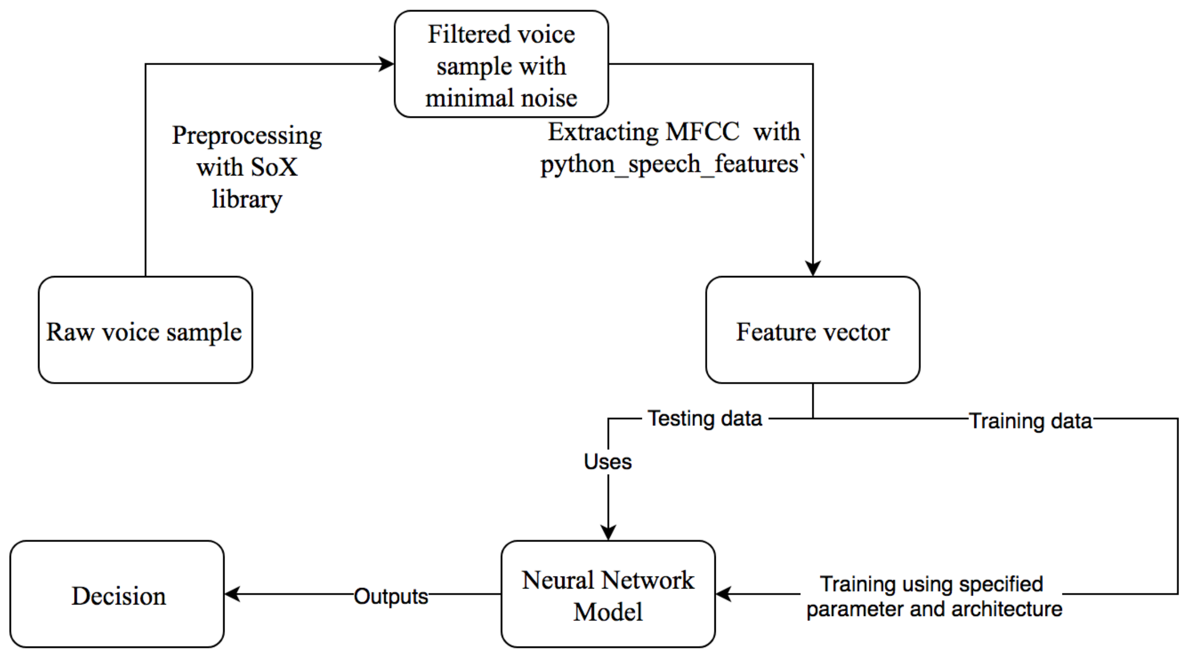

Filtered voice sample with minimal noise
Preprocessing with SoX library
Extracting MFCC with python_speech_features`
Raw voice sample
Feature vector
Testing data
Training data
Uses
Decision
Outputs
Neural Network Model
Training using specified parameter and architecture

Figure 4.2

Figure 4.2 shows the different states the algorithm passes before reaching its output. Initially, raw voice sample is taken as input from the microphone. The preprocessing is done using a command line tool *SoX(Sound Exchange)*. This gives us the sample with minimal noise. The feature vector MFCC is extracted using a python library, *python_speech_features.* The feature vector is then fed to the neural network. Based on the weights and functions used, the net identifies the voice and returns the name of the speaker.

The system starts by recording the user reading a paragraph from any article for a specific amount of time (1 minute in our case). The voice signal is then preprocessed using the SoX framework for eliminating noise and silence from the audio. MFCC (Mel Frequency Cepstral Coefficients) is then extracted from the processed voice signal using a library. These features are then fed to a neural network. After all the speakers have been enrolled, the model is tested.

Testing is fairly simple, the user speaks a couple of words which are recorded, pre-processed and the features extracted is tested on the CNN model built during the training phase of the system.

4.2.1 Preprocessing

We use Sound Exchange(SOX), an audio processing framework, to remove the noise and silent parts from the audio. The silence is removed by using a time threshold; anything below the threshold is removed.

The human voice has a fundamental frequency range of 85 to 180 Hz for male and 165 to 255 Hz for children and female. However, when a person speaks, their frequency isn't fixed. It varies for different words. The energy also spreads to nearby frequencies which gives a diminishing effect to sound, necessary to utter certain specific words in certain languages. However, this spread needs to be limited in order to distinguish between voice and noise.

The application uses a sampling rate of 44.1Khz (subject to availability) for optimum quality audio file for better results. It has been noticed that majority of the information is stored in the first 0-8000 Hz bandwidth. A low pass filter is, therefore, applied to remove higher frequency sounds which is mostly ambient noise. The obtained audio signal is saved in the application and will be referred as processed audio from here on.

As given below, Figure 4.3 shows the original raw voice sample. Figure 4.4 shows the sample processed using a application called Audacity, it completely eliminates noise. Since it had to be scripted, the parameters of SoX were fine tuned to get a similar result. (Figure 4.5)

