Develop a speaker identification system

G12, Authors, Ibrahim M. Sawy, Ahmed Atwa, Ahmed Emad, Ahmed Metwally, Hady Walid.

Abstract—Speaker identification under noisy conditions is one of the challenging topics in the field of speech processing applications. Motivated by the fact that the neural responses are robust against noise, this paper proposes a new speaker identification system using 2-D neurograms constructed from the responses of a physiologically-based computational model of the auditory periphery. The responses of auditory-nerve fibers for a wide range of characteristic frequency were simulated to speech signals to construct neurograms. The neurogram coefficients were trained using the well-known Gaussian mixture model-universal background model classification technique to generate an identity model for each speaker. In this study, three text-independent and one text-dependent speaker databases were employed to test the identification performance of the proposed method. Also, the robustness of the proposed method was investigated using speech signals distorted by three types of noise such as the white Gaussian, pink, and street noises with different signal-to-noise ratios. The identification results of the proposed neural-response-based method were compared to the performances of the traditional speaker identification methods using features such as the Mel-frequency cepstral coefficients, Gamma-tone frequency cepstral coefficients and frequency domain linear prediction. Although the classification accuracy achieved by the proposed method was comparable to the performance of those traditional techniques in quiet, the new feature was found to provide lower error rates of classification under noisy environments.

I. Introduction

Speech communication from mobile devices has traditionally been made using low bit-rate speech codecs. The low bit-rates at which these codecs operate introduce a slight distortion of the speech signal which becomes more severe in noisy conditions. When input into a speech recognizer, this distortion causes a noticeable reduction in accuracy. To overcome this problem the technique of distributed speech recognition (DSR) has been proposed by the ETSI Aurora group. DSR replaces the codec on the terminal device with the feature extraction component of the speech recognizer and so removes codec-based distortion from the speech recognizer input. This results in a significant improvement in speech recognition accuracy. However, because speech feature vectors are designed to be a compact representation, optimized for discriminating between different speech sounds, they do not contain sufficient information to enable reconstruction of the original speech signal. In particular, valuable speaker information, such as pitch, is lost. However, several schemes have been proposed recently for reconstructing speech from a combination of MFCC vectors and pitch. These have been based on either a sinusoidal model or a source-filter model of speech production. An extension of this work also considered the reconstruction of clean speech from noise contaminated MFCC vectors and a robust pitch estimate. In these systems, the MFCC vectors and pitch are extracted using separate speech processors. For example, in a 128- channel auditory model provided robust estimates of the pitch. The aim of this work is to integrate the MFCC extraction and pitch estimation components into a single speech frontend. For both pitch estimation and MFCC extraction, the speech signal is decomposed into a number of discrete frequency bands either by an auditory model or Mel-filter bank. It is therefore reasonable to combine this into a single system and this is described. A detailed evaluation of the pitch extraction component is described and a comparison made with alternative pitch extraction methods. As mentioned in [1]. we will use the algorithm in fig.1

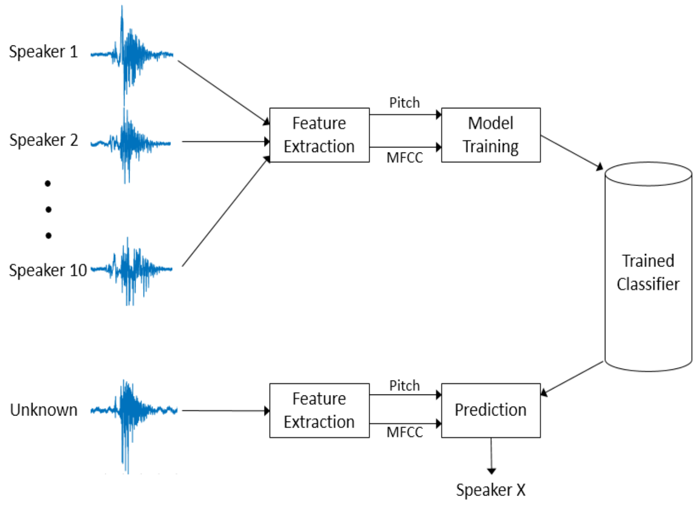


Figure 1 algorithm

II. Features Used for Classification

The output of the auditory model takes the form of a series of time-domain samples from each of the bandpass filters. In conventional MFCC feature extraction a windowing function captures a short-time frame of speech. From this a Fourier transform determines the magnitude spectrum and this is then quantized in frequency using a Mel-spaced filter-bank. To generate a filter-bank vector from the time-domain filter outputs of the auditory model a mean amplitude (MA) filter is employed. This output is the root mean square amplitude, ck, from each bandpass filter, k, at 10ms intervals from a 25ms buffer of time-domain samples, where

$$c_k = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}[x_k(n)]^2}$$

As $x_k(n)$ is the nth time-domain sample from the kth bandpass filter in the 25ms buffer, N is the buffer length (N=200 samples for the 8kHz sampling frequency). This is consistent with the frame width and frame rate used in the Aurora MFCC standard. The final three stages are logarithm, discrete cosine transform and truncation. These are identical to the last three stages in conventional MFCC extraction. It should be noted that the positioning of the auditory filters is close to, but not exactly, Mel-scaled. Therefore, the features extracted by this system are not strictly MFCCs. However, for the purpose of this work they are referred to as auditory model-based MFCCs.

