Neural Network Based Speaker Classification and Verification Systems with Enhanced Features

Zhenhao Ge, Ananth N. Iyer, Srinath Cheluvaraja, Ram Sundaram, Aravind Ganapathiraju

Interactive Intelligence Inc., Indianapolis, Indiana, USA

Email: {roger.ge, ananth.iyer, srinath.cheluvaraja, ram.sundaram, aravind.ganapathiraju}@inin.com

Abstract—This work presents a novel framework based on feed-forward neural network for text-independent speaker classification and verification, two related systems of speaker recognition. With optimized features and model training, it achieves 100% classification rate in classification and less than 6% Equal Error Rate (ERR), using merely about 1 second and 5 seconds of data respectively. Features with stricter Voice Active Detection (VAD) than the regular one for speech recognition ensure extracting stronger voiced portion for speaker recognition, speaker-level mean and variance normalization helps to eliminate the discrepancy between samples from the same speaker. Both are proven to improve the system performance. In building the neural network speaker classifier, the network structure parameters are optimized with grid search and dynamically reduced regularization parameters are used to avoid training terminated in local minimum. It enables the training goes further with lower cost. In speaker verification, performance is improved with prediction score normalization, which rewards the speaker identity indices with distinct peaks and penalizes the weak ones with high scores but more competitors, and speaker-specific thresholding, which significantly reduces ERR in the ROC curve. TIMIT corpus with 8K sampling rate is used here. First 200 male speakers are used to train and test the classification performance. The testing files of them are used as in-domain registered speakers, while data from the remaining 126 male speakers are used as out-of-domain speakers, i.e. imposters in speaker verification.

Keywords—Neural Network, Speaker Classification, Speaker Verification, Feature Engineering

I. INTRODUCTION

Speaker recognition is a popular and broad topic in speech research over decades. It includes speaker detection, i.e. detecting if there is a speaker in the audio, speaker identification, i.e. identifying whose voice it is, speaker verification or authentication, i.e. verifying someone's voice. If the speaker set is closed, i.e. the audio must be from one of the enrolled speakers, then speaker identification is simplified to speaker classification. There are some other building blocks such speaker segmentation, clustering and diarization, which can be further developed based on the fundamental speaker recognition techniques.

Fig. 1 provides digrams for speaker identificatio and verification. the main approaches in this area includes 1) template matching such as nearest neighbor [1] and vector quantization [2], 2) neural network, such as time delay neural network [3], decision tree [4], and 3) probabilistic models, such as Gaussian Mixture Model (GMM) with Universal Background Model (UBM) [5], joint factor analysis [6], i-vector [7], [8], Support Vector Machine (SVM) [9], etc. Methods can be divided into text-dependent and text-independent, where the former achieves better performance with additional information, and the latter is more user friendly and easier to use. Reynolds

[10] and Fauve [11] provided a good overview of some common speech recognition applications with the state-of-the-art performance.
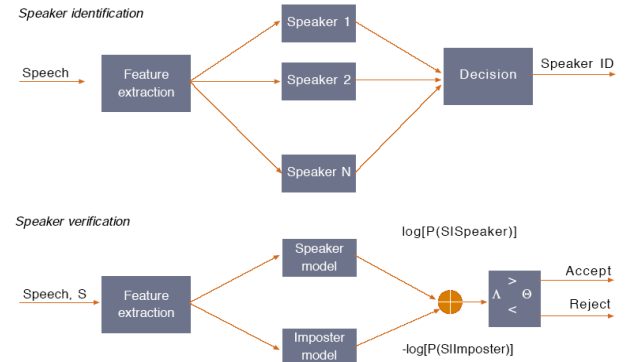

Fig. 1. Major components for speaker identification and speaker verification.

This paper proposes a neural network framework for text-independent speaker classification and verification, using TIMIT 8K database. With optimization in feature and model training, the system achieves 100% classification accuracy with slightly more than 1 second speech, and less than 6% ERR in speaker verification with more than 100 impostor size, using approximately 5 seconds data.

The following sections walk through the major pieces of this work, including feature engineering (Sec. II), design, implementation and results for speaker classification and verification systems (Sec. III and Sec. IV). Finally, the conclusion and future work is given in Sec. V.

