Accent Classification using Deep Learning

Gaurav Singh - 191IT218
Information Technology
National Institute of Technology, Karnataka
Surathkal, India 575025
gauravsingh.191it218@nitk.edu.in

Navyasree B - 191IT135
Information Technology
National Institute of Technology, Karnataka
Surathkal, India 575025
navya.191it135@nitk.edu.in

Abhinav Dugar - 191IT202
Information Technology
National Institute of Technology, Karnataka
Surathkal, India 575025
abhinavdugar191it202@nitk.edu.in

Niraj Nandish - 191IT234
Information Technology
National Institute of Technology, Karnataka
Surathkal, India 575025
nirajn.191it234@nitk.edu.in

***Abstract*—This work describes classification of accents based on the speaker's native language which would enable accent dependent Automatic Speech Recognition. Accurate accent detection has a big role in bringing inclusivity to latest smart products and giving them a global outreach. All the speech files for this project came from Speech Accent Archive which is a repository made by George Mason University. The repository contains over 2000 speakers representing over 100 native languages. However, this project focuses only on three languages Hindi, Arabic and Spanish. Data Augmentation techniques were used to increase the amount of data for each language. Manipulation of the dataset was done using Pandas library. For preprocessing, the audio files were featurized using 13th lowest Mel Frequency Cepstral Coefficients(MFCCs). 2D Convolutional Neural Network model was used. Keras, which is an open-source neural-network library written in Python, was used for the model implementation. The accuracy of our main CNN model was over 76%. We compared this result to the results obtained from Random Forest, MLP and Gradient Boosting models. The accuracy obtained for the other three models were 60.23%, 71.45% and 65.24% respectively.**

I. Introduction

Each individual has their own mannerisms in which they speak. This project revolves around the detection of accents of every individual using their speech. The goal of this project is to classify various accents by the native language of the speakers. Thus this project allows to detect the linguistic and even demographic background of the speakers by comparing different speech outputs with the speech accent archive to determine the variables that are key predictors of each accent. The archive demonstrates that accents are not mere mistaken speech but are actually systematic. The project predicts the speaker's native language after being given a recording of the speaker saying a known script of English words.

There are two motivations to develop this project. First, in call centres, if we can determine the caller's native language by their accent, then we would be able to efficiently route that person to a regional representative speaking a similar language. Secondly, automatic speech recognition (ASR), like Siri or Alexa, uses accent classification. To understand what a person is saying, there must be a model in place that expects how they are going to say what they want to say. A Washington Post survey found out that, non native English speakers were 30% less likely to be understood by Google Home and Amazon's Alexa. Even amongst native english speakers, people with Southern accents were 3-4% less likely to get accurate responses in comparison to those having Western accents. [Accurate accent detection is imperative]Currently there is no existing efficient solution to this problem.

Key Contributions of our work are:

- Develop Accent driven automatic speech recognition to more effectively understand what the user says
- Help call centers to effectively route a user to a regional representative
- Understand user demographic based on the user's accent which would help various platforms to show advertisements based on the user's demographic
- Help Speech recognition systems identify and show topics more relevant to the user based on their accents.

A. Mel Frequency Cepstral Coefficients

The audio files were featurized using the 13th lowest Mel Frequency Cepstral Coefficients(MFCCs). MFCC takes into account human perception for sensitivity at certain frequencies by converting the frequency to Mel Scale and thus are suitable for speech recognitions tasks. We generally take 12-13 Mel Frequency coefficients while training models.
MFCC extraction was done using the Librosa library in python. MFCC takes the input as the numpy array of the audio files in wav format and hence we had to convert the mp3 files to wav and that to a numpy array.

B. Convolutional Neural Network

The audio data is classified with the help of 2D- Convolutional Neural Networks into the three classes. This network

entrails the usage of Convolutional Layers, Max Pooling Layers along with which Batch Normalisation and Dropout are also used to add regularization effects.
The softmax layer at the end gives out the probability distribution of the three classes and gives the required output.