A. Method of Pitch Evaluation

Speech can be broadly categorized as *voiced* and *unvoiced*. In the case of voiced speech, air from the lungs is modulated by vocal cords and results in a quasi-periodic excitation. The resulting sound is dominated by a relatively low-frequency oscillation, referred to as *pitch*. In the case of unvoiced speech, air from the lungs passes through a constriction in the vocal tract and becomes a turbulent, noise-like excitation. In the source-filter model of speech, the excitation is referred to as the source, and the vocal tract is referred to as the filter. Characterizing the source is an important part of characterizing the speech system.

As an example of voiced and unvoiced speech, consider a time-domain representation of the word 'two' (/T UW/). The consonant /T/ (unvoiced speech) looks like noise, while the vowel /UW/ (voiced speech) is characterized by a strong fundamental frequency.

The simplest method to distinguish between voiced and unvoiced speech is to analyze the zero-crossing rate. A large number of zero crossings implies that there is no dominant low-frequency oscillation.

Once you isolate a region of voiced speech, you can characterize it by estimating the pitch. This example uses `pitch` to estimate the pitch. It uses the default normalized autocorrelation approach to calculating pitch.

Apply pitch detection to the word 'two' to see how pitch changes over time. This is known as the *pitch contour*, and is characteristic to a speaker.

By plot the pitch function output over the time, we get the following figure:

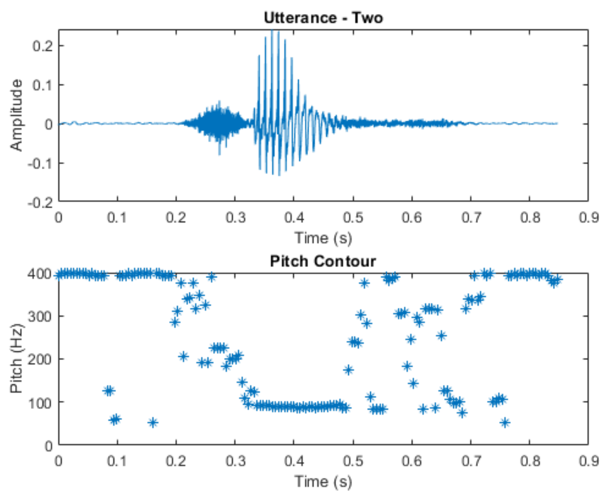


Figure 2 Pitch contour

B. Voiced vs. unvoiced signal

The `pitch` function estimates a pitch value for every frame. However, pitch is only characteristic of a source in regions of voiced speech. The simplest method to distinguish between silence and speech is to analyze the short-term power. If the power in a frame is above a given threshold, you declare the frame as speech.
For voiced consonants, the vocal cords are engaged, making sound. For unvoiced consonants, the vocal cords are not making sound, there is just air passing through them.

In other words, a voiced sound is a strong sound in which the vocal cords vibrate. Unvoiced or voiceless sounds are weak and the vocal cords do not vibrate.

The simplest method to distinguish between voiced and unvoiced speech is to analyze the zero-crossing rate. A large number of zero crossings implies that there is no dominant low-frequency oscillation. If the zero-crossing rate for a frame is below a given threshold, you declare it as voiced.

The shown figure 3 is the description of the algorithm.

After applying this algorithm in the same audio above we get the following result, note that we recoded our data does not have so much zero cross on it, so the following result may not differ from the above:

