Speaker Recognition using K-Nearest Neighbors and MFCC Features

Hamza Zamani1, Taishi Kato1, David Rosenwasser1

1University of California, Los Angeles

hwzamani@g.ucla.edu, taishikato10@g.ucla.edu, dbrosenwasser@gmail.com

Abstract

This paper provides an introduction to the challenges of speaker recognition and offers an approach to form a text-independent speaker verification model. Described herein is an approach implementing mel-frequency cepstral coefficients (MFCC) as the main element of the feature vector to a K-Nearest Neighbors (KNN) algorithm in order to develop a speaker recognition model. Two sets of training data were trained against three different cases. The first set of training data set consisted of various speakers reading the same sentence (text-dependent) while the second set consisted of participants speaking in casual speech (text-independent). By obtaining a high identification rate in all cases, the model would provide evidence to being a considerably robust speaker recognition system.

**Index Terms**: speech recognition, mel-frequency cepstral coefficients, neural networks, knn, feature extraction

# Introduction

This section provides an overview of the speaker recognition goals and challenges. The ability to determine if a voice pattern is identical to another is useful for various purposes including authentication and surveillance. Identification based on voice patterns can be used for password protection or creating vocal signature. Vocal signatures can be used to lock or unlocked access to areas of interest. Alternatively, a known voice pattern can be filtered through a large amount of speech data to target a unique person of interest for various purposes, like recovering missing persons or targeting military foes [1].

## Speaker Recognition Challenges

A difficult challenge presented by the constraints of this project involves the fact that the model must fit both text-dependent and text-independent cases. The text-dependent set consists of various female speakers reading the exact same sentence (read) while the text-independent set involves those same speakers participating in unique casual speech (phone). As the results will show, the recognition accuracy is much higher when the model is trained and tested on the same set and lower during mismatch cases.

The dataset presented is not a closed set, meaning that some speakers in the testing phase may not have been part of the training phase. This may dilute the classification accuracy and contribute to a lower Equal Error Rate (EER). All of the data samples are within three to five seconds in length, which provides a limited amount of data that can be used for the model to make a determination. As detailed later in this paper with the Neural Network implementation, it is difficult to provide a model that is robust and accurate in both cases without a closed data set. Further, Neural Networks require a large amount of data to produce accurate results and the constraints of this project compared only one speaker against another. This constraint, in addition to other factors discussed later in the paper, contributed to the KNN outperforming the Neural Network classification.

# Methodology

This section describes the methodology forming the speaker identification system. Computations were performed using MATLAB and functions provided from the VOICEBOX speech processing toolbox [2]. The speaker recognition model will be given two speech segments and must determine if the samples are spoken by the same speaker or not. The general block diagram of the speech recognition system described by this paper is demonstrated in Figure 1.



Figure 1: *Block Diagram of Speech Recognition System*

Each training sample will enter a pre-emphasis filter before feature extraction. The feature vector consists mainly of Mel-Frequency Cepstral Coefficients which were then fed into the K-Nearest Neighbors (KNN) machine learning implementation to produce a model that can predict test data. The testing sample follows the sample procedure, and is predicted by the model to be the same speaker or not. The scoring was calculated using an Equal Error Rate (EER) which is the percentage of error when the threshold of your scoring function is set such that the False Positive Rate (FPR) is equal to the False Negative Rate (FNR).

## Pre-emphasis

During the pre-emphasis phase, a filter with numerator coefficients 1 and -.95 is applied to each audio sample. The pre-emphasis filter attempts to attenuate the lower frequencies and boost the high frequencies in an attempt to normalize the signal. Through various trial runs of the data, the pre-emphasis filter was determined to be beneficial in achieving a higher accuracy rate.

Adding the pre-emphasis filter produced the improved results as evidenced by Table 1. Filtering in this way reduces the noise of the audio sample and produced more refined results.

Table 1: *Adding Pre-emphasis Filter*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Read | Phone | Mismatch |
| Train Read | +2.22% | +0% | +.74% |
| Train Phone | -.16% | +2.98% | +.10% |

## MFCC Feature Extraction

This section describes the technique and reasoning behind extracting the Mel-Frequency Cepstral Coefficients (MFCCs) from each audio sample to form the feature vector used as the input to the classifier. The linear time invariant (LTI) model of speech production presented below in Figure 2.