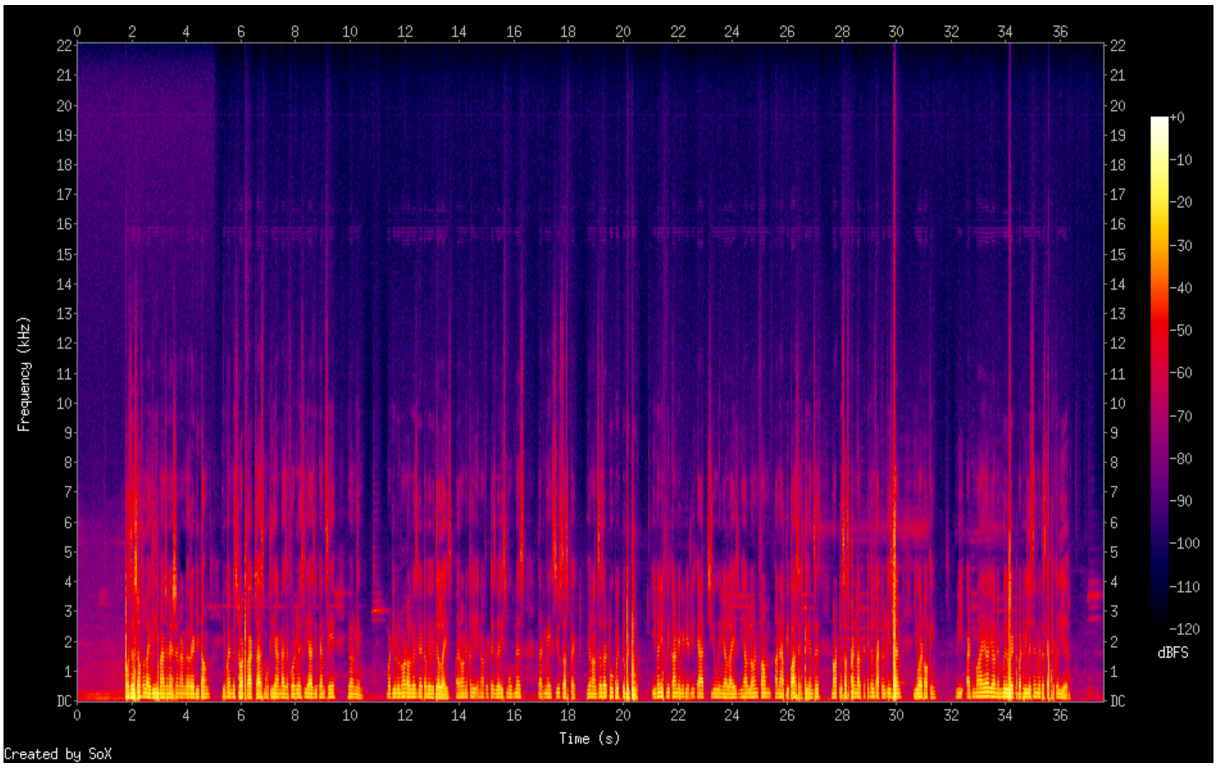


Figure 4.3 Original Voice Sample

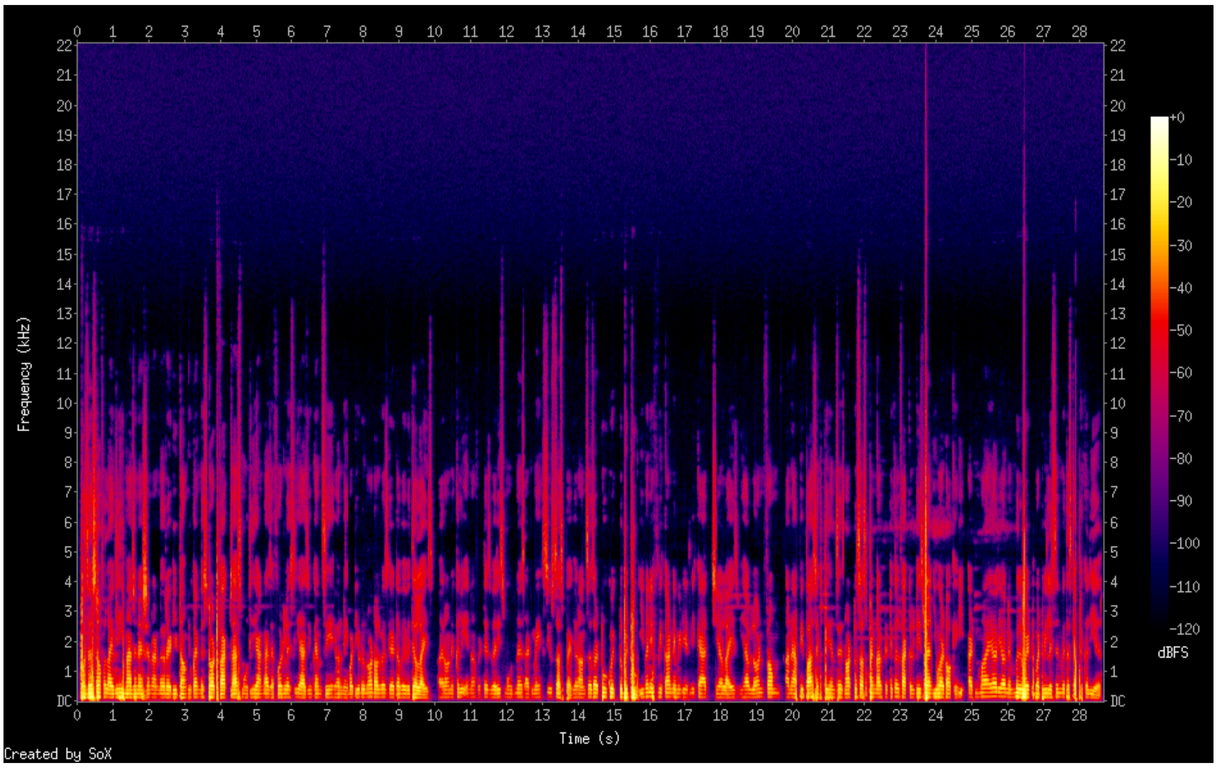


Figure 4.4 Manually Processed using Audacity

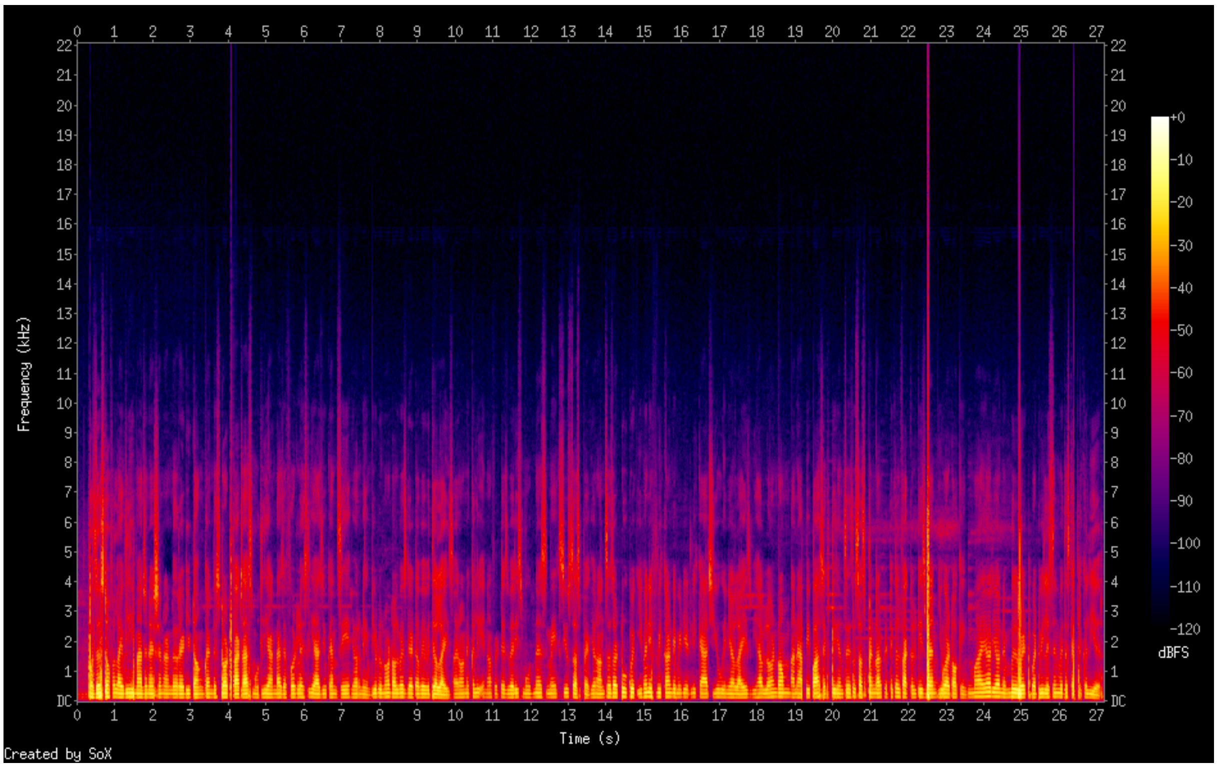


Figure 4.5 Processed using SoX script

4.2.2 Feature Extraction

MFCC(Mel Frequency Cepstral Coefficients) is the state of the art technique in speech feature extraction.They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale (which is a logarithmic scale), which approximates the human auditory system more closely than the linearly-spaced frequency bands used in the normal cepstrum.[11] This frequency warping can allow for better representation of sound. MFCC have found application in various speech recognition applications.