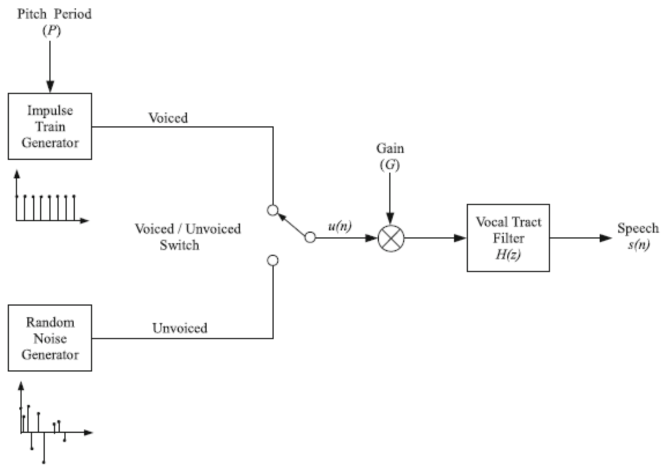


Figure 3 Model of speech production for LP analysis

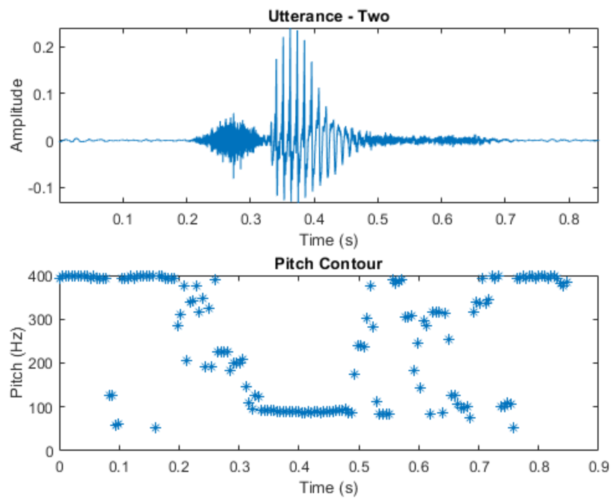


Figure 4 Pitch Contour after removing the zero crossing

C. Method of MFCC Features

The MFCC feature extraction technique 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 detailed description of various steps involved in the MFCC feature extraction is explained below.

D. Pre-emphasis

Pre-emphasis refers to filtering that emphasizes the higher frequencies. Its purpose is to balance the spectrum of voiced sounds that have a steep roll-off in the high frequency region. For voiced sounds, the glottal source has an approximately $-12\ dB/octave$ slope. However, when the acoustic energy radiates from the lips, this causes a roughly +6 dB/octave boost to the spectrum. As a result, a speech signal when recorded with a microphone from a distance has approximately a −6 dB/octave slope downward compared to the true spectrum of the vocal tract. Therefore, pre-emphasis removes some of the glottal effects from the vocal tract parameters. The most commonly used pre-emphasis filter is given by the following transfer function:

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

Where the value of b controls the slope of the filter and is usually between 0.4 and 1.0.

E. Frame blocking and windowing

The speech signal is a slowly time-varying or quasi-stationary signal. For stable acoustic characteristics, speech needs to be examined over a sufficiently short period of time. Therefore, speech analysis must always be carried out on short segments across which the speech signal is assumed stationary. Short-term spectral measurements are typically carried out over $30\,ms$ windows, and advanced every $5\,ms$ (Collect the samples into frames of $30\,ms$ with an overlap of 83.3%.). Advancing the time window every $5\,ms$ enables the temporal characteristics of individual speech sounds to be tracked and the $30\,ms$ analysis window is usually sufficient to provide good spectral resolution of these sounds, and at the same time short enough to resolve significant temporal characteristics. 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. This is done to enhance the harmonics, smooth the edges and to reduce the edge effect while taking the DFT on the signal.

F. DFT spectrum

Each windowed frame is converted into magnitude spectrum by applying DFT.

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

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

G. Mel-spectrum

Mel-Spectrum is computed by passing the Fourier transformed signal through a set of band-pass filters known as Mel-filter bank. A Mel is a unit of measure based on the human ears perceived frequency. It does not correspond linearly to the physical frequency of the tone, as the human auditory system apparently does not perceive pitch linearly. The Mel scale is approximately a linear frequency spacing below 1 kHz, and a logarithmic spacing above 1 kHz. The approximation of Mel from physical frequency can be expressed as

$$f_{mel} = 2{,}595 \log_{10}(1 + \frac{f}{700})$$

Where f denotes the physical frequency in Hz, and f_{mel} denotes the perceived frequency.

Filter banks can be implemented in both time domain and frequency domain. For MFCC computation, filter banks are generally implemented in frequency domain. The center frequencies of the filters are normally evenly spaced on the frequency axis. However, in order to mimic the human ears perception, the warped axis according to the non-linear function given in previous equation. The most commonly used filter shaper is triangular, and in some cases, the Hanning filter can be found. The triangular filter banks with Mel frequency warping is given in figure 6.

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

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

Where M is total number of triangular Mel weighting filters. $H_m(k)$ is the weight given to the k^{th} energy spectrum bin contributing to the m^{th} output band and is expressed as:

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \dfrac{2\left(k - f(m-1)\right)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{2(f(m+1) - k)}{f(m-1) - f(m)} & f(m-1) \le k \le f(m) \\ 0, & k > f(m+1) \end{cases}$$

With m ranging from 0 to $M - 1$.

