Speaker Recognition using Squeezed Mel Frequency Cepstral Coefficient matrix and Neural Networks

1st Ketan Vibhandik
dept. of Electrical Engineering)
Indian Institute of Technology
Tirupati, India
ee17b033@iittp.ac.in

2nd Rama Krishna Gorthi
dept. dept. of Electrical Engineering)
Indian Institute of Technology
Tirupati, India
rkg@iittp.ac.in

***Abstract*—The objective of this work is to perform text-independent speaker recognition under normal noisy and unconstrained conditions. This was done using 2 distinct datasets. The first one was the TIMIT dataset containing audio samples from 160 speakers. The other one was a self-collected data set by us, containing audio data of 51 speakers. With squeezed Mel Frequency Cepstral Coefficient (MFCC) matrix as the input features to a deep neural network and softmax layer for classification, an accuracy of above 98% on test-data was achieved for a total of 211 speakers. The deep-learning model was trained with only 12-sec of audio data per speaker.**

***Index Terms*—speaker recognition, neural-networks, audio pre-processing, Mel-frequency cepstral coefficients.**

I. Introduction

Speaker recognition is the task of recognising the speaker from his or her audio sample. There are 2 major types of speaker recognition: text-dependant and text-independent. Text-dependent speaker recognition only identifies the speaker when a particular keyword or sentence is uttered. Whereas, text-independent speaker recognition can identify the speaker irrespective of the words or text uttered.

Applications of speaker recognition can vary from authentication in high-security systems and forensic tests to searching for persons in large corpora of speech data to auto-generating scripts from the audio of a dialogue. These tasks require high accuracy under real-world conditions. Text-independent speaker recognition under noisy and unconstrained conditions is an extremely challenging task. Real-world conditions become very difficult due to intrinsic variations like emotion, age, speaker's health, etc. and extrinsic variations like background noise, microphone quality, etc.

There have been numerous attempts to perform speaker recognition using different approaches, like, by extracting low-level speech features like pitch, timbre, tone, etc, using feature engineering. In recent times, machine learning is being implemented to help improve the performance of speaker recognition. Also, attempts have been made to make an end-to-end speaker recognition systems using deep convolution network, to completely automate this task.

II. Related work

The traditional stochastic technique uses the GMM-UBM approach(Gaussian mixture model as the universal background model) [1]. It is an unsupervised clustering technique to cluster the MFCC feature vectors. It assumes each cluster to be a multidimensional Gaussian distribution. Principal component analysis (PCA) is applied on this model to get the Eigen-voice vector or the i-vector. Though it does the perform the task well enough, it requires large amount of data. Another stochastic technique is the Hidden Markov Model (HMM). Though it represents the speaker information is a good enough manner, it fails to generalize well with speaker and text variations. Thus, HMM does not perform well for text-independent speaker recognition.

Another recent technique is deep learning based speaker recognition. Time delay neural networks (TDNN) [2] along with frame sub-sampling is applied on small time-frames of the MFCC coefficient matrix. It extracts features from each time-frame. This feature-vector is passed to a softmax layer for classification. Drawback is that it requires much more computation power than linear layered neural network and has high latency.

There are also other spectrogram based techniques which could not perform as well. Recently, attempts have been made to perform an end-to-end speaker recognition from raw audio waveform using deep convolutional neural network. It uses multiple one dimensional convolutional layers to make a custom spectrogram with custom features which are best suited for the task.

III. Domain Knowledge

Humans produce sound by vibration of the vocal cords present in their voice box (larynx) present in their throat. Every person has a unique structure of vocal path which makes their voice distinct from others. Audio can be characterized by it's features like pitch, timbre, etc. Pitch is referred to the lowest harmonic of the frequency spectrum and has the highest energy amongst all harmonics. Every person has a particular and very limited natural pitch range which he or she has a control on. But, what is distinct for every speaker, is the timbre of their voice. Timbre is the pattern made by the 1st harmonic of the voice along with it's overtones. Overtones are the multiples of the lowest frequency component of the sound spectrum which make a unique pattern for every person as seen in the spectrogram in figure 1. Thats the reason why same note played on a piano and on guitar sounds different. Figure 1 shows a spectrogram of human voice.