We have used **python_speech_features**, an audio processing library which extracts MFCC features from a given wav format audio file. Filter Bank Energies, which is an intermediate step in extraction of MFCC, is also used along with MFCC feature vectors. The frame size is of 32 ms with a stride of 16 ms. Only initial 13 MFC coefficients are useful since the later ones are nearly zero.

4.2.3 Neural Network Model

There are two types of approaches to training the neural network using the MFCC vectors.

Nearly every research paper suggests that features of upto 40 frames be stacked and given collectively as input. This is necessary for speech dependent system, where the sequence of words/frames matter.

Since we were focussing only on text independent aspect and are not concerned with the order of frames, the size of input vector corresponds only to the size of features of 1 frame. Every frame has 52 features which includes 13 MFCC, 13 deltas derived from MFCCs, 13 acceleration derived from deltas and 13 filterbank energies. The coefficients were normalized by subtracting the mean and dividing by the standard deviation. 800 such frames are extracted from processed audio and are used to enroll a speaker.

We have adopted an approach which tests two learning models, namely Dense Neural Network (DNN) and Convolutional Neural Network (CNN). We have used DNN because as stated in [5], DNN provides better noise immunity over the next best performing model, i.e. GMM. CNN is tried because CNN is innately used for identifying patterns in the data/features and scales well which is of essence in this problem.

The neural network has been implemented using the Tensorflow API from Google as backend and Keras API as the front end.

DNN : The first layer consists of 3000 nodes followed by 4 layers of 100 nodes each and a dropout of 0.3 between each of them. The dropouts help in avoiding overfitting of data.

CNN : The CNN contains 4 layers, 3 convolution layers and a dense layer in the end. First 2 layers use 52 convolution kernels with filter lengths (window size) of 13 and 7 respectively. The third conv layer uses 13 output kernels with window size of 3. The output is then flattened and a dense layer with 1000 nodes follows. 0.25 of dropout is added at this stage to avoid overfitting. Then, final output layer follows. All layers use 'tanh' as the activation function, except the final layer, which uses 'softmax'.

Testing is done by recording the data, preprocessing it, and checking it against the model. Since a single voice frame is too small (32 ms), 50 such frames from the test audio are tested. The speaker with the highest probability is chosen as the recognized speaker.

Chapter 5

Implementation

The project is implemented mainly in Python 3, with HTML, CSS and JS used for front end development. Flask is the web framework used along with Keras with Tensorflow in the backend. Given below is the functional diagram of the system.

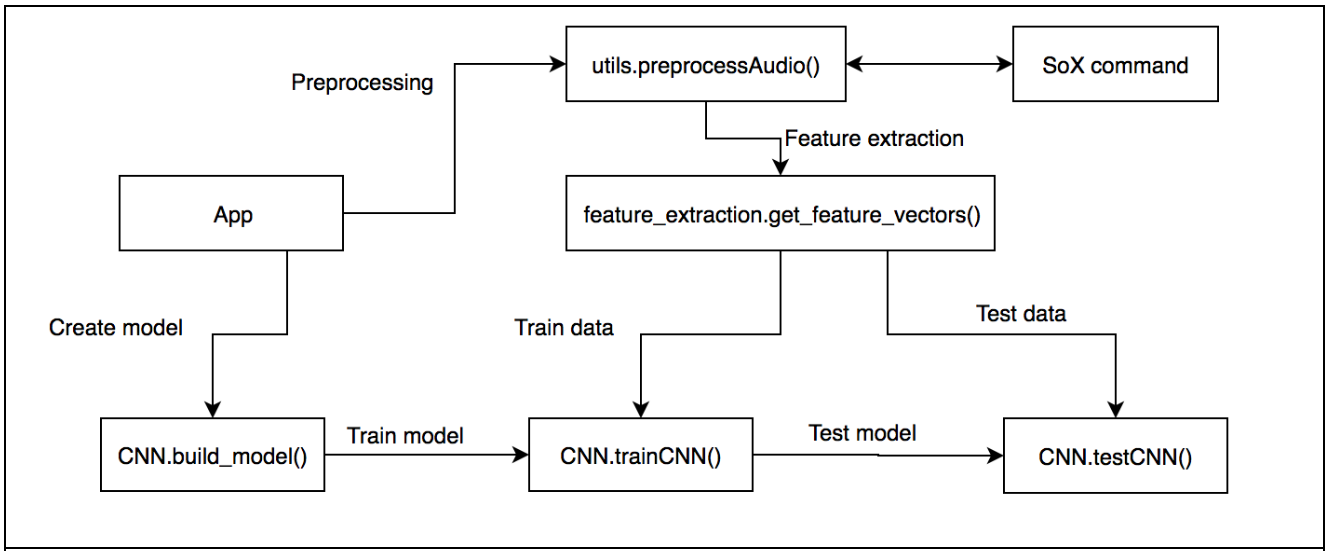


Figure 5.0

Important snippets of code of the project are provided below.

5.1 Application Code

Structure:

```
.
├── static
│   ├── recorder.js
│   └── style.css
├── templates
│   ├── enroll_speaker.html
│   ├── error.html
│   ├── index.html
│   ├── layout.html
│   └── recognize_speaker.html
├── app.py
├── CNN.py
├── config.py
├── feature_extraction.py
├── __init__.py
└── utils.py
```

utils.py

```
from flask import current_app as app

import os
import scipy.io.wavfile as wav


def preprocess_train(file, index):
    source = app.config['RAW_TRAIN_FOLDER']
    dest = app.config['PROCESSED_TRAIN_FOLDER']
    source = os.path.join(source, file)
    dest = os.path.join(dest, '{:03d}_{}'.format(index, file))
    preprocess_audio(source, dest)


def preprocess_test(file):
    source = app.config['RAW_TEST_FOLDER']
    dest = app.config['PROCESSED_TEST_FOLDER']
    source = os.path.join(source, file)
    dest = os.path.join(dest, file)
    preprocess_audio(source, dest)


def preprocess_audio(source, dest):
    (rate,sig) = wav.read(source)
    if int(rate) > 8000:
        cmd = "sox {} {} silence 1 0.3 -45d -1 0.1 1% lowpass 7000".format(source, dest)
        os.system(cmd);
    else:
        cmd = "sox {} {} silence 1 0.3 -55d -1 0.1 1% lowpass 3500".format(source, dest)
        os.system(cmd);
```