Source

Transfer Function

Radiation

Sound

Figure 2: *LTI Model of Speech Production*

The cepstral domain is extraordinarily valuable for speech and speaker recognition due to its ability to separate the source, transfer function, and radiation effects of the speech signal being produced. We are interested in capturing the transfer function information in the MFCC because the transfer function contains a representation of the vocal tract and the vocal tract is representative of the uniqueness of the speaker [3][4]. In addition, the use of MFCCs has become firmly established as an excellent feature vector for speech and speaker recognition problems. Thus, this paper will use MFCCs to obtain a representation of the transfer function of the speech signal under analysis to use for speaker recognition.

Equation (1) describes the equation for obtaining the MFCC.

The MFCC approach is a frequency analysis based on a filter bank with approximately critical band spacing of the filters and bandwidths. The MFCC is modeled after the human auditory system which contains proportionally large and narrow filters at the lower end of the frequency scale than the higher end [5]. Because of this, the unit of the MFCC is the mel, which is a warped frequency representation of MFCC. Warping the frequency scale into mel scale allows for improved resolution at lower frequencies which contains the unique characteristics of speech that will be used as the feature vector.

### Feature Extraction

To obtain the MFCC features, the samples must first be divided in overlapping segments as shown in Figure 3 in order to perform the sampling of the signal. The segment under analysis, or the frame, is multiplied by a Hamming window in the time domain before the Fourier transform is taken. As shown in equation (2), the length of the window is related to the sampling frequency, where *fs* = 22050Hz for each audio sample. Each frame contains a 50% overlap with the previous sample. 50% is necessary for the Hamming window as the window shape is tapered at the ends, unlike a rectangular filter.

(2)

Hansen and Hasan offer an efficient method to obtain the MFCC which is used during this project [6]. The logarithm of the Fourier spectrum of the windowed sample is computed before the mel-scale filter bank analysis is performed. The logarithm of the Fourier transform of a signal transforms the representation to the cepstral domain which, as described previously, offers valuable properties for speaker verification because of the transfer function. The filter bank analysis produces the cepstral energy in each channel representing different frequency bands. The highest and lowest filters were tapered down to zero because the ends of the sampling phase produce higher error rates as they overlap with a zero value [5].

Twelve MFCCs were extracted from each frame to partially create the frame vector. It is common practice to take thirteen or twenty MFCCs, yet the 0th coefficient does not convey information relevant to the overall shape of the spectrum so for our purposes it was discarded [6]. One property of the cepstral coefficients is that the values approach zero rapidly, therefore no more than twelve coefficients were needed to produce meaningful values [7]. Various test runs yielded nearly identical EER for twelve and twenty MFCCs and using fewer coefficients produced faster runtimes.

The mean across the frames were computed to form a single vector MFCC representation of the entire signal under test as described in Figure 3. The standard deviation was calculated across each frame of the signal’s twelve MFCC coefficients and concatenated with the feature to form a 24 element feature vector per audio sample. The addition of the standard deviation improved the EER as the feature vector was represented in a more dynamic way. Table 2 demonstrates the improvement after adding standard deviation to the feature vector.

Table 2: *Adding Standard Deviation to the Feature Vector*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Read | Phone | Mismatch |
| Train Read | +3.15% | +5% | +5.73% |
| Train Phone | +9.05% | +3.98% | +3.70% |

The standard deviation is a measure of the extent of the deviation of the MFCC coefficients as a whole, and logically this can contribute to the speaker recognition model as one speaker may have more variations in their speech than another speaker. The variations in speech become a useful parameter for the model to classify between speakers.