II. DATA PREPARATION AND FEATURE ENGINEERING

The following 3 subsections introduce the database used in this paper, and the process of converting raw speech into features used that used in speaker classification and verification, including a) preprocessing, and b) feature extraction, normalization and concatenation.

A. Database

Speech of all 326 male speakers from 8 different dialect regions in the "train" folder of the TIMIT corpus with 8K sampling rate is used here. Data of males from the "test" folder and data of females from both "train" and "test" folders are currently reserved for future development. For each speaker, there are 10 data files containing one sentence each with duration about 2.5 seconds. They are from 3 categories: "SX" (5 sentences), "SI" (3 sentences) and "SA" (2 sentences). Data

are first sorted alphabetically by speaker name in their dialect region folders, then combined to form a list of data containing 326 speakers. They are then divided into 2 groups: first 200 speakers (group A) and remaining 126 speakers (group B). For speaker classification "SX" sentences in group A are used to train the text-independent Neural Network Speaker Classifier (NNSC), while the "SA" and "SI" sentences in group A are used to test. For speaker verification, since it is based on NNSC, only "SA" and "SI" sentences are used to avoid overlapping with any training data used in model training. Speakers in group A are used as in-domain speakers, and speakers in group B are used as out-of-domain speakers (imposters).

B. Preprocessing

Preprocessing mainly consists of a) scaling the maximum of absolute amplitude to 1, and b) Voice Activity Detection (VAD) to eliminate the unvoiced part of speech. Experiments show both speaker classification and verification can perform significantly better if speakers are evaluated only using voiced speech, especially when the data is noisy.

An improved version of Giannakopoulos's recipe [12] with short-term energy and spectral centroid is developed for VAD. Given a short-term signal $s(n)$ with N samples, the energy is:

$$E = \frac{1}{N} \sum_{n=1}^{N} |s(n)|^2, \qquad (1)$$

and given the corresponding Discrete Fourier Transform (DFT) $S(k)$ of $s(n)$ with K frequency components, the spectral centroid can be formulated as:

$$C = \frac{\sum_{k=1}^{K} k S(k)}{\sum_{k=1}^{K} S(k)}. \qquad (2)$$

The Short-Term Energy (STE) E is used to discriminate silence with environmental noise, and the Spectral Centroid (SC) C can be used to remove non-environmental noise, i.e. non-speech sound, such as coughing, mouse clicking and keyboard tapping, since they normally have different SCs compared to human speech. When computing the frame-level E and C, a 50 ms window size and a 25 ms hop size are used.

To set the overall threshold, only when E and C are both above their thresholds T_E and T_C, the speech frame is considered to be voiced, otherwise, it will be removed. These thresholds are adjusted to be slightly higher to enforce a stricter VAD algorithm and ensure the quality of the captured voiced sections. This is achieved by tuning the signal median smoothing parameters, such as step size and smoothing order, as well as setting the thresholds T_E and T_C as a weighted average of the local maxima in the distribution histograms of the short-term energy and spectral centroid respectively. Fig. 2 is an example of applying different median filter smoothing step sizes to STE and SC. Larger step size (e.g. 7) and order (e.g. 2) are used in order to achieve more stricter VAD.

C. Feature Extraction, Normalization and Concatenation

The 39-dimensional Mel-Frequency Cepstral Coefficients (MFCCs) with delta and double delta were generated from the preprocessed speech, following Ellis's recipe [13]. They