feature_extraction.py

```
from flask import current_app as app

import os
import numpy as numpy
import scipy.io.wavfile as wav
from python_speech_features import mfcc, delta, logfbank


no_of_features = 13
no_of_fbank_features = 13
no_of_columns = (3 * no_of_features) + no_of_fbank_features


def get_feature_vectors(file, directory, no_of_frames, start_frame):
    (rate,sig) = wav.read(os.path.join(directory, file))
    fbank_feat = logfbank(sig,rate,nfft=2048)
    mfcc_feat = mfcc(sig,rate,winlen=0.032,winstep=0.016,numcep=13,nfft=2048)

    d_mfcc_feat = delta(mfcc_feat, 2)
    dd_mfcc_feat = delta(d_mfcc_feat, 2)

    mfcc_vectors = mfcc_feat[start_frame:start_frame+no_of_frames,:no_of_features]
    dmfcc_vectors = d_mfcc_feat[start_frame:start_frame+no_of_frames,:no_of_features]
    ddmfcc_vectors = dd_mfcc_feat[start_frame:start_frame+no_of_frames,:no_of_features]
    fbank_vectors =
fbank_feat[start_frame:start_frame+no_of_frames,:no_of_fbank_features]

    feature_vectors = numpy.hstack((mfcc_vectors, dmfcc_vectors, ddmfcc_vectors,
fbank_vectors))
    return feature_vectors


def get_feature_vectors_with_index(file, directory, no_of_frames, start_frame):
    feature_vectors = get_feature_vectors(file, directory, no_of_frames, start_frame)
    # get speaker index from filename
    speaker_index = int(file.split("_")[0])
    #append speaker index to feature vectors
    np_speaker_index = numpy.array([speaker_index])
    temp = numpy.tile(np_speaker_index[numpy.newaxis,:], (feature_vectors.shape[0],1))
    concatenated_feature_vector = numpy.concatenate((feature_vectors,temp), axis=1)
    return concatenated_feature_vector
```

CNN.py

```
from flask import current_app as app

import os
import numpy as numpy
from keras.models import Sequential
from keras.layers import Convolution1D, MaxPooling1D, Dense, Dropout, Activation,
Flatten, Reshape
from keras.optimizers import SGD
from keras.utils import np_utils
from keras.callbacks import Callback
from app.feature_extraction import get_feature_vectors_with_index, get_feature_vectors,
no_of_columns
from app.utils import find_majority, get_test_speaker_name


cnn_model = None
classes = None
mean = None
std_deviation = None
speaker_name = "None"
```

```
def trainCNN():
    global cnn_model, classes, mean, std_deviation
    directory = app.config['PROCESSED_TRAIN_FOLDER']
    no_of_frames = 800
    start_frame = 10
    classes = len(os.listdir(directory)) + 2
    dataset = numpy.empty([0, no_of_columns + 1])

    for file in os.listdir(directory):
        dataset = numpy.concatenate((dataset, get_feature_vectors_with_index(file,
directory, no_of_frames, start_frame)), axis=0)

    my_data = dataset
    numpy.random.shuffle(my_data)
    Y = numpy.copy(my_data[:, no_of_columns:])
    X = numpy.copy(my_data[:, :no_of_columns])
    mean = X.mean(0, keepdims=True)
    std_deviation = numpy.std(X, axis=0, keepdims=True)
    normalized_X = (X - mean) / std_deviation
    one_hot_labels = np_utils.to_categorical(Y, num_classes=classes+1)

    cnn_model = build_cnn(normalized_X, one_hot_labels, classes)


def build_cnn(normalized_X, one_hot_labels, classes):
    temp = normalized_X.reshape(normalized_X.shape[0], no_of_columns, 1)

    model = Sequential()
    model.add(Convolution1D(52, 13, activation='tanh', input_shape=(no_of_columns,1)))
    model.add(Convolution1D(52, 7, activation='tanh'))
    model.add(Convolution1D(13, 3, activation='tanh'))
    model.add(MaxPooling1D(pool_size=(1)))
    model.add(Flatten())
    model.add(Dense(1000, activation='tanh'))
    model.add(Dropout(0.25))
    model.add(Dense(classes+1, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.fit(temp, one_hot_labels, epochs=10, batch_size=100, verbose=1,
callbacks=[TrainCallback()])
    return model


class TrainCallback(Callback):
    def on_epoch_end(self, epoch, logs={}):
    print(epoch)


def testCNN():
    global cnn_model, classes, mean, std_deviation, speaker_name
    model = cnn_model
    directory = app.config['PROCESSED_TEST_FOLDER']
    no_of_frames = 50
    test_frames = 50
    start_frame = 1
    test_model = numpy.empty([0, no_of_columns])

    test_model = numpy.concatenate((test_model, get_feature_vectors('test.wav',
directory, no_of_frames, start_frame)), axis=0)

    test_X = test_model
    normalized_test_X = (test_X - mean) / std_deviation
    test_X = test_X.reshape(test_X.shape[0], no_of_columns, 1)
    normalized_test_X = normalized_test_X.reshape(normalized_test_X.shape[0],
no_of_columns, 1)
    predictions = model.predict(normalized_test_X)
    b = [sum(predictions[current: current+test_frames]) for current in range(0,
len(predictions), test_frames)]
    predicted_Y = []
    for row in b:
        predicted_Y.append(row.argmax(axis=0))
    indices = numpy.argmax(predictions, axis=1)
    majority = []
    for i in range(0, len(indices), test_frames):
        majority.append(find_majority(indices[i:i + test_frames]))
    for p, m in zip(predicted_Y, majority):
        print(p, m[0])
    speaker_name = get_test_speaker_name(p)
    print(speaker_name)
```

5.2 Application UI

We have built a web application using Flask web framework. The application contains a total of 3 pages.

A home page (desktop version in Figure 5.2.0 & mobile in Figure 5.2.2) which has links to the three tasks the application can do :

1. Enroll Speaker(Figure 5.2.4) : This page helps the user enroll his voice into the application for recognition. The application records a minimum of 40 seconds of voice sample for proper training of the model.
2. Train model(Figure 5.2.1) : This task has to be done, once any new user is added to the system.
3. Test(Figure 5.2.3) : This page is similar to the Enroll page, here any enrolled speaker can test the system by recording a 5-10 second voice sample.

Figure 5.2.5 shows log that is recorded on the server side which can be used for debugging the system.