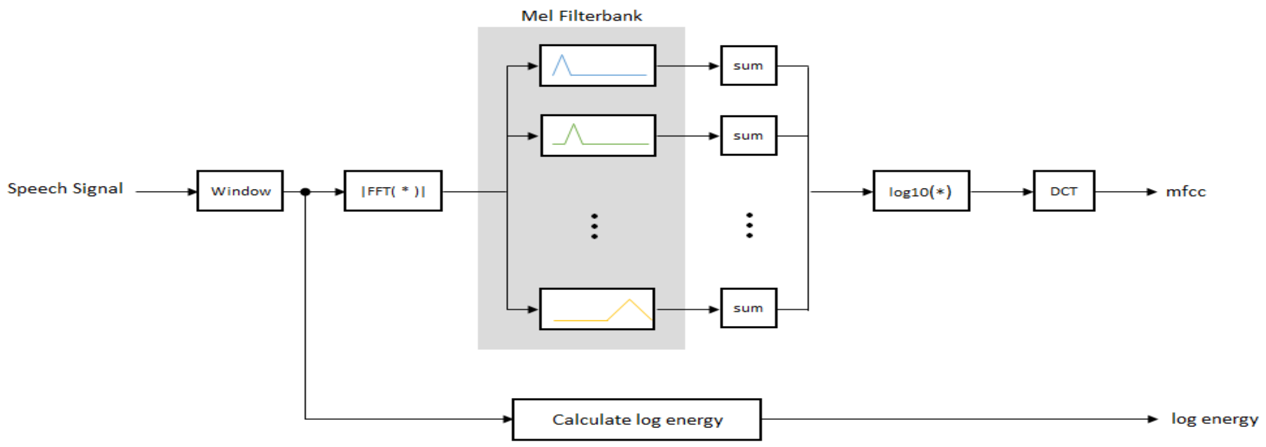


Figure 5 Mel-filter banks block diagram

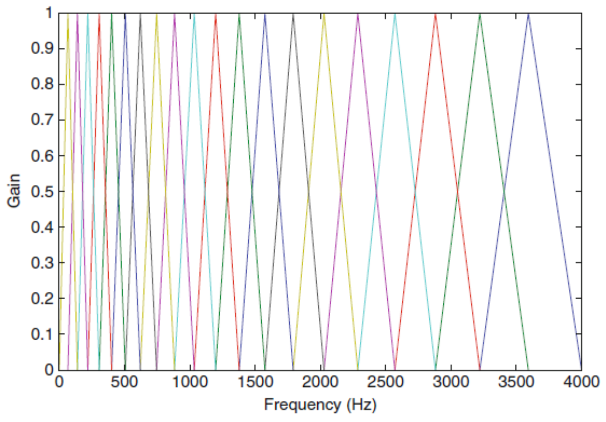


Figure 6 Mel-filter Banks

H. Discrete Cosine Transform (DCT)

Since the vocal tract is smooth, the energy levels in adjacent bands tend to be correlated. The DCT is applied to the transformed Mel frequency coefficients produces 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 que-frequency peak corresponding to the pitch of the signal and a number of formants representing low que-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 coefficients ignoring or truncating higher order DCT components. Finally, MFCC is calculated as:

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

Where $c(n)$the cepstral coefficients and C is is the number of MFCCs. Traditional MFCC systems use only 8–13 cepstral coefficients. The zeroth coefficient is often excluded since it represents the average log-energy of the input signal, which only carries little speaker-specific information.

I. Dynamic MFCC features

The cepstral coefficients are usually referred to as static features, since they only contain information from a given frame. The extra information about the temporal dynamics of the signal is obtained by computing first and second derivatives of cepstral coefficients. The first order derivative is called delta coefficients, and the second order derivative is called delta-delta coefficients. Delta coefficients tell about the speech rate, and delta-delta coefficients provide information similar to acceleration of speech. The commonly used definition for computing dynamic parameter is:

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

Where $c_m(n)$ denotes the mth 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. Generally, T is taken as 2. The delta-delta coefficients are computed by taking the first order derivative of the delta coefficients.

II. Dataset

This model uses recorded and preprocessed data that we collected and cleansed manually, the data is the 10 Arabic digits recorded one time each by 10 individuals with a total of 100 wav files, then the data is processed by removing the noise and the empty parts of it using prat, then we applied 5 different augmentation techniques to achieve Sound Augmentation, while increasing the data to be used by 500 more examples, We applied the modifications with a sole purpose of avoiding overfitting.