II. Literature Survey

Our base paper for the implementation of the CNN model was the 'Application of Convolutional Neural Networks in Accent Identification' (2019) by Keven Chionh, Maoyuan Song and Yue Yin. They achieved a training accuracy of 0.814 and a test accuracy of 0.779. We further added on to the project and explored traditional neural network models such as Random Forest, MLP and Gradient Boosting to compare the results obtained by each. A few other relevant papers along with their methodology and accuracy achieved are mentioned in the table below.

TABLE I
Comparison of relevant papers in accent detection

Name of paper	Author(s)	Methodology	Remarks
Application of Convolutional Neural Networks in Accent Identification	Keven Chionh, Maoyuan Song, Yue Yin	Random Forest, MLP and Gradient Boosting	Accuracy: 77.9% Only foreign accents taken(Japanese, Korean, Italian, Arabic)
Deep Learning Approach to Accent Classification	Leon Mak An Sheng, Mok Wei Xiong Edmund	Using Random Forest, Gradient Boosting, MLP and CNN	Accuracy: 88% Only foreign accents taken into account, no Indian regional accents.
VFNet: A Convolutional Architecture for Accent Classification	Asad Ahmed, Pratham Tangri, Anirban Panda, Dhruv Ramani, Samarjit Karmakar Nevr´onas	VFNet model	Accuracy: 70.33% Only English, Arabic and Mandarin accents used.
Deep Learning for Classification of Speech Accents in Video Games	Sergio Poo Hernandez and Vadim Bulitko Shelby Carleton Astrid Ensslin and Tejasvi Goorimoorthee	AlexNet model	Accuracy: 71% Only British and American accents identified.

III. Problem Statement

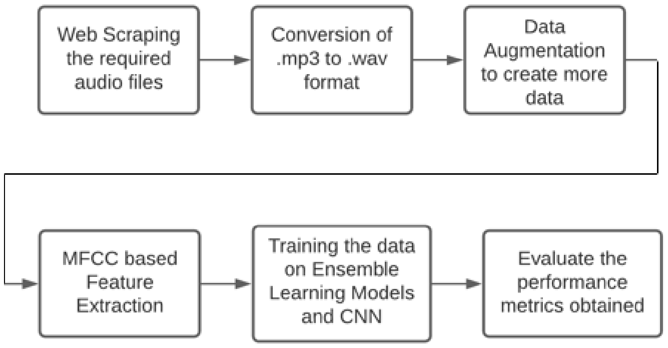
Accent Detection for Audio Classification solved using Deep Learning Techniques

A. Objectives

1) Web scraping the required audio files from Speech Accent Archive along with the biographical data
2) Conversion of .mp3 to .wav format and Data Augmentation to create more data
3) Feature Extraction of Audio Files using MFCC
4) Training the obtained data on a Convolutional Neural Networks based model
5) Extracting the performance metrics from the model

IV. Methodology

The Pipeline of the project is as displayed below:


Web Scraping the required audio files
Conversion of .mp3 to .wav format
Data Augmentation to create more data
MFCC based Feature Extraction
Training the data on Ensemble Learning Models and CNN
Evaluate the performance metrics obtained

A. Web Scraping

The dataset used for this project was obtained by scraping through the Speech Accent Archive (http://accent.gmu.edu/), which is a repository of Spoken English created by George Mason University. Overall it contains 2000 speakers representing over 100 native languages who read the same paragraph in English:

Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station.

Since, all the speakers are saying a common sentence in English and the recording quality is almost uniform across all speakers, this dataset is ideal for studying accents.

Along with the audio files, the biographical data of each speaker was saved to a csv file, for future manipulation using Pandas Library.

B. Conversion of MP3 to WAV

The scraped .mp3 files are then converted into .wav format, as it is an uncompressed audio format and is more universally used for audio processing tasks.