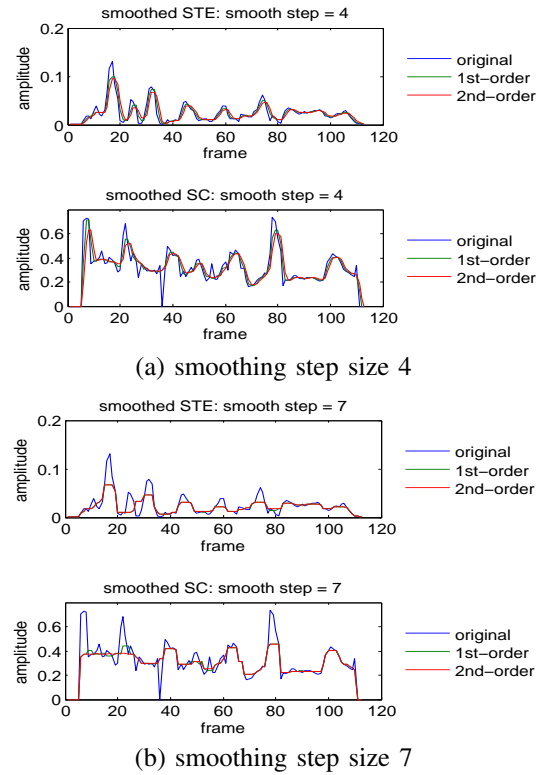(a) smoothing step size 4

(b) smoothing step size 7

Fig. 2. Short-term energy and spectral centroid with different median filter smoothing steps and orders.

were extracted using overlapped 25 ms Hamming windows which hop every 10 ms. Then, the features of each speaker were normalized with his own mean and variance (speaker-level MVN, or SMVN), instead of using the overall mean and variance (global-level MVN, or GMVN). Fig. 3 shows SMVN though converges slower, but helps to achieve better feature frame level training and validation accuracies in network training. It is slightly counter-intuitive, since SMVN overlaps speaker patterns on top of each other. However, it can match the instances of patterns from the same speaker better than GMVN as the training goes.
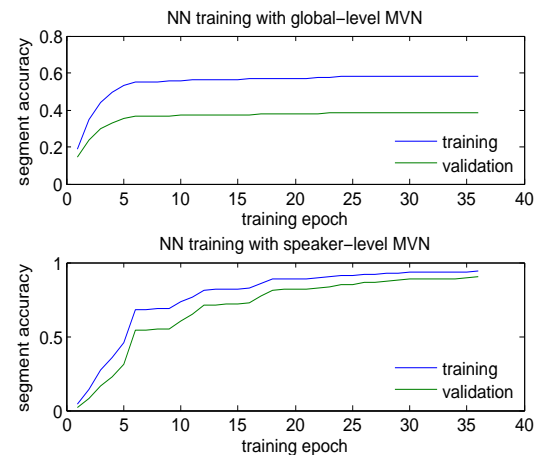

Fig. 3. Comparison of global-level MVN vs. speaker-level MVN in NN training in terms of training and validation frame accuracies.

To capture the transition patterns within longer durations, these 39-dimensional feature frames were concatenated to form overlapped longer frames. In this work, 10 frames (100 ms) were concatenated with hop size of 3 frames (30 ms) as shown in Fig. 4.
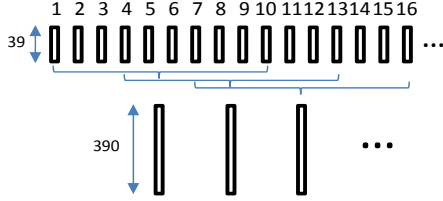

Fig. 4. Feature concatenation example with a window size of 10 frames and a hop size of 3 frames.

III. NEURAL NETWORK SPEAKER CLASSIFICATION

The concatenated features (e.g. 390 dimensional feature vectors) are used as the input to a neural network speaker classifier. As mentioned in the first paragraph of Sec. II, the "SX" and "SI" sentences of the first 200 male speakers were used for training, and the remaining "SA" sentences from the same set of speakers were used for testing.

A. Cost Function and Model Structures

Ng's neural network training recipe for hand-written digit classification [14] is used here, which treats the multi-class problem as K separate binary classifications. It is considered to be the generalization of the cost function of binary classification using logistic regression, which is built on slightly different concepts compared with the cross-entropy cost function with softmax as the output layer [15].

Given M samples, K output classes, and L layers, including input, output and all hidden layers in between, the cost function can be formulated as:

$$J(\Theta) = -\frac{1}{M}\left[\sum_{m=1}^{M}\sum_{k=1}^{K}\left(y_k^{(m)}\log(h_\theta(x^{(m)})_k) \right.\right. \quad (3)$$
$$\left.\left. + (1-y_k^{(m)})\log(1-h_\theta(x^{(m)})_k)\right)\right]$$
$$+\frac{\lambda}{2M}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\theta_{ji}^{(l)})^2$$

where $h_\theta(x^{(m)})_k$ is the kth output of the final layer, given mth input sample $x^{(m)}$, and $y_k^{(m)}$ is its corresponding target label. The 2nd half of Eq. (3) is the regularization factor to prevent over-fitting, where λ is the regularization parameter and $\theta_{ji}^{(l)}$ is the j-th row, i-th column element of the weight matrix $\Theta^{(l)}$ between l-th and $(l+1)$-th layers, i.e. the weight from i-th node in l-th layer to j-th node in $(l+1)$-th layer.