The benefits of data augmentation are two:

- The first is the ability to generate 'more data' from limited data.
- The second one is to avoid overfitting: For a network it is somewhat problematic to memorize a larger amount of data, as it is very important to avoid overfitting. This occurs because the model memorizes the full dataset instead of only learning the main concepts underlying the problem. To summarize, if our model is overfitting, it will not know how to generalize and, therefore, will be less efficient.

The augmentation techniques were as follows:

- Change pitch and speed.
- Value augmentation.
- Add distribution noise.
- Apply Harmonic Percussive Source Separation.
- Stretching.

In the next figure 7 an example of the augmentation techniques, we followed.

III. Training the Classifier

We trained a classifier based on the 100 records we collected earlier. we used a K-nearest neighbor (KNN) classifier. KNN is a classification technique naturally suited for multiclass classification. The hyperparameters for the nearest neighbor classifier include the number of nearest neighbors, the distance metric used to compute distance to the neighbors, and the weight of the distance metric. The hyperparameters are selected to optimize validation accuracy and performance on the test set. We set the number of neighbors to 5 and the metric for distance chosen is squared-inverse weighted Euclidean distance.

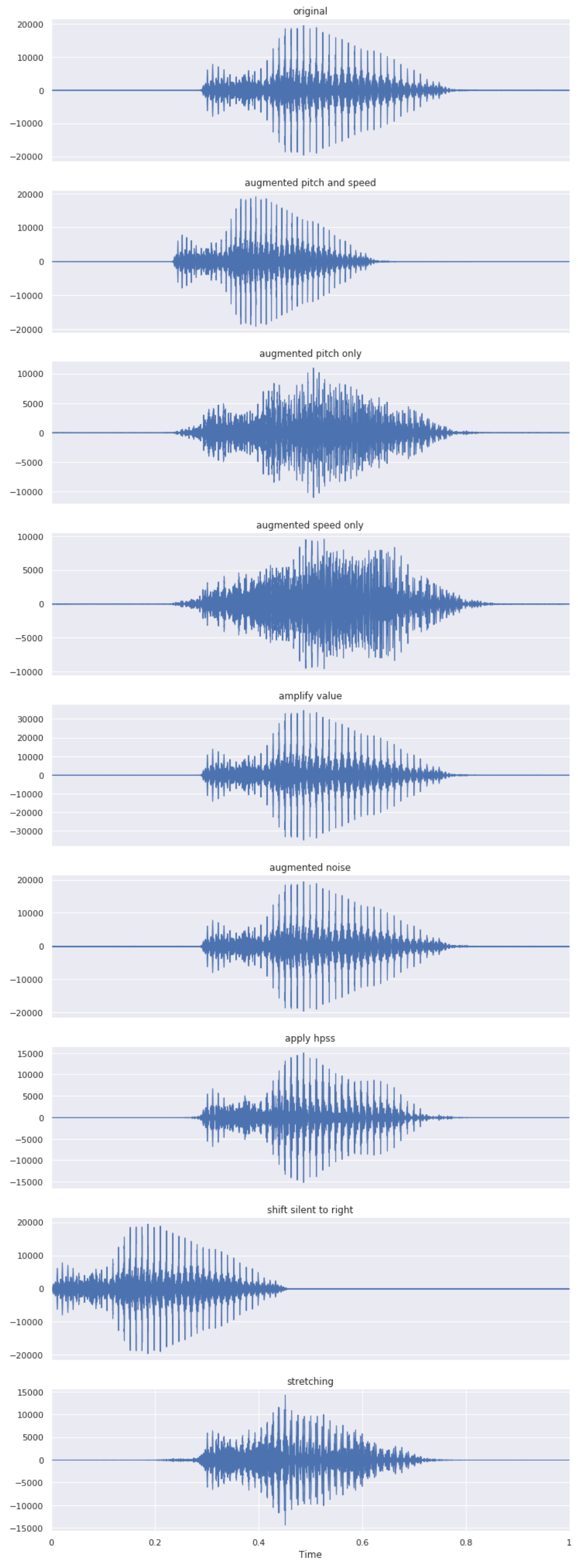


Figure 7 Waves resulted from python scrip

We used `fitcknn` function from MATLAB that is Fit k-nearest neighbor classifier; we trained the classifier and printed the cross-validation accuracy. *crossval* and `kfoldLoss` are used to compute the cross-validation accuracy for the KNN classifier. And we achieved 97.53% validation accuracy.

Also, we trained another model on all the 600 records we generated, we used a python-based algorithm, through LSTM Neural Network we managed to achieve high validation accuracy of 96.88%.