Web Application Screenshots

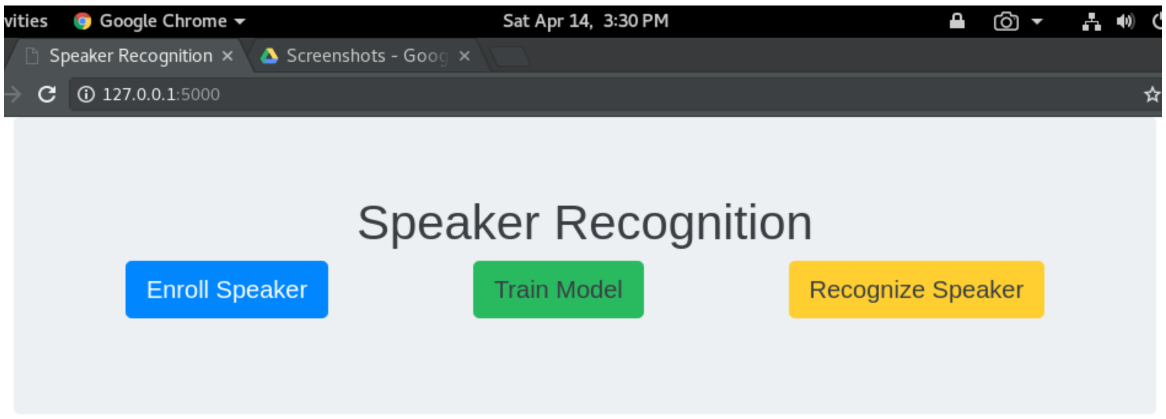


Figure 5.2.0
Home page (Desktop)

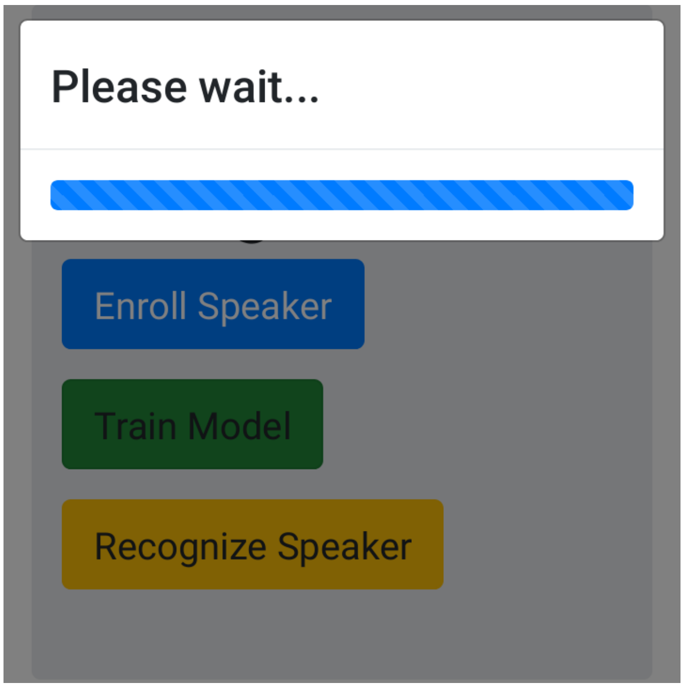


Figure 5.2.1
Training in progress

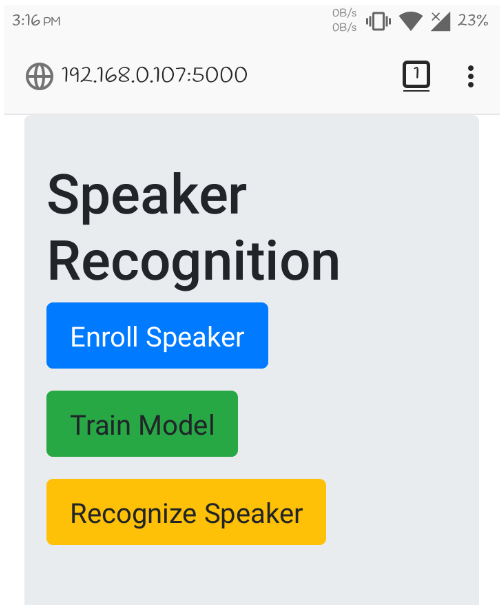
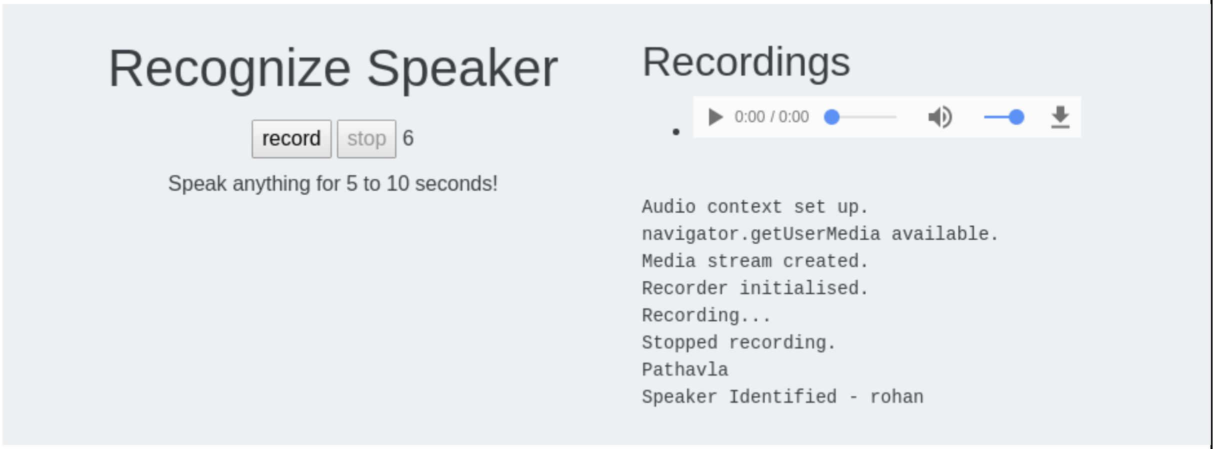
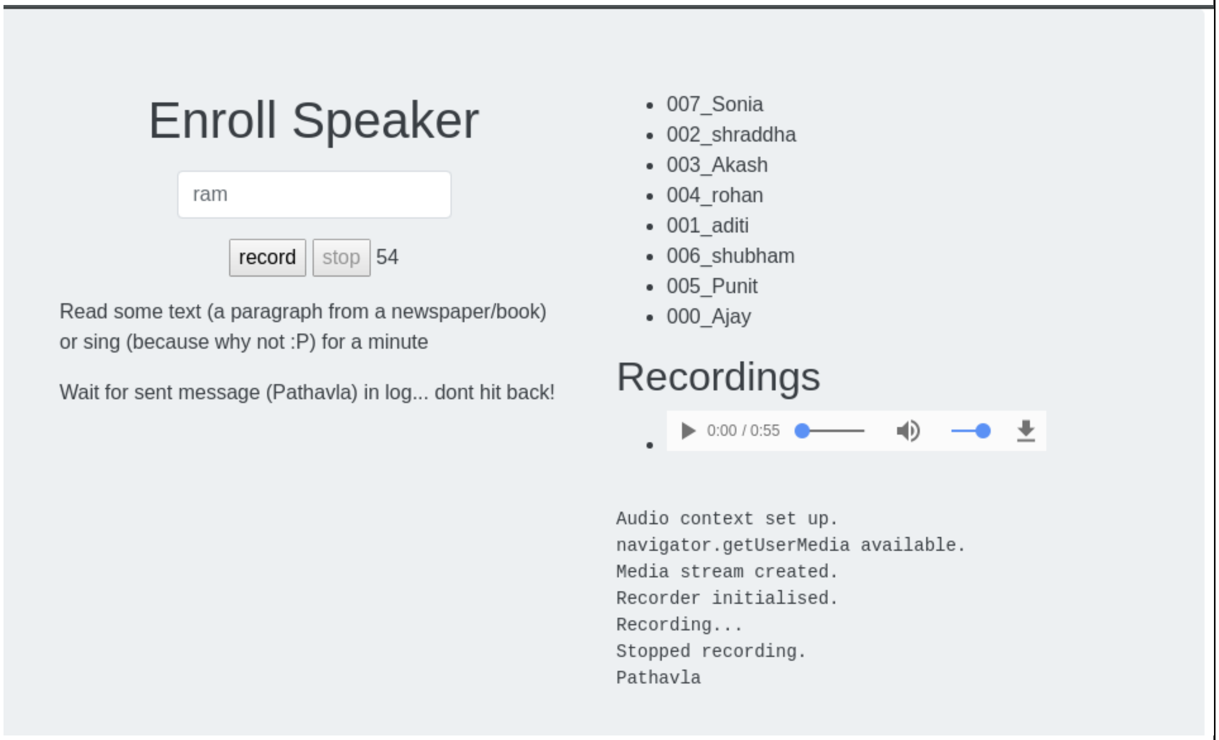