C. Data Augmentation

The amount of data for regional languages present in the data set was insufficient for efficient training of the model. Existing audio files were augmented to get more data. Data Augmentation techniques we used includes addition of white noise, rolling of audio, changing of pitch and stretching of

audio files.
For addition of white noise, we used noise factor of 0.005 and 0.0009. Shifting of audio was done using np.roll() with shift of 1600 and 90000. `librosa.effects.times_stretch` function was used to stretch audio by factors of 0.8 and 1.2 Finally, `librosa.effects.pitch_shift` was used to change pitch by factors of 0.1 and 0.2

D. MFCC based Feature Extraction

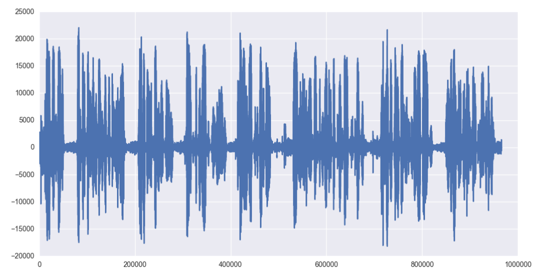


Fig. 1. Waveform of audio file

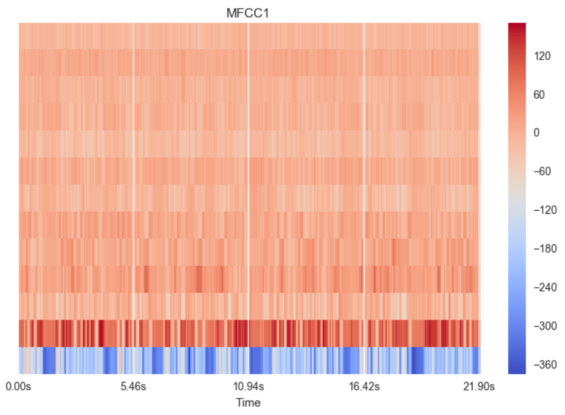


Fig. 2. Visual Representation of MFCC

1) Pre-Emphasis: Here filtering occurs in which more importance is given to the higher frequencies. Balancing of the spectrum of voiced sounds that have a high roll-off in the high frequency region is done here. The pre-emphasis filter is given by the following function:

$$H(z) = 1 - bz^{-1} \tag{1}$$

where *b* controls the slope of the filter and is usually between 0.4 and 1.0 .[1]

2) Frame Blocking and Windowing: For stable acoustic properties, audio needs to be checked over a required short period of time. Short-term spectral measurements are typically carried out over 20 ms windows, and advanced every 10 ms[2,3]. for each frame, a window is applied to change the signal towards the frame's specific boundaries. Generally, Hanning or Hamming windows are used[1].

3) DFT Spectrum: Each windowed frame is converted into magnitude spectrum by applying DFT.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}}; \qquad 0 \le k \le N-1 \tag{2}$$

where N is the number of points used to compute the DFT.

4) Mel Spectrum: It is calculated by passing the Fourier transformed signal through a set of band-pass filters known as Mel-filter bank. A Mel is a unit based on frequency perceived by human ears. The Mel scale is a linear frequency spacing below 1 kHz and a logarithmic spacing above 1 kHz.[4] The approximation of Mel is as follows:

$$f_{Mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3}$$

where f denotes the physical frequency in Hz, and f_{Mel} denotes the perceived frequency.[2]

Filter banks are generally implemented in frequency domain of Mel-filter banks for MFCC calculation with the central frequencies of the filter spaced evenly around the frequency axis. Triangular filter or Hanning filter are commonly used.

The Mel Spectrum $X(k)$ is computed by multiplying the magnitude spectrum by each of the triangular Mel weighting filters.

$$s(m) = \sum_{k=0}^{N-1} \left[|X(k)|^2 H_m(k) \right]; \qquad 0 \le m \le M-1 \tag{4}$$

where M is total number of triangular Mel weighting filters[5,6]. $H_m(k)$ is the weight given to the k^{th} energy spectrum contributing to the m^{th} output band is expressed as:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{2(k-f(m-1))}{f(m)-f(m-1)} & f(m-1) \le k \le f(m) \\ \frac{2(f(m+1)-k)}{f(m+1)-f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{5}$$

with M ranging from 0 to $M-1$.