In this work, there is only 1 hidden layer ($L = 3$) with 200 nodes ($s_2 = 200$), the input feature dimension is 390 ($s_1 = 390$), and the speaker classifier was trained with data from 200 speakers ($s_3 = K = 200$). Therefore, the network structure is 390 : 200 : 200, with weight matrices $\Theta^{(1)}$ (200×391) and Θ^2 (200×201). The additional 1 column is a bias vector, which is

left out in regularization, since the change of bias is unrelated to over-fitting. In this example, the regularization part in Eq. (3) can be instantiated as

$$\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\theta_{ji}^{(l)})^2 = \sum_{i=1}^{390}\sum_{j=1}^{200}(\theta_{j,i}^{(1)})^2 + \sum_{i=1}^{200}\sum_{j=1}^{200}(\theta_{j,i}^{(2)})^2. \quad (4)$$

B. Model Training and Performance Evaluation

The neural network model is trained through forward-backward propagation. Denoting $z^{(l)}$ and $a^{(l)}$ as the input and output of the l-th layer, the sigmoid function

$$a^{(l)} = g(z^{(l)}) = \frac{1}{1+e^{-z^{(l)}}} \quad (5)$$

is selected as the activation function, and the input $z^{(l+1)}$ of the $(l+1)$-th layer can be transformed from the output $a^{(l)}$ of the l-th layer, using $z^{(l+1)} = \Theta a^{(l)}$. Then, $h_\theta(x)$ can be computed through forward propagation: $x = a^{(1)} \to z^{(2)} \to a^{(2)} \to \cdots \to z^{(L)} \to a^{(L)} = h_\theta(x)$. The weight matrix $\Theta^{(l)}$ is randomly initiated using continuous uniform distribution between $(-0.1, 0.1)$ and then trained through backward propagation of $\partial J/\partial\theta_{j,i}^{(l)}$, by minimizing $J(\Theta)$ using Rasmussen's conjugate gradient algorithm, which handles step size (learning rate) automatically with slope ratio method[16].

In evaluating the classifier performance, the sigmoid output of the final layer $h_\theta(x^{(m)})$ is a K-dimensional vector, each element in the ranges of $(0, 1)$. It serves as the "likelihood" to indicate how likely it is to classify m-th input frame into one of the K speakers. The speaker classification can be predicted by the sum of log likelihood of M input frames (prediction scores), and the predicted speaker ID k^* is the index of its maximum:

$$k^* = \arg\max_{k\in[1,K]}\left(\sum_{m=1}^{M}\log(h_\theta(x^{(m)})_k)\right). \quad (6)$$

M can range from 1 to the entire frame length of the testing file. If $M = 1$, the accuracy achieved is based on individual frames, each of which is 100 ms (window duration T_{win} in feature concatenation) with 30 ms of new data, compared with the previous frame. On the other hand, if M is equal to the total number of frames in file, the accuracy is file-based. The average duration of sentences (i.e. file length) is about 2.5 seconds. In general, larger M leads to higher accuracy. Given the best model available with the network structure 390 : 200 : 200, Fig. 5 demonstrates an example of file-level prediction score of 13-th speaker (MPGR0). It shows the peak of positives (in the green circle) is slightly dropped but still distinguishable enough to all other negatives, from the file `SI1410` in the training set, to the file `SA1` in the testing set.

