SPEAKER RECOGNITION USING GMM CLASSIFIER

I. INTRODUCTION AND OBJECTIVES

In today's era of data technology, audio information plays a crucial role in communication as well as increasing volume of data. Using these audio data for biometric identification can be very helpful in improving security. Speech is the primary form of human communication and plays an important part in understanding behavior and cognition. Speech Recognition using Artificial intelligence is a technique that is used to understand and recognize the words being spoken by a speaker.

Speaker recognition is the process of automatically recognizing the speaker by using the speaker-specific information included in speech waves to verify identities being claimed by people accessing systems.[1] Speaker recognition systems are generally categorized as two types : text dependent and text independent. Text dependent recognition is done when the speaker enunciates a pre-defined paraphrase, whereas text independent recognition takes features from a speaker, trains on this data and thereafter, recognises the speaker later on. Unlike fingerprint detection, retinal scans or face recognition, speaker recognition only uses a microphone to record a person's voice, therefore also cutting down the expenses of expensive machines used for authentication.

The purpose of this project is to implement text independent speaker recognition in Python and investigate this approach using Gaussian Mixture Model (GMM), moreover, to evaluate the performance of the algorithm using a confusion matrix. Applicable services for speaker recognition includes voice dialing, telephone shopping, database accesses, voice mail and many more. [1]

II. METHODS

A. APPROACH

To solve the problem, we decided to approach it by extracting the features from N number of training audio signals (from the database), followed by creating and training N number Gaussian Mixture Models, one for each extracted features, for the audio signals in order

to create a unique voice print for each of these audio signals. After creation of the GMM models, we validated the data by checking various voice prints against the trained voice print.

B. MFCC or MEL-FREQUENCY CEPSTRAL COEFFICIENTS

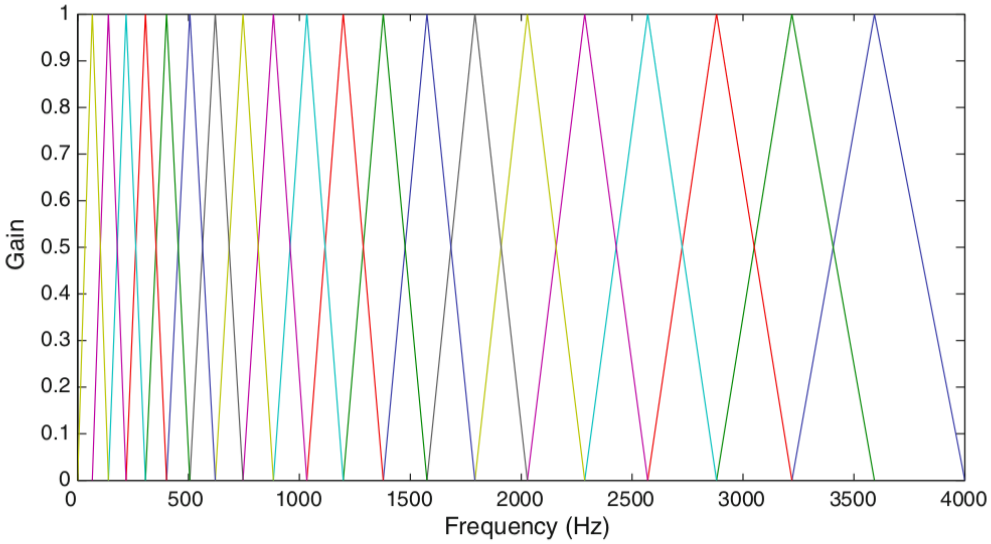

Figure 1 *: Mel Filter Bank*

The figure 1, above, shows the filer bank used for the Mel frequencies. The MFCC feature extraction technique basically includes windowing the signal, applying the DFT, taking the log of the magnitude, and then warping the frequencies on a Mel scale, followed by applying the inverse DCT. The description of the various steps is given below:

i. Pre-emphasis:

In this initial phase, the higher frequencies are filtered out to emphasize. The main goal being balancing the spectrum of the voices or sounds having steep roll-off in the high-frequency region. For sounds with speech, the glottal source has an approximately −12 dB/octave slope [2]. However, when the acoustic energy radiates from the lips, the spectrum boosts by a roughly +6 dB/octave. This results in a speech signal, when recorded with a microphone, having approximately a −6 dB/octave slope downward compared to the true spectrum of the vocal tract. Therefore, some of these glottal effects from vocal tract parameters are eliminated by the pre-emphasis process. The most commonly used pre-emphasis filter is given by the following transfer function

$$H(z) = 1 - bz{-}1 \qquad \textit{(equation 1)}$$

where the value of b controls the slope of the filter and is usually between 0.4 and 1.0 [2].

ii. Frame blocking and windowing:

For steady acoustic characteristics, this signal needs to be inspected over an adequate short period of time as generally a speech signal is a slow time-varying sound signal. Therefore, speech recognition analysis should always be carried out on shorter segments, considering the signals stationary during these shorter time-periods. Short-term spectral measurements are typically carried out over 20 ms windows, and advanced every 10ms [3, 4]. By advancing the time window every 10 ms or 20 ms, tracking the temporal characteristics of sounds from an individual speech and analysis window of these sounds is usually adequate to provide good spectral resolution of the speech, and sounds. The purpose of the overlapping analysis is that each speech sound of the input sequence would be approximately centered at some frame. On each frame, a window is applied to taper the signal towards the frame boundaries. Generally, Hanning or Hamming windows are used to enhance the harmonics, smooth the edges, and to reduce the edge effect while taking the DFT on the signal [2].

iii. DFT spectrum:

Each windowed frame is converted into a magnitude spectrum by applying DFT or Discrete Fourier Transform.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\Pi nk}{N}} \qquad where\ 0 \leq k \leq N-1 \qquad \textit{(equation 2)}$$

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

iv. Mel spectrum:

A Mel is a unit of measure based on the perceived frequency by human's ear. It differs from the physical frequency or intensity of the tone since the human auditory system can not perceive the pitch linearly. The Mel scale is approximately a linear frequency spacing below 1 kHz and a logarithmic spacing above 1 kHz [5]. The approximation of Mel from physical frequency can be expressed as :

$$f_{Mel} = 2595 log_{10}(1 + \frac{f}{700}) \qquad \textit{(equation 3)}$$

where *f* denotes the physical frequency in Hz, and *fMel* denotes the perceived frequency [3].

The Mel spectrum is computed by passing the Fourier transformed signal through a set of band-pass filters known as Mel filter bank (shown in Figure 1 above). Filter banks can be implemented in both time domain and frequency domain. However, the filter banks used to compute the MFCC are usually implemented in the frequency domain. The center frequencies of the filters are evenly spaced on the frequency axis.

However, in order to mimic the human ear's perception, the warped axis, according to the nonlinear function given in Eq. (3), is implemented. The most commonly used filter shaper is triangular, and in some cases the Hanning filter can be used [2]. The triangular filter banks with Mel frequency warping are given in Fig. 1.

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

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

where M is the total number of triangular Mel weighting filters [6, 7]. $Hm(k)$ is the weight given to the *kth* energy spectrum bin contributing to the *mth* output band and is expressed as:

$$H_m(k) = 0 \quad when\ k < f(m-1) \qquad \textit{(equation 5)}$$

$$H_m(k) = \frac{2(k-f(m-1))}{f(m)-f(m-1)} \quad when\ f(m-1) \le k \le f(m) \qquad \textit{(equation 5)}$$

$$H_m(k) = \frac{2(f(m+1)-k)}{f(m+1)-f(m)} \quad when \quad f(m) \le k \le f(m+1) \qquad \textit{(equation 5)}$$

$$H_m(k) = 0 \quad when\ k > f(m+1) \qquad \textit{(equation 5)}$$

with m ranging from 0 to M-1.

v. Discrete cosine transform (DCT):

The DCT is applied to the transformed Mel frequency coefficients to produce a set of cepstral coefficients. Prior to computing DCT, the Mel spectrum is usually represented on a log scale. This results in a signal in the cepstral domain with a frequency peak which corresponds to the pitch of the signal and a number of formants representing low frequency peaks. Since most of the signal information is represented by the first few MFCC coefficients, the system can be made robust by extracting only those ones and ignoring or truncating rest of DCT components [2].