5) Discrete cosine transform(DCT): It is applied to the transformed Mel frequency coefficients to give a result which is a set of cepstral coefficients. It is represented on a log scale which results in a signal in cepstral domain with quefrequency peak corresponding to pitch of the signal and a number of formats.

$$c(n) = \sum_{m=0}^{M-1} \log_{10}(s(m)) \cos\left(\frac{\pi n(m-0.5)}{M}\right); \qquad n = 0,1,2,...,C-1 \tag{6}$$

where *c(n)* are the cepstral coefficients, and *C* is the number of MFCCs.[1]

6) Dynamic MFCC features: Cepstral coefficients are the static features because they contain information from a given frame. More information about the dynamics of the signal is obtained by computing 1^{st} and 2^{nd} derivative of cepstral coefficients.[7-9] 1^{st} order derivative is called delta coefficients which gives information about speech rate and 2^{nd} order derivative is called delta delta coefficients which give information similar to acceleration of speech.[7]

$$\Delta c_m(n) = \frac{\sum_{i=-T}^{T} k_i c_m(n+i)}{\sum_{i=-T}^{T} |i|} \tag{7}$$

where $c_m(n)$ denotes the m^{th} feature for the n^{th} time frame, k_i is the i^{th} weight, and T is the number of successive frames used for computation.

E. Machine Learning Models

1) Ensemble Learning Methods: We first used traditional machine learning methods like Random Forest and Gradient Boosting. Random Forest is a classification algorithm consisting of many decision trees,it uses bagging and features randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.Whereas Gradient boosting is a type of machine learning boosting which relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error.This implementation was done with the help of the Scikit Learn Library. Since these models were to be used just for Baseline Performance with our neural network we used the default values provided by the scikit learn library.

2) Multi Level Perceptron: This neural network architecture consists of various stacked fully connected layers of neurons. The last layer of the MLP is a softmax layer, which gives the predictions for the three classes

Prediction for each input data in the batch is made by forward propagation and the loss is computed by categorical cross-entropy loss function. Loss is then back propagated to find the error and the weight is then adjusted so as to descend the loss function and decrease the loss value.

$$L_i = -\sum_j t_{ij} log(p_{ij}) \tag{8}$$

$$softmax(y)_i = \frac{exp(y_i)}{\sum_j exp(y_j)} \tag{9}$$

*The categorical cross-entropy loss function (**Equation 8**) and softmax function(**Equation 9**)*

3) 2D - Convolutional Neural Network: We used 2D-CNN Architecture which was implemented using the Keras neural network Library consisted of 2 convolutional layers with 3X3 filters and Rectified Linear Unit Layers (ReLu) as an activation function x := max(0, x) and max pooling layers were used with filter dimension of 2X2. Batch Normalisation has been used to reduce the training time.

The convolutional layers learn local patterns and preserve relationships between various pixels using subsections of the input data. While, densely connected layers do not have this property and instead learn global features.

The max pooling layers help in reducing the dimensions of the input and hence reducing the computational time. They also help in retaining the important features from the input data.

The densely connected layers are the final layers in which the input is received after flattening it into 1 Dimension, the last layer has a softmax layer which lets us know the probability of each of the output class. Our optimizer for the model is Adadelta which is a more robust extension of Adagrad that adapts to the learning rates based on a moving window of gradient updates.

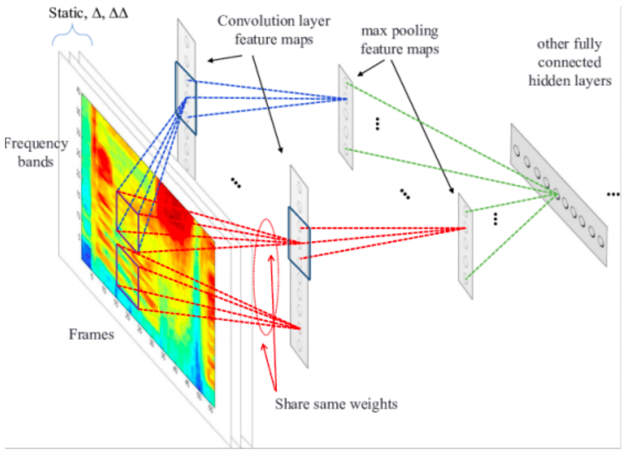


Fig. 3. General architecture of CNN[10]