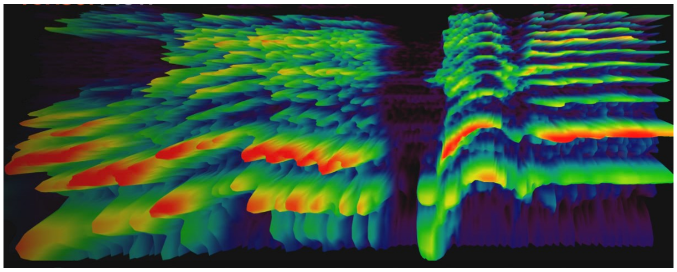


Fig. 1. Decibel-powered Mel-frequency spectrogram. The horizontal axis has time on it and the vertical axis has frequency on it. The color gradient shows the power at a particular frequency at a particular time.

We have tried to exploit this fact and capture this unique pattern in the frequency spectrum of human voice. It is perfectly done by Mel-Frequency Cepstral Coefficients. Explained in section IV.

In every language, there are voiced and unvoiced sounds [3]. Most of the energy of the frequency spectrum belongs to the voiced sounds. They form horizontal continuous lines seen in the audio spectrogram. The peaks in figure 1 are caused due to the voiced part of spectrogram. All vowels are voiced.

We have tried to exploit these facts to recognise the speaker.

IV. Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCC) are decorrelated logarithmic outputs of triangular filters of equal bandwidth and spacing in Mel-scale. Figure 2 shows the steps involved in MFCC extraction [4].

Pattern of the frequency spectrum is to be captured as explained in the section III. But, the position of this pattern shifts in frequency as time varies as we speak . It can be seen in figure 1. But, as seen in figure 1, the pattern remain roughly the same. It just shifts along the frequency axis. So as to remove this temporal changes, we divide the audio into 30 millisecond frames (with 50% overlap). Thus, the frequency pattern for most of the frames is stochastically stationary.

Power spectrum for each frame is obtained by squaring its Fast Fourier Transform (FFT). The power spectrum of each frame contains lots of redundant information. So we apply 100 triangular filters, of equal bandwidth and spacing in Mel-scale, to the power spectrum. These filters are called Mel-filters. Mel-scale (natural logarithmic scale) is used because human ear is more sensitive to frequency changes at lower frequencies than at higher frequencies. That is, we have higher frequency resolution at lower frequencies. In musical terms, we perceive note change from note A2 (110 Hz) to A2 (116Hz) and from note A6 (1760 Hz) to A6 (1864 Hz) equally as they are equally spaced in natural log scale. 100 frequency bins are obtained for each frame. For 2 seconds of audio, a 100 × 13 ($frequency \times time$) matrix is obtained. Humans also hear loudness in logarithmic scale. So natural log of this matrix is taken.

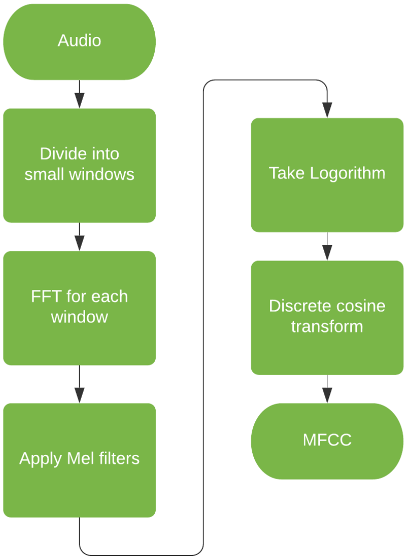


Fig. 2. MFCC extraction

Now, to capture the required frequency spectrum pattern, we take Discrete Cosine Transform (DCT) along the frequency axis. This decorrelates the frequencies. Thus 100 MFCCs are obtained for each time-frame. We use only first 50 coefficients and discard the rest to get a 50 × 13 MFCC matrix. The reason for taking 50 MFCCs is explained in section VI. This feature matrix is perfectly able to capture the timbre (patterns made by the frequency spectrum as explained in section III) of the speaker's voice. Thus it has high discriminative power for speaker voices.