IV. Evolution

The internet gave rise to new ways of using data. Using this, we can communicate directly or indirectly with machines by training them, which is known as Machine Learning. Before this, we have to access a computer to communicate with machines.

Research and development are beginning to eliminate some of the use of computers to a great extent. We know this technology as Automatic Speech Recognition. Based on Natural Language Processing (NLP), it allows us to interact with machines using our natural language in which we speak.

The initial research in the field of Speech Recognition has been successful. Since then, speech scientists and engineers aim to optimize the speech recognition engines correctly. The ultimate goal is to optimize the machine's interaction according to the situations so that error rates can be reduced and efficiency can be increased.

Some organizations have already started the development of fine-tuning speech recognition technologies. For more than a decade, Virginia based GoVivace Inc. has continually specialized in the design and development of speech recognition technologies and solutions.

Voice recognition, whether used for authentication or identification, will become increasingly crucial not only in the expansion of use-cases for speech-driven command applications, but also in providing more self-service and self-certification opportunities for customers and consumers. As reflected in the Visa survey, it will be as effective as a tool for customer retention and acquisition as it will be for cost reduction and fraud prevention.

As shown in fig.8 the MATLAB results are as shown, as mentioned before we achieve accuracy of 97.53%.

we get validation accuracy from each class according to MFCC feature extraction and also from pitch which is the fundamental frequency of each person and the k-means classify his voice we used k=5 (five neighbors).

And as shown in fig.9 the accuracy matrix of the test set, we use the same classifier to predict the labels for the test set which is 20% of the full data.

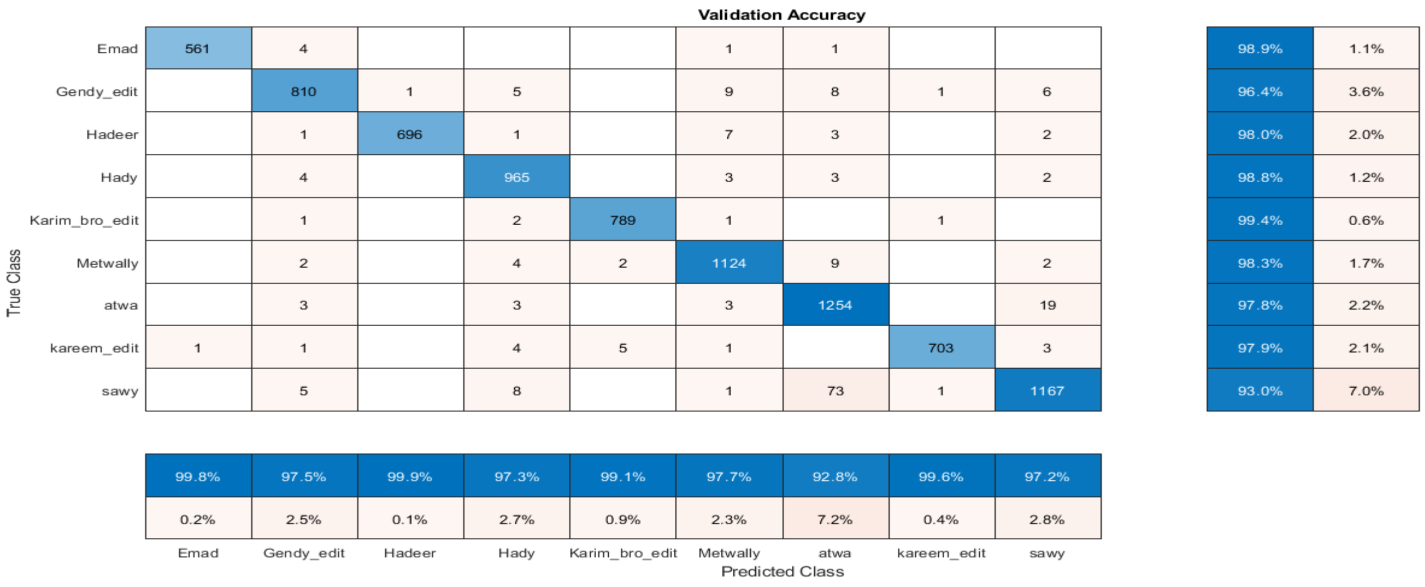
Also as shown in fig.8, the evaluation accuracy of the LSTM model using Python.

V. CONCLUSION

We build a small Developed speaker identification system by first, extract features from a voice signal like the pitch and MFCC from the records then machine learning models using both MATLAB and python codes.

In the MATLAB code we use dataset consist of 100 voice recorded and we use K-nearest neighbor classifier to train the model and we get validation accuracy of 97.53% and test accuracy around 73%.