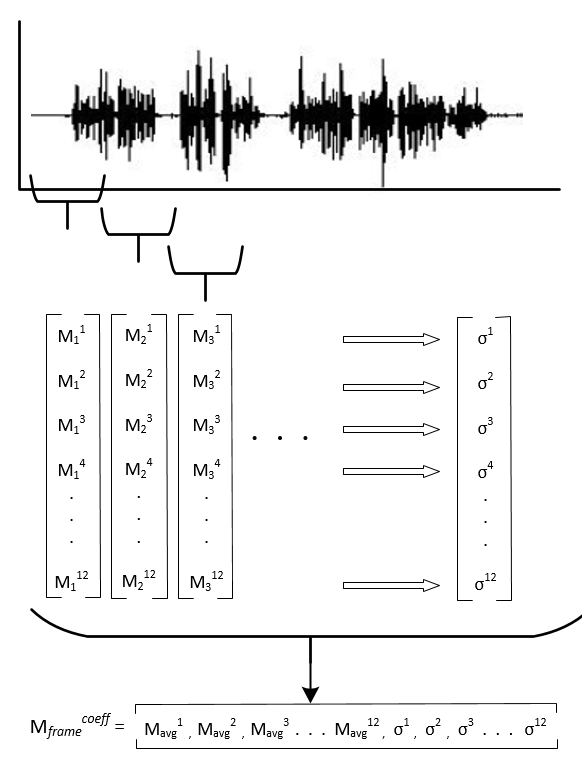


Figure 3: *Feature Extraction of MFCC*

### Feature Extraction Discussion

The initial baseline of the project implemented using the pitch of the speaker as the main element of the feature vector, yet this approach yielded minimal success. The data set was entirely consistent of female speakers, thus the pitch does not possess as much variability as it would between males and females and produced a worse than 50% accuracy rate. The average pitch was concatenated with the MFCCs to provide additional features for the machine learning approach to compare against, yet pitch was not proven to be useful for speaker recognition. The calculation of the pitch performed on every frame of every sample was computationally expensive, heavily impacted runtimes, and produced minimal improvements in ERR so it was struck from the feature vector. Removing the pitch from the feature vector reduced the runtime from an average of 1516.22 seconds to 30.25 seconds.

During trial runs, the delta and delta deltas of the MFCC were implemented to increase the feature length and to provide the model with more data for comparison. The delta and delta delta MFCCs provide a dynamic representation of the signal as it passes through the frames. Since speech is inherently a dynamic signal changing regularly in time, it is reasonable to seek a representation that includes some aspect of the dynamic nature of speech [5]. The delta and delta delta MFCCs track formants which are valuable for a speech recognition problem but did not provide a use for the speaker recognition problem that this paper presents. Table 3 shows the reduced performance adding delta delta MFCCs to the feature vector. Note that adding delta MFCCs to the feature showed nearly identical results.

Table 3: *Adding delta delta MFCCs*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Read | Phone | Mismatch |
| Train Read | -2.22% | +0% | +.97% |
| Train Phone | -4.45% | -.76% | +.06% |

## K-Nearest Neighbors

The KNN classifier was proven to outperform the Neural Network classifier discussed in 3.1 and was used for the speaker verification. Figure 4 describes the KNN classification process.

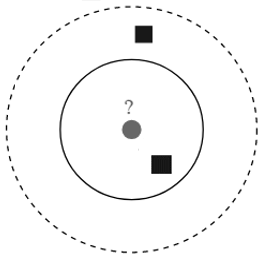


Figure 4: *Typical KNN*

The shapes in the figure above are representations of the testing data (circle) and training data (squares) being mapped across the KNN classifier’s feature space. The ring and dotted ring are representations of different scanning distances that the KNN uses to determine if the test data is a member of the trained data’s class or not. In other words, if the speaker under test is within the distance specified to the trained speaker, the KNN will determine the speaker to be the same and vice versa.

To reduce the dimensions of the features, the two samples are differenced from one another, effectively halving the dimensions or inputs to the classifier. This simplification of the feature space is commonly used to improve the classifiers ability to separate classes because the differences are more apparent. The KNN classifier was optimized in terms of neighbors and the distance method used. The ‘OptimizeHyperparameters’ parameter in MATLAB was chosen to be implemented. This parameter runs a large number of iterations to determine the most appropriate scanning distance and distance method to use. The Figure below illustrates the EER rate as the number of neighbors changes in value.