V. Experimenting

A. Overview

We tried different approaches for classification. Only 12-sec of audio data per person was used for each approach. Initially k-nearest neighbours was implemented for classification. The amount of computation it requires to classify increases linearly with the number of speakers. Another approach we tried was using 2-D deep convolutional neural network (CNN) with 50 × 13 MFCC feature matrix. It was followed by linear layers to classify the high-level features extracted by the CNN. This was a quite a big model and required high computational power.

Next, we tried using linear-layered neural network model for classification. The MFCC feature matrix was squeezed along the time-axis to get a 50 dimensional feature vector; which was input to this model. It was a relatively smaller model (size-wise) model and used less computational power.

We also tried to implement end-to-end speaker recognition using deep convolution networks but couldn't achieve satisfactory results given the amount of data and computation power we had.

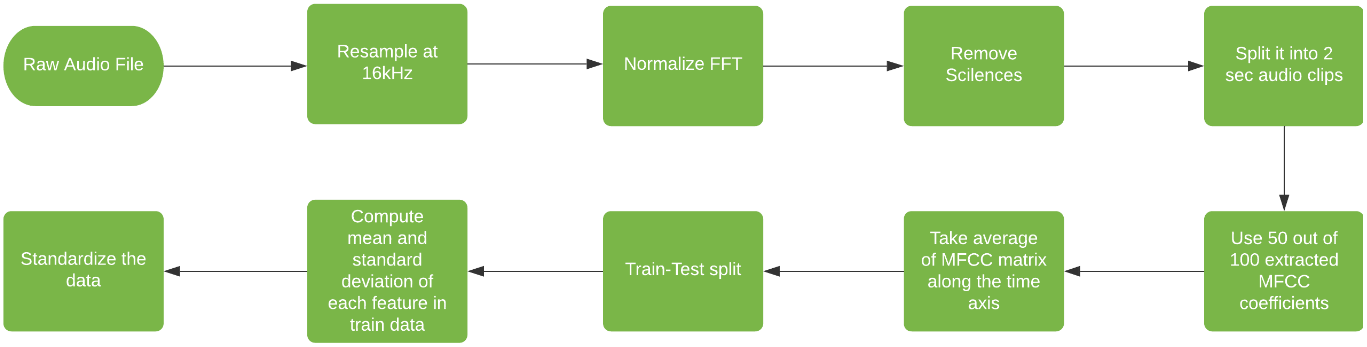

Raw Audio File
Resample at 16kHz
Normalize FFT
Remove Scilences
Split it into 2 sec audio clips
Use 50 out of 100 extracted MFCC coefficients
Take average of MFCC matrix along the time axis
Train-Test split
Compute mean and standard deviation of each feature in train data
Standardize the data

Fig. 3. Final Methodology

B. Data Set

The data set used was a part of the TIMIT dataset [5] containing a total data of 160 speakers. This data-set had audio samples of American English speakers. This data had less noise and was of a high quality.

The second data-set used was self-collected one by us. It contained audio samples of 51 speakers in 5 different Indian languages. This data set was recorded in an unconstrained, natural noisy condition with mobile microphones.

This made up to a total speaker data of 211 speakers

C. Final Methodology

After trying multiple approaches mentioned in subsection V-A, we finalized this methodology which gave best results. Figure 3 shows the steps involved.

The data-set had different sampling rates for different audio files. Therefore, the raw audio was resampled to 16kHz to have consistent amount audio content per frame. The audio data also had inconsistent loudness, and therefore, it needed to be normalized. Normalisation of the time-domain waveform did not work as in the case of an large abrupt peak due to background noise lead to suppression of speaker data. Therefore, Fast Fourier Transform (FFT) of each audio sample was normalised with respect to the maximum frequency component.

Audio waveform was split into 2-sec time-frames as it was large enough to get the speaker features. For this to work, silences in the audio couldn't be afforded as it could result in frames with no speaker voice at all. Therefore, next step was to remove silences. For this, audio samples were divided into 25 millisecond-frame with an overlap of 10 ms and RMS value of audio waveform was calculated for each frame. Frames with RMS value lower than the threshold value were considered as silences and were discarded.