$$J_{L2} = J + \frac{\lambda}{2}||W||^2 \tag{10}$$

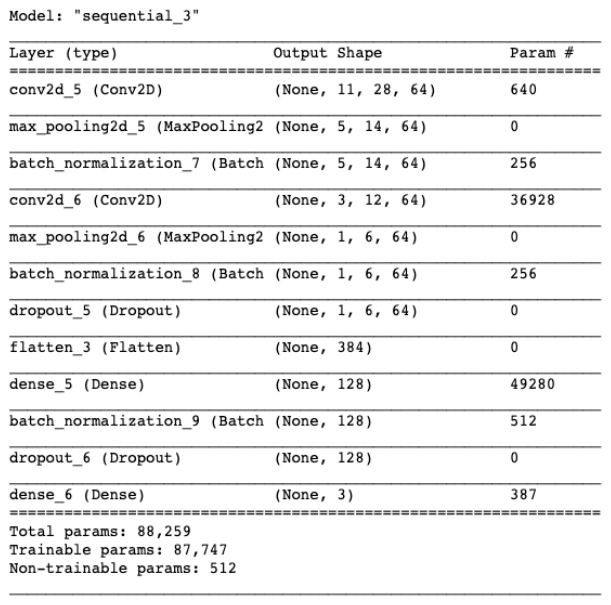
L2 Regularisation in Neural Networks

For Regularisation we have used Dropout, which drops the hidden units and their connections between the layers. We have also used L2 Regularisation to reduce overfitting.

V. Results

A. Model Analysis

The below table contains the model and its respective test accuracy obtained. Random Forest and Gradient Boosting