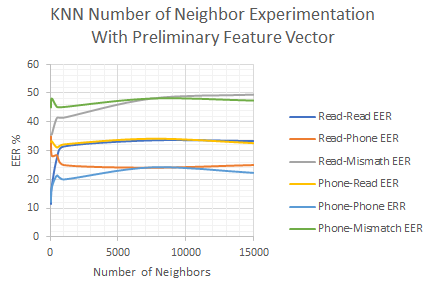


Figure 5: *KNN Optimization*

Through using OptimizeHyperparameters and an empirical analysis of the EER results, the standard distance used to measure the speaker under test to the other speaker models was calculated using a standard Euclidean geometry while the number of neighbors (rings) was optimized to seventy-five. The final results of the methodology presented in this paper are shown below in Table 4. The runtime of the implementation described in this paper was performed at an average of 30.24 seconds. Conclusions are discussed in Section 4.

Table 4: *Final Results (Old Dataset)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Read | Phone | Mismatch |
| Old Dataset | **Train Read** | 8.37% | 20.46% | 36.96% |
| **Train Phone** | 28.88% | 10.64% | 42.22% |
| New Dataset | **Train Read** | 24.55% | 20% | 36.21% |
| **Train Phone** | 28.45% | 20% | 38.51% |

Typically, by lowering the number of neighbors, the speaker recognition accuracy improved. However, if the parameter was too low, the data would be overfit and provide inaccurate results for a different set of data as we discovered.

# Discussion and Alternate Approaches

This paper presented an approach to implement MFCCs as the main element of the feature vector to a K-Nearest Neighbors (KNN) algorithm in order to develop a speaker recognition model. While this paper presented numerous alternate approaches, this group chose to keep the feature vector and classifier relatively simply to produce fast run times and accurate results. Each audio sample was represented by a 24 length feature vector consisting of twelve MFCC coefficients and the standard deviation of those coefficients across the frames. The feature vector was fed into a KNN implementation in order to make a determination if the speaker under test is identical to the speaker trained. The methodology presented in this paper was developed against the initial dataset given at the start of this project.

During the development of the speech recognition model presented herein, various approaches were explored that were not implemented in the final submittal. The following sections will describe and discuss alternate approaches that were researched to obtain the optimum speaker recognition model and ideas for future work.

## Neural Networks

Recent proliferation of neural networks in both academia and industry has made usage of the method widely available and simple to implement for research purposes. The popularity of the neural network has helped discover various applications of these algorithms in interesting and diverse fields, including speech and speaker recognition.

Modeled after the human neural system, neural networks are capable of classifying data in high dimensional features spaces. Rather than statistical pattern matching, neural networks invoke supervised learning to develop complex functions to model behavior and provide universal function approximation [8].

### Neural Network Architecture

Though implementation of a neural network is relatively straightforward, there is no method for determining *a priori*, or the optimal architecture, without human guidance. Parameters such as the number of hidden layers, neurons per layer, and more, are all subject to user tuning and are highly dependent on the nature of the input data. These parameters in addition to the algorithm settling at local minima can make optimization difficult and time consuming [9]. Figure 6 below describes a basic model of a Neural Network.



Figure 6: *Binary classification neural network with a feature vector of four parameters with a single hidden layer.*

For the training sets discussed in this project, a network consisting of three hidden layers with thirty neurons each produced the best EER results out of the number configurations tested. Figure 7 below shows a representation of the implemented neural network.



Figure 7: *Block diagram representation of implemented network*

The selection of the input layer size was selected based on the feature vector chosen for testing. As described earlier, the feature consisted of twelve MFCC coefficients per audio sample concatenated with the standard deviation across the sampled frames. Next, the hidden layers consisted of three 30 neuron layers until finally a single neuron output layer was configured for the binary classification of the speaker data. A single neuron output layer was chosen over a two neuron layer due to computation efficiency and insignificant changes in results.