After getting normalized and silence-free data, (50×13) MFCC feature matrix was computed for each 2-sec audio frames as explained in section IV. The first 50 coefficients were used out of 100 computed ones.

Timbre is the only relevant information in the audio required for speaker recognition. Any time-series information, like the words uttered and language spoken, is irrelevant. Some time-frames of the MFCC matrix are affected by background noises. Therefore, average of the MFCC feature matrix was taken along the time-axis (columns) to obtain a single 50 dimensional feature vector representing the audio. This discarded any time-series information present. Thus, speaker information was represented in a very compact manner by a single 50-dimensional vector.

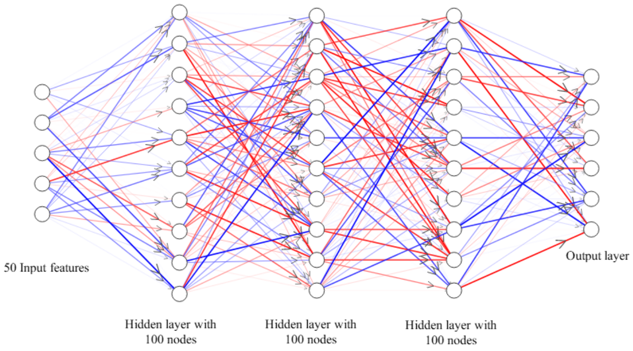

50 Input features
Hidden layer with 100 nodes
Hidden layer with 100 nodes
Hidden layer with 100 nodes
Output layer

Fig. 4. Neural Network Model

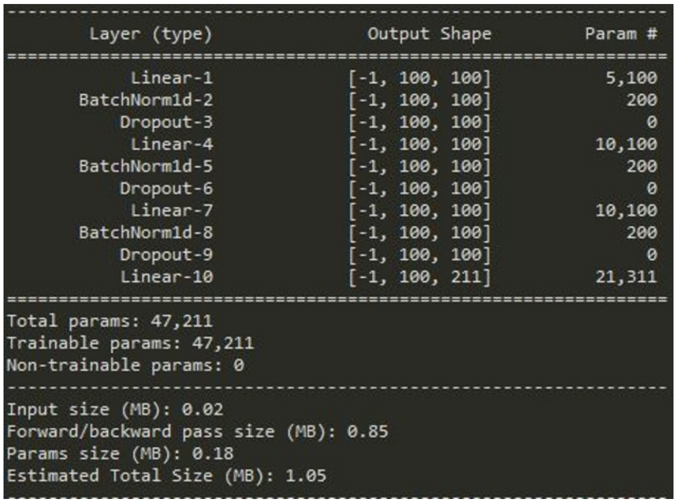
These features were then fed to a linear-layered deep-neural-network (section V-D) to obtain features with even higher discriminative properties. Softmax layer was used to classify the features as of one of the 211 speakers.

The same procedure was repeated by adding Gaussian noise to entire audio data to check is robustness against noise.

D. Model Details

The neural network model had 3 linear hidden layers with 100 nodes each and an output layer of 211 nodes corresponding to 211 speakers as shown in the figure 4. Input to each this model is a 50-dimensional feature vector. Every hidden layer had a batch normalization layer following it and had a dropout of the probability of 25%.

The input data was split into a 65-35 train-test split. Mean and standard deviation of each feature along the training data