In the python code we use data set consist of 600 voice records by augmenting the data used above (100 voice-recorded), and we use a LSTM model to train and we get validation accuracy of 95%. Following we are showing the resulting images from both MATLAB and Python codes.

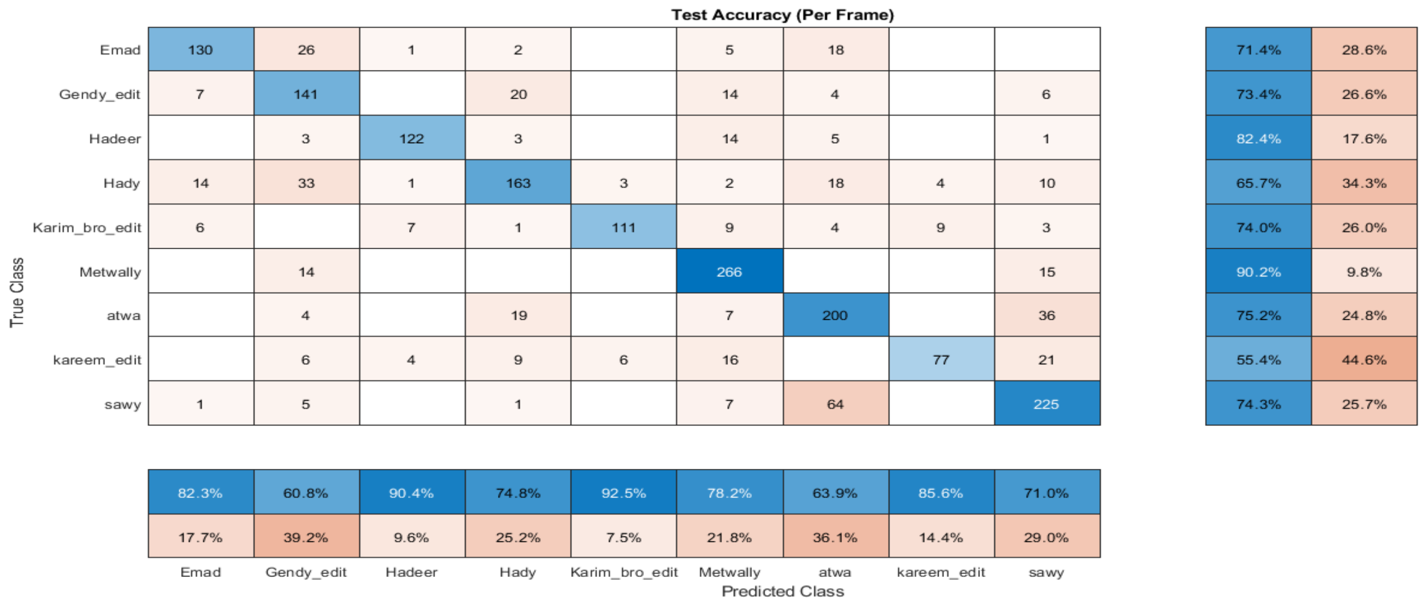


Validation Accuracy

True Class	Emad	Gendy_edit	Hadeer	Hady	Karim_bro_edit	Metwally	atwa	kareem_edit	sawy		
Emad	561	4				1	1			98.9%	1.1%
Gendy_edit		810	1	5		9	8	1	6	96.4%	3.6%
Hadeer		1	696	1		7	3		2	98.0%	2.0%
Hady		4		965		3	3		2	98.8%	1.2%
Karim_bro_edit		1		2	789	1		1		99.4%	0.6%
Metwally		2		4	2	1124	9		2	98.3%	1.7%
atwa		3		3		3	1254		19	97.8%	2.2%
kareem_edit	1	1		4	5	1		703	3	97.9%	2.1%
sawy		5		8		1	73	1	1167	93.0%	7.0%
	99.8%	97.5%	99.9%	97.3%	99.1%	97.7%	92.8%	99.6%	97.2%		
	0.2%	2.5%	0.1%	2.7%	0.9%	2.3%	7.2%	0.4%	2.8%		