Finally, MFCC is calculated as [2]

$$c(n) = \sum_{m=0}^{M-1} log_{10}((s(m))cos(\frac{\Pi n(m-0.5)}{M}) \quad where\ n = 0, 1, 2.... C-1$$

(equation 6)

where $c(n)$ are the cepstral coefficients, and C is the number of MFCCs. The zeroth coefficient is often excluded since it represents the average log-energy of the input signal and therefore, carries very little speaker-specific information.

vi. Dynamic MFCC features:

Since the cepstral coefficients only contain information from a given frame, they are usually referred to as static features. The extra information about the temporal dynamics of the signal is computed from the first and second derivatives of cepstral coefficients [8–10]. The first-order derivative is referred to as delta coefficients, and the second-order derivative is referred to as delta–delta coefficients. The speech rate and the information about the acceleration of the speech is obtained by the delta coefficient and delta-delta coefficients respectively. The commonly used definition for computing dynamic parameter is [8]

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

where $cm(n)$ denotes the *mth* feature for the *nth* time frame, *ki* is the *ith* weight, and T is the number of successive frames used for computation. The delta–delta coefficients are computed by taking the first-order derivative of the delta coefficients.

C. DELTA FEATURES

The idea behind using delta (differential) and delta-delta (acceleration) coefficients is that in order to recognize speech better is to find the trajectory of the MFCC coefficients over time. The delta coefficients are computed using the following formula.[11]

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \qquad \textit{(equation 8)}$$

where dt is a delta coefficient from frame t computed in terms of the static coefficients ct−n to ct+n. n is usually taken to be 2. The acceleration coefficients are computed similarly, but using the differential instead of the static coefficients.

D. GAUSSIAN MIXTURE MODEL

Normally, linear regression and sub-space models are based on reducing the dimensions of the data in order to capture a single essential information from the data or signal which is not enough in our case, as generally a pre-recorded voiced sound usually contains both speech and noise in them, which would cause a different range of dimensions for a range of possible signals. Therefore, a better model match would be *statistical distribution* of the signal, by using simple Gaussian processes.

However, the Gaussian Processes generally ignores multivariate signals. And since a speech from two different individuals differ greatly in more than one dimension such as speech rate, frequencies, or intensity of the sound, a Gaussian process would be insufficient as it would ignore a large number of structures from the sound signal, rendering the model inefficient. [12]

Therefore, the Gaussian Mixture Model is the best model to process a signal with a variety of classes and structures, for example, voiced and unvoiced sounds as well as speech from two different individuals with different structures. The Gaussian Mixture Model is generally represented as

$$f(x) = \sum_{k=1}^{K} \alpha_k f\left(x; \Sigma_k, \mu_k\right), \qquad \textit{(equation 9)}$$

where αk adds up to unity or 1.

III. RESULTS

At first, we converted an audio signal to an amplitude-time chart, therefore, revealing the audio signals in time-domain and the change in the amplitude of the data over time. This was followed by transformation of the audio signal into frequency-domain.

Frequency-Domain Representation of an audio signal helps us provide the detail about the different frequencies present in the signal. In order to get the audio signals into the frequency-domain, several steps were performed. Applying Fourier Transform, which is a mathematical concept that can be used to convert a continuous signal from time-domain to frequency domain. We used numpy's FFT algorithm to apply 1-D discrete Fourier Transformation of the signal shown below in Fig. 2.