### Training

Implementation of a neural network for speech recognition has been researched extensively but many involve classification of text-dependent speech or text-independent speech with a closed speaker set, some achieving 100% classification accuracy [10][11]. Other techniques include using a convolutional neural net to classify the spectrogram of a text-dependent closed set utterance [12]. The unique challenge for this project is the combination of text-dependent and text-independent sample sets coupled with an open speaker set for the final classification.

An additional constraint implemented during Neural Network training is the ratio of excitatory classes in comparison of the overall training set. Table 5 shows the ratio of training vectors that have zero labels in comparison to vectors containing one label.

Table 5: *Ratio of excitatory targets in read and phone training sets*

|  |  |  |  |
| --- | --- | --- | --- |
| Training | Vector Size | Excitatory | % Excitatory |
| READ | 9730 | 147 | 1.51 % |
| PHONE | 11175 | 150 | 1.34 % |

The nature of the training set allows for the neural network to classify all vectors as non-match speakers thereby artificially achieving a classification accuracy of 98.49% and 98.66% for read and phone training sets respectively given the training set. In reality, neural nets perform relatively poor given any excitatory test conditions. The issue of obtaining enough training data of all class types is an added difficulty of correctly training the classifier. Though not implemented for this design, there are methods to balance the training set which are further detailed in the next section. The confusion matrices for the net trained with read data and phone data are shown in Figure 8.

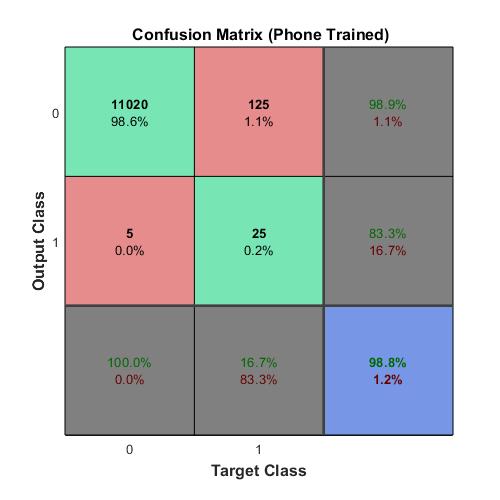
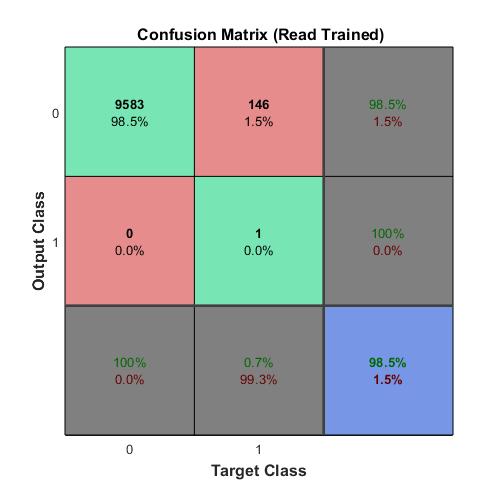


Figure 8: *Confusion Matrices*

It can be seen by the figures presented above that due to the lack of balance in classes; both confusion matrices deceptively report exceptionally high performance. This miscalculation is proven by simply inputting a feature vector that corresponds to a matching speaker set. Notice for both outputs the misclassified values almost exactly match the number of matching classes presented in the read and phone training sets.

To mitigate the issue of overrepresented classes, the training set was artificially balanced by pruning the overrepresented class, i.e. speech instances where the speaker is not the same, while keeping all the set where the speaker is the same [13]. First all excitatory classes were kept, next the same number of non-excitatory classes were randomly selected from the training set to produce a pruned training set balance 1:1. Results using pruned and un-pruned data are presented in the follow section.

### Neural Network Results

After implementation and tuning of the network, the results achieved were subpar in comparison to the KNN implementation and were therefore disregarded for the submittal. The calculated EER of the Neural Network model is shown below in Table 6.