Figure 5.2.2
Home page(Mobile)



```
Audio context set up.
navigator.getUserMedia available.
Media stream created.
Recorder initialised.
Recording...
Stopped recording.
Pathavla
Speaker Identified - rohan
```

Figure 5.2.3
Test page



Read some text (a paragraph from a newspaper/book) or sing (because why not :P) for a minute

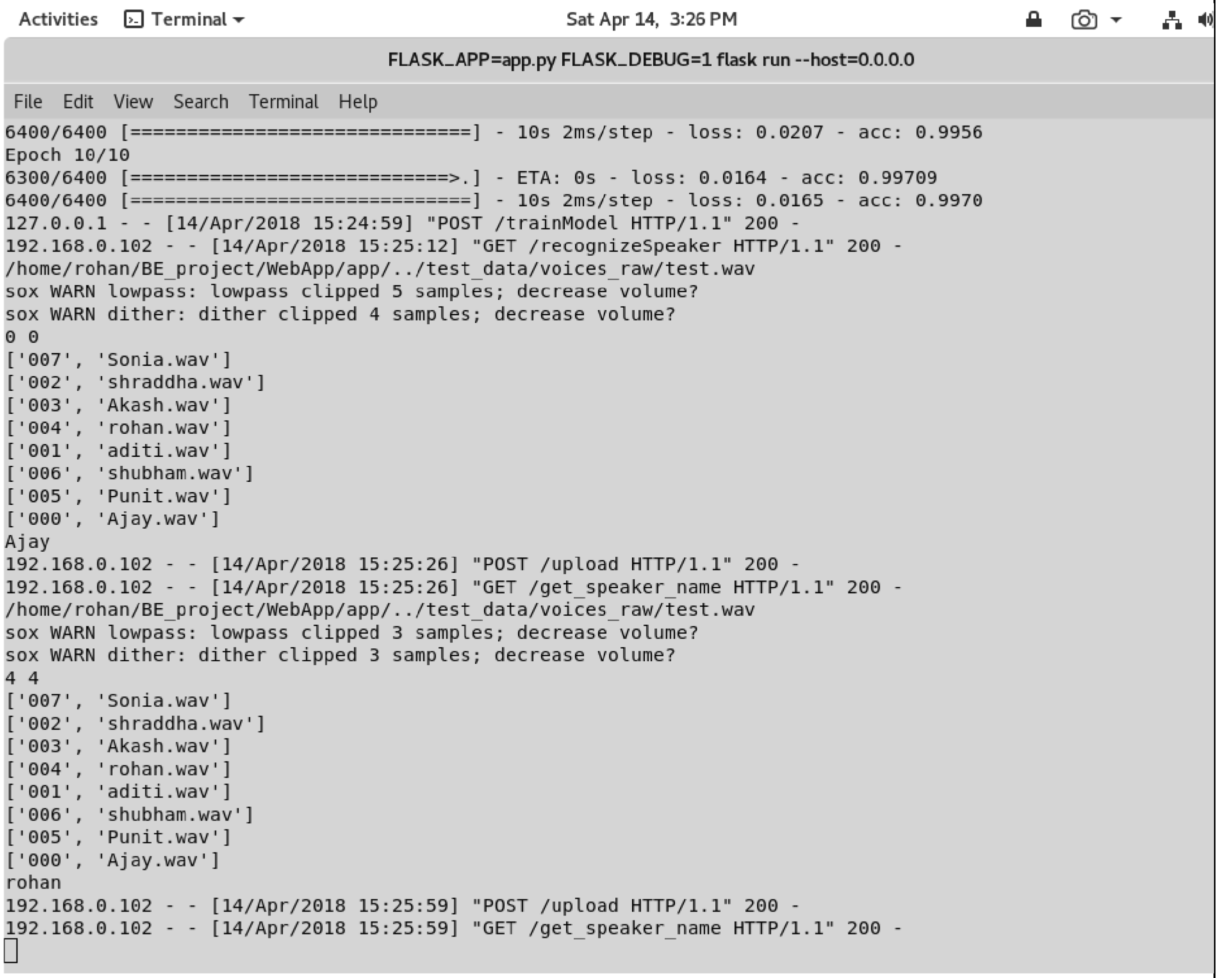
Wait for sent message (Pathavla) in log... dont hit back!

- 007_Sonia
- 002_shraddha
- 003_Akash
- 004_rohan
- 001_aditi
- 006_shubham
- 005_Punit
- 000_Ajay

Recordings

```
Audio context set up.
navigator.getUserMedia available.
Media stream created.
Recorder initialised.
Recording...
Stopped recording.
Pathavla
```

Figure 5.2.4
Enrollment page



```
6400/6400 [==============================] - 10s 2ms/step - loss: 0.0207 - acc: 0.9956
Epoch 10/10
6300/6400 [============================>.] - ETA: 0s - loss: 0.0164 - acc: 0.99709
6400/6400 [==============================] - 10s 2ms/step - loss: 0.0165 - acc: 0.9970
127.0.0.1 - - [14/Apr/2018 15:24:59] "POST /trainModel HTTP/1.1" 200 -
192.168.0.102 - - [14/Apr/2018 15:25:12] "GET /recognizeSpeaker HTTP/1.1" 200 -
/home/rohan/BE_project/WebApp/app/../test_data/voices_raw/test.wav
sox WARN lowpass: lowpass clipped 5 samples; decrease volume?
sox WARN dither: dither clipped 4 samples; decrease volume?
0 0
['007', 'Sonia.wav']
['002', 'shraddha.wav']
['003', 'Akash.wav']
['004', 'rohan.wav']
['001', 'aditi.wav']
['006', 'shubham.wav']
['005', 'Punit.wav']
['000', 'Ajay.wav']
Ajay
192.168.0.102 - - [14/Apr/2018 15:25:26] "POST /upload HTTP/1.1" 200 -
192.168.0.102 - - [14/Apr/2018 15:25:26] "GET /get_speaker_name HTTP/1.1" 200 -
/home/rohan/BE_project/WebApp/app/../test_data/voices_raw/test.wav
sox WARN lowpass: lowpass clipped 3 samples; decrease volume?
sox WARN dither: dither clipped 3 samples; decrease volume?
4 4
['007', 'Sonia.wav']
['002', 'shraddha.wav']
['003', 'Akash.wav']
['004', 'rohan.wav']
['001', 'aditi.wav']
['006', 'shubham.wav']
['005', 'Punit.wav']
['000', 'Ajay.wav']
rohan
192.168.0.102 - - [14/Apr/2018 15:25:59] "POST /upload HTTP/1.1" 200 -
192.168.0.102 - - [14/Apr/2018 15:25:59] "GET /get_speaker_name HTTP/1.1" 200 -
```