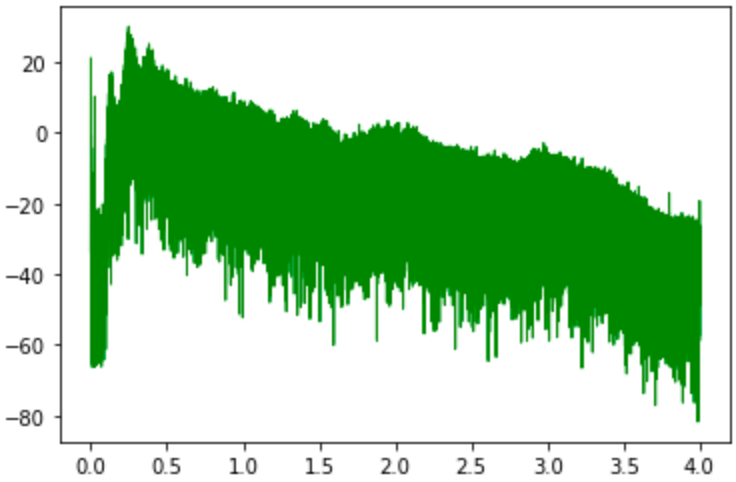

***Figure 2**: Frequency-Domain Audio Signal*

After getting the audio into frequency-domain, the desired features can be extracted. We first extracted MFCC (Mel Frequency Cepstral Coefficients) from the signal using mfcc from Python's speech_recognition library. MFCC uses Discrete Cosine Transform on the audio signal and allows us to extract features which are demonstrated in Fig. 3.

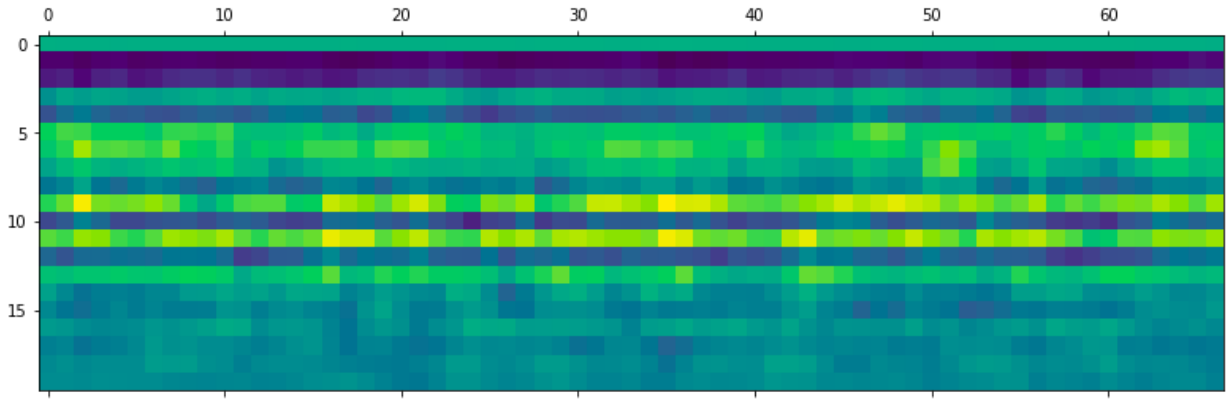

***Figure 3:** MFCC Features of Speech Signal*

After extracting the features, we trained our Gaussian Mixture Model using the training data and then validated it against testing data. The testing and training data were created by recording or taking excerpts from several youtube content creators as it would provide us with a diverse group of speakers. A special note was taken to minimize the unnecessary noise, created from music, in the youtube videos.

Several of these samples were used to train a Gaussian Model, leaving out two at a time. These two samples are considered as probe samples, and used to cross check the validation results of these models.

At first, after creation of Gaussian Models for samples 1-5, *the code has been pasted in the VI. Appendix*, the testing data chose 3 matching samples with 2 probe samples. The expected result was 60% accuracy with a misclass of 40%. Upon examining the confusion matrix formed by this input, we observed that the accuracy and the misclass of the confusion matrix, shown in Fig 4, were 60% and 40% respectively. Thus validating our expected output.