Table 6: *Neural Network EER*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Read | Phone | Mismatch |
| Un-pruned set | Train Read | 30.98% | 29.94% | 46.67% |
| Train Phone | 21.67% | 26.67% | 47.59% |
| Pruned set | Train  Read | 37.07% | 31.59% | 49.63% |
| Train  Phone | 51.22% | 53.33% | 45.71% |

Moreover, since the network must be trained in order for it to classify the run times of the design as a whole, the implementation negatively impacts the metrics used for selection of a neural net for the finalized approach. As the feature set grows, the network takes longer to converge. Run times for both training sets are displayed in Table 7 below. The fast run times suggested by the table provide evidence the neural network is not training properly to the data presented.

In the case of using a pruned data set, the results overall were interestingly worse when the neural net was trained with a balanced set. This could be due to the fact that there were simply not enough training cases for the network to properly develop a function to represent the classification.

Table 7: *Mean Script Runtime for Neural Network*

|  |  |  |
| --- | --- | --- |
|  | Read | Phone |
| Mean Runtime (sec) | 16.54 | 18.01 |

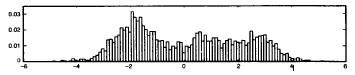
Additional challenges were encountered simply due to the nature of the training sets and structure. The project guidelines specified two separate training tasks and the neural network architecture that performed well with the read training set would not necessarily perform well with the phone training set and vice versa. This forces the architecture to be mediocre for both training sets as opposed to being optimized for one.

As mentioned in the previous section there are methods to balance the training data to include a more even distribution of classes to allow for proper training of the neural network. Another possible method would be to artificially inject noise, or slightly perturb existing sets of data (with matching speakers) and add those to the training set, or more simple use mean shifting MFCC values

Though the KNN implementation outperformed the neural network given the project conditions, with an enhanced trainset and further tuning it is plausible that the neural network could outperform the KNN especially if enhanced high-dimensional features are extracted from the speech data.

## Gaussian Mixture Models (GMMs)

Gaussian Mixture Models form speaker models to produce a speaker identification system. GMMs can be thought of like an expansion of KNN, where the GMM speaker model for the trained set is compared against the GMM speaker model of the testing set to make determinations on the identification of the speaker. Reynolds and Rose [14] provide an extremely effective speaker text-independent speaker identification system using GMMs that this project’s personnel tried to recreate with GMM functions in MATLAB’s speech processing toolbox VOICEBOX. This paper does not present any results because we were unable to extract any classification data. As described earlier, each sample was represented in a one by twenty-four length feature vector. Yet when fitting the 10 Gaussians around the feature vector histogram as shown in Figure 9, the output resulted in a 10 by 24 length feature vector per sample. Figure 9 demonstrates a GMM with ten component densities being fit around a feature histogram.



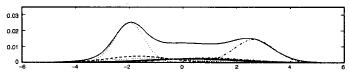


Figure 9: *Gaussian Mixture Model*

We could not figure out how to reduce the dimensions to produce reasonable results, so we shifted focus to the Neural Network approach discussed in Section 3.1.

## Linear Predictive Coding (LPC)

Following the presentation, this group decided to experiment with additional elements for the feature vector. Linear Predictive Coding is a widely used across disciplines to detect the peaks in signals. In speech processing, the peaks of the short time Fourier transform (STFT) spectrum correspond to the formant frequencies represented in voiced speech [5]. The formant frequencies correspond to the poles in the transfer function H(z) shown below.

(3)

The poles, or location of the formant frequencies, are affected by the vocal tract structure of the speaker. By using the LPC coefficients as additional elements to the feature vector, we should be able to detect the uniqueness of a speaker and form a better speaker recognition model. Figure 10 shows the linear predictive spectrum plotted across the STFT, where the peaks correspond to the poles in the transfer function.

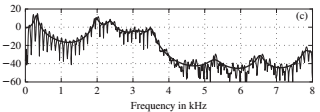


Figure 10: *Linear Predictive Spectrum*

After experimenting with extraction of the ak’s and various formants, we decided to use the average f1, f2, f3 and their standard deviations across each frame as additional elements to the feature vector. As with MFCCs, the standard deviation is valuable as it represents the variations in a person’s speech. Table 8 below demonstrates the feature vector with additional LPC features. The average runtime of adding the LPC features was 50s, compared to 30s without LPC features.