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 11, 28, 64)        640
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 14, 64)         0
_________________________________________________________________
batch_normalization_7 (Batch (None, 5, 14, 64)         256
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 3, 12, 64)         36928
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 1, 6, 64)          0
_________________________________________________________________
batch_normalization_8 (Batch (None, 1, 6, 64)          256
_________________________________________________________________
dropout_5 (Dropout)          (None, 1, 6, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 384)               0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               49280
_________________________________________________________________
batch_normalization_9 (Batch (None, 128)               512
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 3)                 387
=================================================================
Total params: 88,259
Trainable params: 87,747
Non-trainable params: 512
_________________________________________________________________
```

Fig. 4. Summary of CNN

models were used to create a baseline.

Model	Test Accuracy(%)
Random Forest	60.23%
Gradient Boosting	65.24%
MLP	71.45%
CNN	74.66%

The ensemble learning methods, Random Forest and Gradient Boosting gave a respectable accuracy of around 60-65% on the test data individually.Even though the ensemble learning methods were not as complex as compared to the neural network models, they were easier to implement and gave quicker results.

B. Neural Network Analysis

We did not use pre-trained models such as VGG,ResNet etc, this is because they have much more number of layers compared to our 2D-Convolutional Neural Network and since our data isn't as complex as compared to real life objects we do not require them. We have implemented both the models (MLP and 2D-CNN) using both Data Augmentation and without Data Augmentation, the results obtained are shown below:

Model	Data Augmentation	Train/Validation/Test Accuracy(%)
MLP	No	67.19 / 58.86 / 63.44
MLP	Yes	78 / 69.45 / 70
CNN	No	70.27 / 60.24 / 66.13
CNN	Yes	78.34 / 69.76 /74.66

From the above results we can see that the CNN model performed much better than the MLP model, as we had expected it to. This is due to the ability of Convolutional Neural Networks to work much better with image like input, like in our case where we had extracted MFCC's of the audio files and provided it as the input for the model.

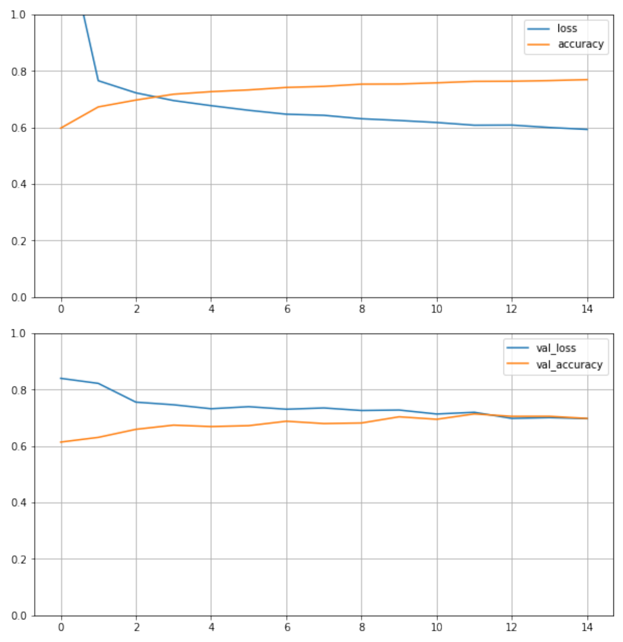


Fig. 5. Training and Validation Accuracy for 2D-CNN

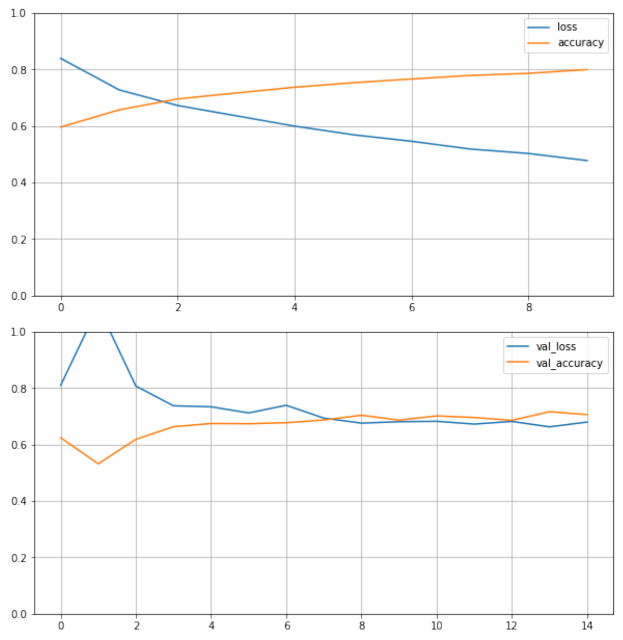


Fig. 6. Training and Validation Accuracy for MLP

We used 13 MFCC features for the CNN model and increasing it did not give much difference in the overall accuracy for the model. Since every speech instance was of different length we chose to trim down the length of all the audio files into 30 seconds, which gave optimum results.
Data Augmentation gave better results and helped boost the performance of the neural network models as evident from the results table. We could only apply limited augmentation to the data due to low computing power of our systems.

It is also evident that regularisation methods such as L2 Regularisation and Dropout helped improve the accuracy of the models.