Predicted Class

Figure 8 validation accuracy

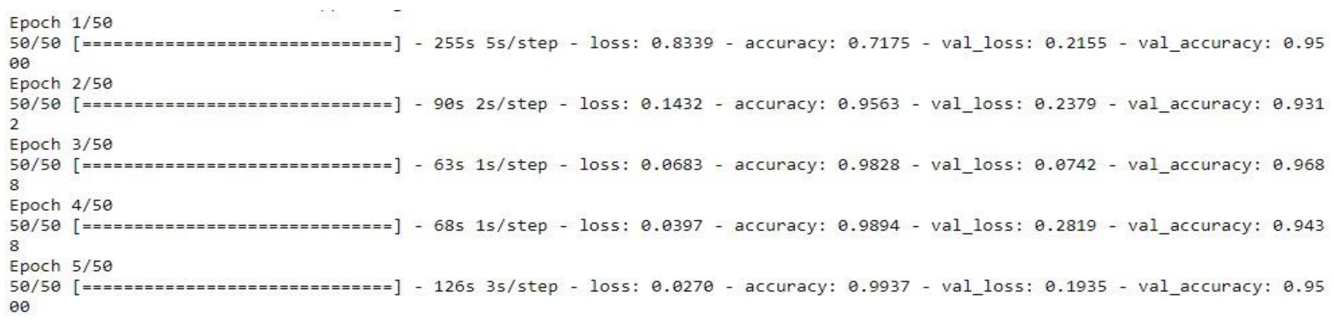


Test Accuracy (Per Frame)

True Class	Emad	Gendy_edit	Hadeer	Hady	Karim_bro_edit	Metwally	atwa	kareem_edit	sawy		
Emad	130	26	1	2		5	18			71.4%	28.6%
Gendy_edit	7	141		20		14	4		6	73.4%	26.6%
Hadeer		3	122	3		14	5		1	82.4%	17.6%
Hady	14	33	1	163	3	2	18	4	10	65.7%	34.3%
Karim_bro_edit	6		7	1	111	9	4	9	3	74.0%	26.0%
Metwally		14				266			15	90.2%	9.8%
atwa		4		19		7	200		36	75.2%	24.8%
kareem_edit		6	4	9	6	16		77	21	55.4%	44.6%
sawy	1	5		1		7	64		225	74.3%	25.7%
	82.3%	60.8%	90.4%	74.8%	92.5%	78.2%	63.9%	85.6%	71.0%		
	17.7%	39.2%	9.6%	25.2%	7.5%	21.8%	36.1%	14.4%	29.0%		

Predicted Class

Figure 9 Test accuracy



```
Epoch 1/50
50/50 [==============================] - 255s 5s/step - loss: 0.8339 - accuracy: 0.7175 - val_loss: 0.2155 - val_accuracy: 0.95
00
Epoch 2/50
50/50 [==============================] - 90s 2s/step - loss: 0.1432 - accuracy: 0.9563 - val_loss: 0.2379 - val_accuracy: 0.931
2
Epoch 3/50
50/50 [==============================] - 63s 1s/step - loss: 0.0683 - accuracy: 0.9828 - val_loss: 0.0742 - val_accuracy: 0.968
8
Epoch 4/50
50/50 [==============================] - 68s 1s/step - loss: 0.0397 - accuracy: 0.9894 - val_loss: 0.2819 - val_accuracy: 0.943
8
Epoch 5/50
50/50 [==============================] - 126s 3s/step - loss: 0.0270 - accuracy: 0.9937 - val_loss: 0.1935 - val_accuracy: 0.95
00
```

Figure 10 validation accuracy using LSTM model using python

VI. References

[1] https://www.mathworks.com/help/audio/ug/speaker-identification-using-pitch-and-mfcc.html

[2] https://awardsclever387.weebly.com/Speaker Identification Using Pitch and MFCC

[3] Integrated pitch and MFCC extraction for speech reconstruction and speech recognition applications

[4] A Robust Speaker Identification System Using the Responses from a Model of the Auditory Periphery

[5] Spoken-Digit-Recognizer/dataset for python trial code

[6] Wen-Ya-Lin, Spoken-Digit Recognizer

[7] wenya-chungyuan-jauhhsiang/Spoken-Digit-Recognizer

[8] Indra den Bakker - Python Deep Learning Cookbook-Packt (2017)

[9] L. R. Rabiner and B. H. Juang, Fundamentals of Speech Recognition. Englewood Cliffs, NewJersy: Prentice-Hall, 1993.

[10] G. Seshadri and B. Yegnanarayana, “Perceived loudness of speech based on the characteristics of glottal excitation source,” Journal of Acoustic Society of America, vol. 126, p. 2061–2071, October 2009.

[11] ESTI document - ES 201 108 – STQ: DSR – Front-end feature extraction algorithm; compression algorithm, 2000.

[12] R.D. Patterson et al, SVOS Final Report: The Auditory Filterbank, APU Report 2341, 1988.

[13] L.R. Rabiner and R.W. Schafer, “Digital Processing of Speech Signals”, Prentice-Hall, 1978.

[14] O. M. Mubarak, E. Ambikairajah, and J. Epps, “Analysis of an mfcc-based audio indexing system for efficient coding of multimedia sources,” in The 8th International Symposium on Signal Processing and its Applications, (Sydney, Australia), 28–31 August 2005.