Table 8: *Adding Avg F1, F2, F3 and F1, F2, F3 StdDevs*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Read | Phone | Mismatch |
| Train Read | 26.66% | 21.66% | 34.81% |
| Train Phone | 26.66% | 21.66% | 39.02% |

For future research, this group is interested in experimenting with Linear Predictive Cepstral Coefficients (LPCCs) which represents the LPC spectrum in the cepstral domain. Like the MFCCs, LPCCs capture the transfer function of the LTI model of speech production, which contains useful properties for speaker recognition.

# Conclusions

This paper has demonstrated implementation of MFCC coefficients and the standard deviation across frames to provide a relatively accurate speaker recognition system with fast runtimes. While this paper presented numerous alternate approaches, this group chose to keep the feature vector and classifier relatively simple for the final submittal. The most difficult classification aspect of this project involved the mismatch case which still remains a challenge. While the initial results displayed in Table 4 show excellent improvement from the baseline dataset, using the new dataset produced much lower results. We attribute this to overfitting of the data. Generally as the KNN number of neighbors decreased, the classification accuracy improved, but at a certain point we seemed to have overfit the data.

After the presentation, our group decided to add formant frequencies and their standard deviations to the feature vector. This addition improved EER results slightly, which makes sense because the formant frequencies are representations of vocal tract structure, and vocal tract structure is representative of the uniqueness of the speaker. By having more data providing speaker uniqueness, our speaker recognition model expectedly performs better.

# Acknowledgements

The project members would like to thank UCLA faculty and staff for their guidance and Mathworks for MATLAB and Voicebox for their tool suite and digital speech processing functions.

# References

1. N. Singh, R.A. Khan, R. Shree, “Applications of Speaker Recognition,” *Procedia Engineering*, vol. 38, pp. 3122-3126, 2012.
2. Brookes, Mike. *VOICEBOX: Speech Processing Toolbox for MATLAB*, Imperial College Department of Electrical & Electronic Engineering,
3. S.J. Park, C. Sigouin, and J. Kreiman, “Speaker Identity and Voice Quality: Modeling Human Responses and Automatic Speaker Recognition,” *INTERSPEECH 2016, September 8-12, San Francisco, USA,* 2016.
4. C. Y. Espy-Wilson, S. Manocha, S. Vishnubhotla, “A New Set of Features for Text-Independent Speaker Identification,” *INTERSPEECH 2006 – ICSLP*, 2006.
5. Rabiner, Lawrence R., and Ronald W. Schafer. *Theory and Applications Digital Speech Processing*. Pearson, Prentice-Hall, 2011.
6. J.H.L. Hansen, T. Hasan, “Speaker Recognition by Machines and Humans, A Tutorial Review,” *IEEE Signal Processing Magazine, November 2015,* 2015.
7. Alwan, A. “Lecture 13 EE M214A.” Powerpoint, UCLA, Los Angeles, (2019).
8. Hornik, Kurt et al. “Multilayer feedforward networks are universal approximators.” *Neural Networks* 2 (1989): 359-366.
9. D. Ringach and Y. Baram, "A Diffusion Mechanism for Obstacle Detection from Size-Change Information" in *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 14, no. 01, pp. 76-80, 1994.
10. E. Variani, X. Lei, E. McDermott, I. L. Moreno and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, pp. 4052-4056.
11. Zhenhao Ge, Ananth N. Iyer, Srinath Cheluvaraja, Ram Sundaram: “Neural Network Based Speaker Classification and Verification Systems with Enhanced Features”, 2017; [http://arxiv.org/abs/1702.02289 arXiv:1702.02289].
12. Salehghaffari, H., “Speaker Verification using Convolutional Neural Networks”, 2018; [http://arxiv.org/abs/1803.05427 arXiv:1803.05427].
13. K. R. Farrell, R. J. Mammone and K. T. Assaleh, "Speaker recognition using neural networks and conventional classifiers," in *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, pp. 194-205, Jan. 1994.
14. Reynolds, D. A. and Rose, R. C. (1995). “Robust text-independent speaker identification using Gaussian mixture speaker models.” IEEE Transactions on Speech and Audio Processing, Vol. 3, No. 1, pp. 72-83.