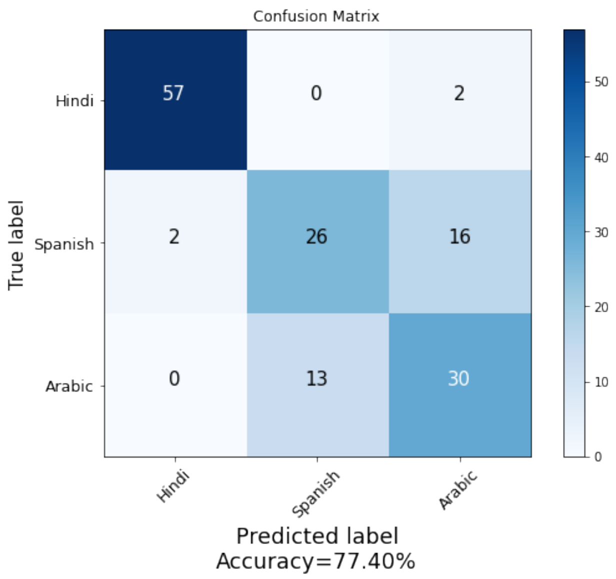


Fig. 7. Confusion Matrix Plot

TABLE II
EVALUATION METRICS

Native Language	Precision	Recall	F1 Score
Hindi	0.97	0.97	0.97
Spanish	0.67	0.59	0.63
Arabic	0.62	0.70	0.66

VI. CONCLUSIONS

This project explores the working of Ensemble Learning Models and 2D Convolutional Neural Network models in detection and classification of accents. Currently, it deals with three languages, namely, Spanish, Arabic and Hindi and we have an accuracy of 74.6%.This is appreciable result considering that the languages used did not have enough data to train the model. For instance, English gave 99% accuracy because it was the only language with enough data, but since the aim of this project was to develop accent classification for lesser popular languages, English was not used. We aim to incorporate more regional languages and improve our accuracy.Due to data augmentation techniques we were able to increase the accuracy of the languages to some extent and were able to achieve 97% accuracy with Hindi. Other techniques of preprocessing apart from MFCCs can be experimented with. Future Work for the project would involve creating an interface that tests the model with audio input from the user.

VII. ACKNOWLEDGEMENT

We would like to thank Professor Ram Mohana Reddy Guddeti for giving us this opportunity to get a hands-on experience of working with neural networks. We are also truly indebted to Natesh sir for his valuable inputs during the course of this project.

INDIVIDUAL CONTRIBUTION

The Gantt Chart and Individual Contribution table is given as follows:

Team member	Work done
Niraj Nandish	Extraction of .mp3 files using web scraping Evaluation Metrics
Navyasree B	Conversion of .mp3 to .wav Data Augmentation
Gaurav Singh	MFCC Feature extraction Preparation of input data for the model
Abhinav Dugar	Web Scraping Biographical Data into .csv format Built Neural Network Model Built Ensemble Learning Model

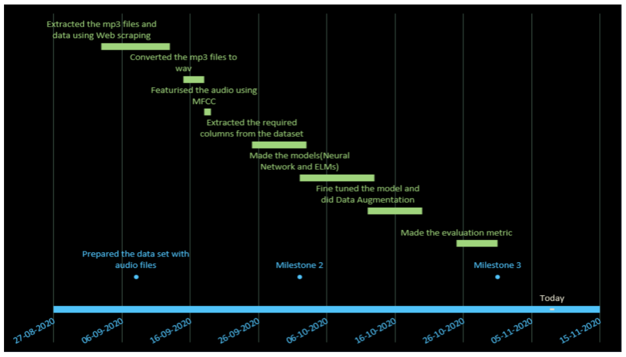


Fig. 8. Gantt Chart

The Github Repository can be accessed here:
(https://github.com/abhid2001/Accent-Classification)

BASE PAPER

Application of Convolutional Neural Networks in Accent Identification (2019) by Keven Chionh, Maoyuan Song and Yue Yin

VIII. REFERENCES

1. J.W. Picone,Signal modeling techniques in speech recognition. Proc.IEEE 81,1215–1247(1993)

2. J.R. Deller, J.H. Hansen, J.G. Proakis, Discrete Time Processing of Speech Signals

3. J. Benesty, M.M. Sondhi, Y.A. Huang, Handbook of Speech Processing (Springer, New York, 2008)

4. J. Volkmann, S. Stevens, E. Newman, A scale for the measurement of the psychological magnitude pitch.

5. Z. Fang, Z. Guoliang, S. Zhanjiang, Comparison of different implementations of MFCC. J. Comput. Sci. Technol. 16, 582–589 (2000)

6. G.K.T. Ganchev, N. Fakotakis, Comparative evaluation of various MFCC implementations on the speaker verification task, in Proceedings of International Conference on Speech and Computer (SPECOM) (2005)

7. L. Rabiner, B.-H. Juang, B. Yegnanarayana, Fundamentals of Speech Recognition (Pearson Education, London, 2008)

8. S. Furui, Comparison of speaker recognition methods using statistical features and dynamic features.

9. J.S. Mason, X. Zhang, Velocity and acceleration features in speaker recognition, in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)

10. Leon Mak An Sheng, Mok Wei Xiong Edmundm Deep Learning Approach to Accent Classification (Stanford)