Figure 5.2.5
Log of the server running the application

Chapter 6

Results and Discussions

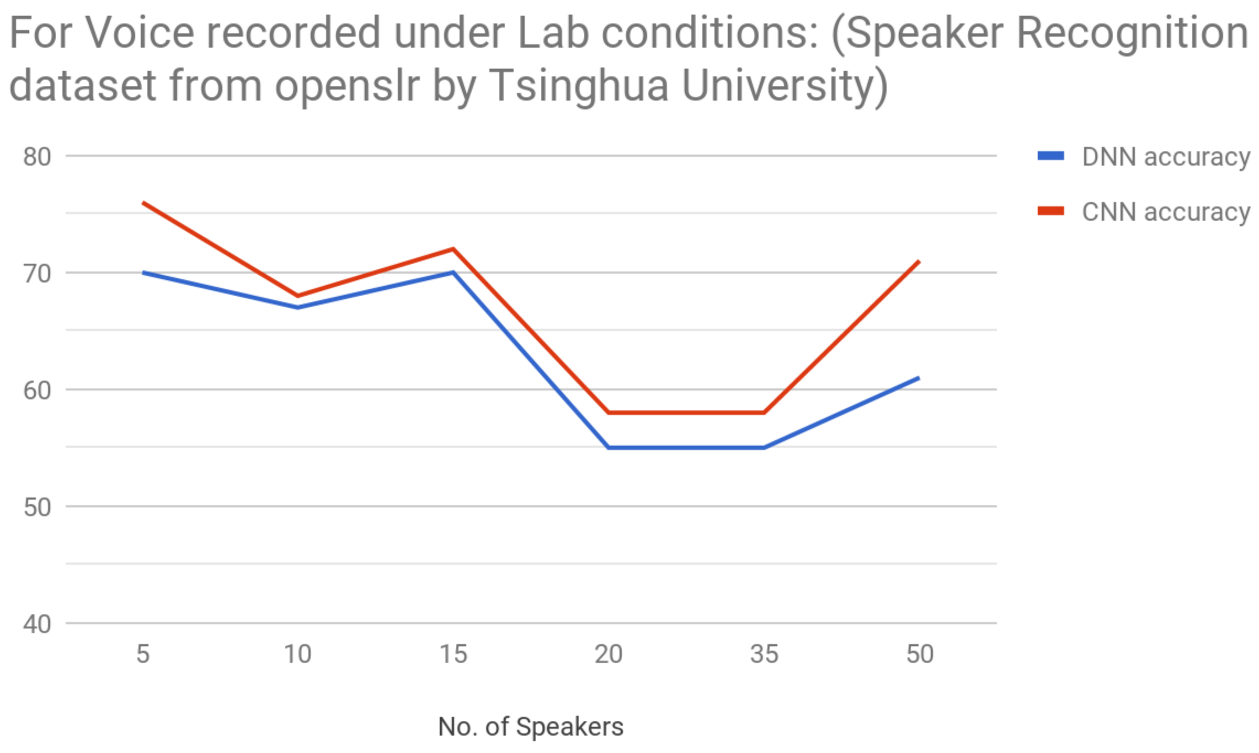
In this chapter, we present and discuss the results we have got from the application by the tests conducted on it under various conditions.

6.1 Results

The voice sample is recorded and processed as mentioned in the previous chapter. 50 frames are extracted from the processed audio and all of them are tested on the neural net. The probability outputs of all frames are added and the speaker with highest probability is the output. The output has 2 classes, either identified, or not.

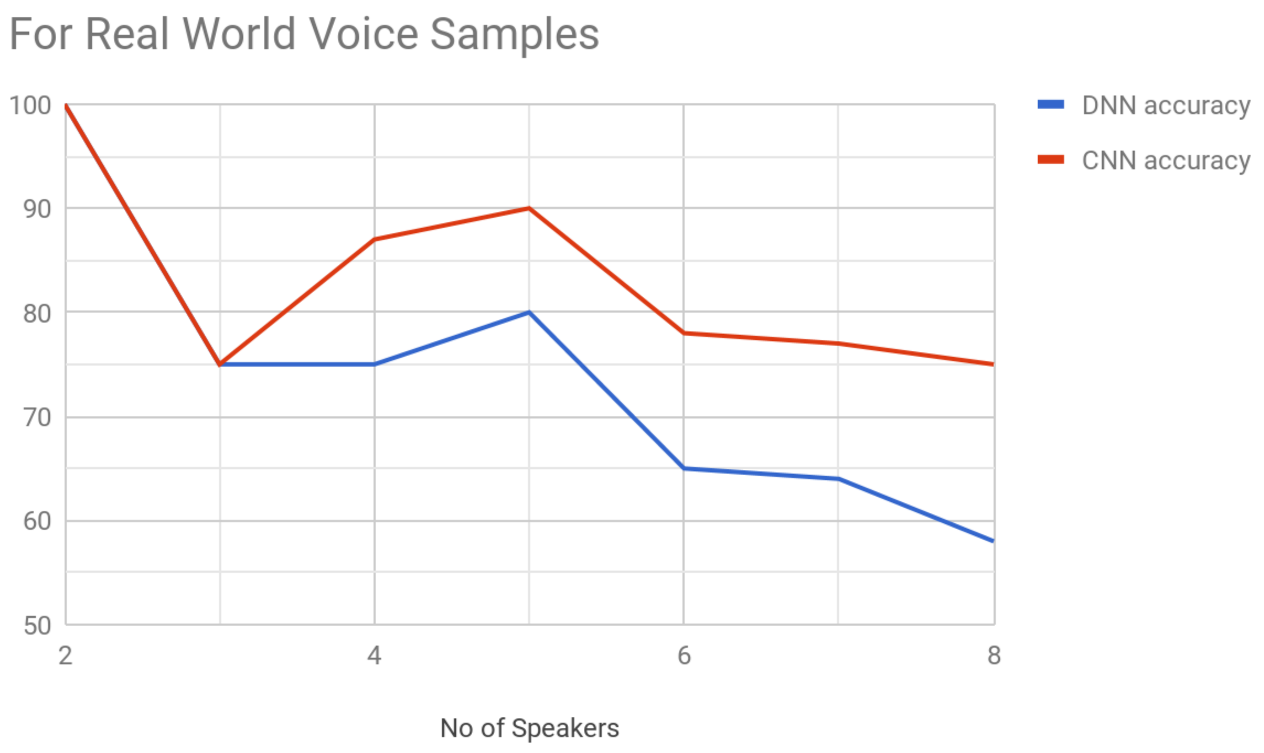
For Voice recorded under Lab conditions: (Speaker Recognition dataset from openslr by Tsinghua University) - Table 1