// confusion matrix 1

Similarly, a testing sample of 1-5 classes was taken, *the code snippets are pasted in the VI. Appendix,* since all the testing files are already in the training database, we expect an accuracy of 100%, with 0% of misclass. The confusion matrix, Fig. 5, again confirms that the results from the Gaussian Mixture Models accurately represent the expected output.

//confusion matrix 2

IV. CONCLUSION

GMM classifier is currently very popular and has become a standard for text independent speaker recognition. It doesn't require any large amount of data to produce correct results and can work very efficiently on smaller amounts of data.

There were majorly two main motivations behind Gaussian Mixture Models modeling speaker identity. Firstly, the component Gaussians were shown to represent characteristic spectral shapes (vocal tract configurations) from the phonetic sounds which comprise a person's voice. Hence, by modeling the underlying acoustic classes, this speaker model has a better capability to model the short term variations of a person's voice, allowing high identification performance for shorter signals.

Our trained GMM models had hundred percent accuracy in almost all cases because our our database had very few number of audio signals for training and testing, and also some of our training data samples had very less noise so our which made our model more capable of correctly predicting the classes of data samples.

Also the improvements that can be done to make our code more efficient at predicting is :

1. Get a larger database of audio samples
2. Have some audio samples(both training and testing) with some fair amount of noise
3. Run more tests and accordingly tune the mfcc parameters for better feature extraction

V. REFERENCES

1. http://www.scholarpedia.org/article/Speaker_recognition
2. J.W.Picone,Signalmodelingtechniquesinspeechrecognition.Proc.IEEE**81**,1215–1247(1993)
3. J.R.Deller,J.H.Hansen,J.G.Proakis,*DiscreteTimeProcessingofSpeechSignals*(PrenticeHall, NJ,1993)
4. J.Benesty,M.M.Sondhi,Y.A.Huang,*HandbookofSpeechProcessing*(Springer,NewYork,2008)
5. J. Volkmann, S. Stevens, E. Newman, A scale for the measurement of the psychological magnitude pitch. J. Acoust. Soc. Am. **8**, 185–190 (1937)
6. Z.Fang,Z.Guoliang,S.Zhanjiang,Comparison of different implementations of MFCC. J.Comput. Sci. Technol. **16**, 582–589 (2000)
7. 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), pp. 191–194
8. L.Rabiner,B.-H.Juang,B.Yegnanarayana,*Fundamentals of SpeechRecognition*(PearsonEducation, London, 2008)
9. S.Furui,Comparison of speaker recognition methods using statistical features and dynamic features. IEEE Trans. Acoust. Speech Sig. Proc. **29**, 342–350 (1981)
10. J.S.Mason,X.Zhang,Velocity and acceleration features in speaker recognition,in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (1991), pp. 3673–3676
11. https://desh2608.github.io/2019-07-26-delta-feats/
12. Aalto University Wiki; Retrieved on 13 April 2022 from https://wiki.aalto.fi/pages/viewpage.action?pageId=151492301

VI. APPENDIX

A. CONTRIBUTION
    a. Punit Patel
        i. Introduction and Objectives
        ii. Methods (A, C, D)
        iii. Editing and Optimization of the Code
        iv. Results
        v. Appendix

    b. Bikramjeet Singh Atwal
        i. Methods (B)
        ii. Creation of new Data
        iii. Testing and Optimization of the Code
        iv. Conclusion

B. CODE
    a. The code is inside the IPYNB notebook which is provided in the folder.

C. DATABASE
    a. The audio samples used for training are marked with "_train.wav", while the one used for testing is marked as "_test.wav"
    b. All the audio samples are provided in the zip folder with the ipynb notebook in the zip folder.

D. HOW TO RUN THE CODE
    a. The IPYNB file attached should be opened with the Jupyter Notebook.
    b. The Data attached with the code should be inside same directory as Notebook
    c. To run the code, open the Jupyter Notebook and Navigate to the directory with ipynb notebook and the data.
    d. Open the Notebook and Press "Run All Cells". This should run the notebook.
    e. If you want to change the data under training and testing, please change the loops where the data is being read.