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1             [-1, 100, 100]           5,100
       BatchNorm1d-2             [-1, 100, 100]             200
           Dropout-3             [-1, 100, 100]               0
            Linear-4             [-1, 100, 100]          10,100
       BatchNorm1d-5             [-1, 100, 100]             200
           Dropout-6             [-1, 100, 100]               0
            Linear-7             [-1, 100, 100]          10,100
       BatchNorm1d-8             [-1, 100, 100]             200
           Dropout-9             [-1, 100, 100]               0
           Linear-10             [-1, 100, 211]          21,311
================================================================
Total params: 47,211
Trainable params: 47,211
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.02
Forward/backward pass size (MB): 0.85
Params size (MB): 0.18
Estimated Total Size (MB): 1.05
----------------------------------------------------------------
```

Fig. 5. Model Details

was computed, which was used to standardize each input feature of the feature vector of train and test data, before feeding it to the model.

The model had a total of 47,211 trainable-parameters and a total size of 1.05MB (as shown in figure 5).

VI. Result

Best accuracies were achieved by linear-layered neural network as shown in Table I. Other approaches mentioned in subsection V-A did not perform as well and their performance degraded after adding noisy self-collected data and by adding Gaussian noise.

Surprisingly, linear layered model, with squeezed MFCC matrix input vector, outperformed CNN model, which has greater ability to extract features.

Accuracy of over 98%, using 12-sec of speaker data for training was achieved using linear-layer model. This was done for the total data-set of 211 speakers which also included self-collected data with unconstrained background noises. Adding Gaussian noise did not significantly affect the model's performance.

The comparison of the results achieved using different approaches is given in the Table I.

Another major observation was that by varying the number of MFCC features, the accuracies were significantly affected. Its effect on the linear layered model is shown in figure 6.

VII. Conclusion

- Accuracy greater than 98% was achieved using only 12 seconds of training-data per person.
- The final model has a very low latency. Thus it could be used in real-time applications.
- It performed well with the self collected dataset which had natural background noise. Adding Gaussian noise also did not degrade its performance. Thus, it is robust against noise.

TABLE I
Accuracies with different approaches

Method	Data set	Number of speakers	Accuracy
K-nearest neighbours	TIMIT	160	73%
Convectional neural network	TIMIT	160	90.6%
Linear layer Neural network	TIMIT	160	98.7%
Linear layer Neural network	TIMIT and self-collected	211	98.4%
Linear layer Neural network	With added noise	211	97.2%

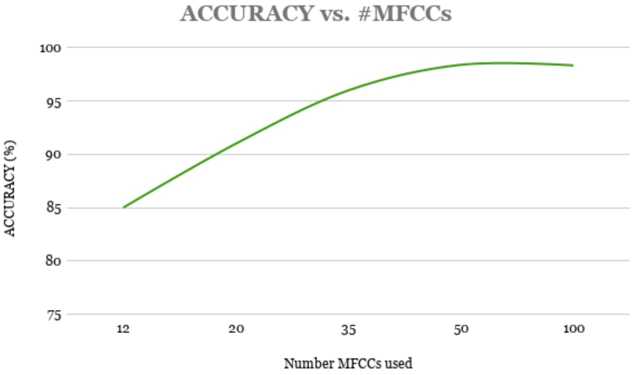


Fig. 6. Varying number of MFCCs used

- Squeezing the MFCC feature matrix along the time-axis also improved robustness to noises and speaker-voice inconsistencies. It better represented the timbre information (patterns in the frequency spectrum)
- FFT normalisation helped overcome inconsistencies in the audio intensity due to abrupt spikes in time-series data.
- Silence removal made taking time-frames as small as 2-sec per data point possible. This helped clean the data and achieve higher accuracy. Time-resolution was also increased by using smaller time frames
- Resampling the audio lead to consistant amount of data in each time frame.
- During training, batch normalisation and standardizing the input data helped in faster convergence and to achieve better accuracies Dropout helped in regularisation.

References

[1] T. Hasan and J. H. Hansen, "A study on universal background model training in speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 1890–1899, 2011.

[2] D. Snyder, D. Garcia-Romero, and D. Povey, "Time delay deep neural network-based universal background models for speaker recognition," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 92–97.

[3] L. G. A. Martinez, "Voiced and unvoiced sounds." [Online]. Available: https://en.calameo.com/read/0045223116498adbd933c

[4] J. Lyons, “Mel frequency cepstral coefficient (mfcc) tutorial.” [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[5] World BankThe World Bank, “Timit data set,” tIMIT Acoustic-Phonetic Continuous Speech Corpus, https://catalog.ldc.upenn.edu/LDC93S1.