Table 1		
No. of Speakers	DNN accuracy	CNN accuracy
5	70	76
10	67	68
15	70	72
20	55	58
35	55	58
50	61	71



For Real World Voice Samples - Table 2

Table 2		
No of Speakers	DNN accuracy	CNN accuracy
2	100	100
3	75	75
4	75	87
5	80	90
6	65	78
7	64	77
8	58	75



6.2 Discussions

The testing accuracy of dataset was calculated by testing 50 voice frames against the model and finding the speaker with the highest probability.

In CNN, The accuracy for voice recorded in lab conditions, ie, with minimal noise and a stable microphone is about 71% for a sample of 50 speakers. However for real world conditions, where some noise from fan, background noise from windows or typing noise is inevitable, the accuracy is 75% for upto 8 speakers.

DNN was also tried which resulted in an accuracy of 61% for lab recorded voice and 58% for real world conditions

Pre processing of sample was done using both, SoX and Audacity. Audacity cleared low energy noise completely which resulted in some data loss since some words need lower voices for proper pronunciation. SoX didn't filter the noise completely but there was no data loss.

Chapter 7

Conclusion and Future Scope

In this chapter, we have summarised the conclusions we have arrived at by discussing results as well as from the experiences gained while doing this project. We have also suggested the possible ways the system can be improved and implemented in the future.

7.1 Conclusion

The text independent, language independent speaker recognition system was developed with the idea that the future smart devices should have the ability to recognize their user's voice without the boundaries of languages and keywords.

CNN: Upto 10 speakers can be fed into the net for a decent accuracy of 75-80% for real world samples. For voice recorded in lab conditions, the net gives an accuracy of 70% for upto 50 speakers with some inconsistencies as seen in the previous chapter.

DNN : DNN gives an accuracy of 75-80% for 5 speakers and then its performance degrades to upto 60% for 8 speakers. This is consistent with the fact that CNN is a learning model which excels at identifying patterns in the input and can scale much better than DNN.

Pre processing was better using SoX than Audacity since using Audacity resulted in loss of some data, which reduced the quality of voice sample.

7.2 Future Scope

Deep Learning is one of the newer technologies used in the field of Speaker Recognition. Since this is an extensively researched problem, there have been many studies which show that combining traditional mathematical models with machine learning models are able to give better accuracy than using only one of them.

Techniques such as i-vetcor and Gaussian Mixture Model can be combined with Neural Net to get even higher accuracies. Extracting more features and fine tuning frame size, frequency range, noise processing may give an increase of testing accuracy.

Further studies can be conducted to improve the accuracy of the model and to scale up the number of users. Studies regarding how different age groups would be handled by the model and how the changes in the voice of speaker overtime can be incorporated into it.

Once a good enough accuracy is obtained, its application in IoT devices could bring automation and user personalisation to a whole new level.

Chapter 8

References

Following are the papers and other references researched to build the application

[1] Lior Uzan and Lior Wolf, “I Know That Voice: Identifying the Voice Actor Behind the Voice”, *Biometrics (ICB) International Conference*, 2015

[2] Roger Achkar, Mustafa El-Halabi, Elie Bassil, Rayan Fakhro and Marny Khalil, “Voice Identity Finder Using the Back Propagation Algorithm of an Artificial Neural Network”, *Complex Adaptive Systems Publication 6*, 2016

[3] Geeta Nijhawan and Dr. M.K Soni, “Speaker Recognition Using MFCC and Vector Quantisation”, *Recent Trends in Engineering and Technology Vol. 11*, 2014

[4] Douglas A. Reynolds, “Speaker Identification and verification using Gaussian mixture speaker model”, *Speech Communication archive Vol. 17 Issue 1-2*, 1995

[5] Fred Richardson, “Deep Neural Network Approaches to Speaker and Language Recognition”, *IEEE Signal Processing Letters Vol. 22 No. 10*, 2015

[6] David Snyder, Pegah Ghahremani and Sanjeev Khudanpur, “Deep neural network-based speaker embeddings for end-to-end speaker verification”, *Spoken Labs Spoken Communications*, 2014

[7] Zhenhao Ge, Ananth N. Iyer, Srinath Cheluvaraja, Ram Sundaram and Aravind Ganapathiraju, “Neural Network Based Speaker Classification and Verification Systems with Enhanced Features”, *Intelligent Systems Conference*, 2017

[8] Geeta Nijhawan and M.K. Soni, “A Comparative Study of Two Different Neural Models For Speaker Recognition Systems”, *IJITTE Vol. 1 Issue 1*, 2012

[9] Ahilan Kanagasundaram, David Dean, Sridha Sridharan and Clinton Fookes, “DNN based Speaker Recognition on Short Utterances”, *Cornell University Library*, 2016

[10] Qiyue Liu, Mingqiu Yao, Han Xu and Fang Wang, “Research on Different Feature Parameters in Speaker Recognition”, *Journal of Signal and Information Processing*, 2013

[11] Shahenda Sarhan, Mohamed Abu El Soud, Nagham Mohammed Hasan, July 2015, "Text Independent speaker identification based on MFCC and Deep Neural Networks", https://www.researchgate.net/publication/291165354

[12] Chao Li, ,Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, Zhenyao Zhu ,5 May 2017 ,"Deep Speaker: an End-to-End Neural Speaker Embedding System",arXiv:1705.02304v1 [cs.CL]

[13] Peng Qi, Lu Wang, 2011, "Experiments of GMM Based Speaker Identification", 8th International Conference on Ubiquitous Robots and Ambient Intelligence.

[14] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, Javier Gonzalez-Dominguez,"Deep Neural Networks For Small Footprint Text-dependent Speaker Verification", 2014 IEEE International Conference on Acoustic, Speech and Signal Processing.

[15] Amirsina Torfi, Nasser Nasrabadi, Jeremy Dawson, 6 Nov 2017 "Text-Independent Speaker Verification Using 3D Convolutional Neural Networks", arXiv:1705.09422v4 [cs.CV]

[16] Fundamentals of Speaker Recognition - Homayoon Beigi

[17] Improved Text-Independent Speaker Recognition using Gaussian Mixture Probabilities Balakrishnan Narayanaswamy

[18] Joint Speech and Speaker Recognition Using Neural Networks - Xiaoguo Xue

[19] Feature Extraction - http://recognize-speech.com/feature-extraction