Using this model, the file-level training and testing accuracies at 200 speaker size are both 100%, as indicated in Table I. The frame-level testing accuracy is 71.42%, which indicates that 71.42% frames in the testing set, with duration as little as 0.1 second, can be classified correctly. It also shows the minimum, mean, and maximum number of consecutive feature frames needed and their corresponding durations in order to achieve 100% accuracy, evaluated through all files in both training and testing datasets. Since the next frame provides

(a) `SI1410` in training
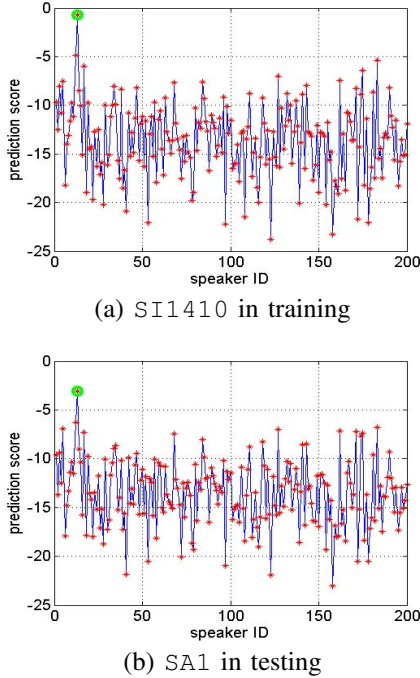

(b) `SA1` in testing

Fig. 5. File-level prediction scores of 13th speaker (MPGR0) in training and testing sets respectively.

TABLE I.   NN-BASED SPEAKER CLASSIFICATION PERFORMANCE WITH FIRST 200 MALE IN 8K TIMIT (0.1 SEC./FRAME, ~2.5 SEC./FILE)

Dataset	Accuracy (%)		Frame (sec.) needed for 100% accuracy		
	frame	*file*	*min*	*mean*	*max*
train	93.29	100	2 (0.13)	3.23 (0.17)	5 (0.22)
test	71.42	100	6 (0.25)	13.55 (0.48)	37 (1.18)

only 30ms (hop duration T_{hop} in feature concatenation) additional information, compared with the current frame, given the number of frames needed N, the formula to compute the corresponding required duration T is

$$T = (N - 1) \times T_{hop} + 1 \times T_{win}. \qquad (7)$$

With this formula, it requires only 13.55 frames (0.48 second) on average, to achieve 100% accuracy in the testing dataset.

Using the training data to test is normally not legitimate, and here it is used merely to get a sense of how the accuracy drops when switching from training data to testing data.

C. Model Parameter Optimization

The current neural network model with the structure $390 : 200 : 200$ is actually the best one in terms of highest frame-level testing accuracy, after grid searching on a) the number of hidden layers $(1, 2)$, and b) the number of nodes per hidden layer $(50, 100, 200, 400)$, with a subset containing only 10% randomly selected training and testing data.

Once the ideal network structure is identified, the model training is conducted with a regularization parameter λ in the cost function $J(\Theta)$, which is iteratively reduced from 3 to 0 through training. This dynamic regularization scheme is

experimentally proved to avoid over-fitting and allow more iterations to reach a refined model with better performance.

The training is set to be terminate once the testing frame accuracy cannot be improved more than 0.1% in the last 2 consecutive training iterations, which normally takes around 500 to 1000 iterations. The training set is at 200 speaker size with 20 seconds speech each. It is fed in as a whole batch of data, which requires about 1 hour to train, on a computer with i7-3770 CPU and 16 GB memory. Therefore, the computational cost is certainly manageable.

IV.   NEURAL NETWORK SPEAKER VERIFICATION

This section first introduces the mechanism of converting speaker classification into speaker verification; then describes the method of developing speaker-specific thesholds to shift verification outputs; finally it evaluates the system with metrics such as Equal Error Rate (EER).

A. Verification Mechanism

In speaker verification, the assumption that any input speaker will be one of the in-domain speakers is no longer kepted. When the testing speaker is claimed to be speaker k and the highest output score is also from the k-th output nodes, he might be a imposter, who is more similar to speaker k, and less similar to the rest of $K - 1$ enrolled (in-domain) speakers. So Eq. (6) in Subsec. III-B is no longer hold and a threshold is necessary to determine if the testing speaker is similar enough to the targeting speaker and can be verified as speaker k.

Let the mean K-dimensional output prediction vector over feature frames for client speaker k, given features x_l of speaker l be:

$$\boldsymbol{O}(k,l) = \frac{1}{M} \sum_{m=1}^{M} \log(h_\theta(x_l^{(m)})_k), \qquad (8)$$

where M is the number of frames in the testing feature. In this project, client speakers are the first 200 male speakers in TIMIT ($K = 200$), and the imposters (out-of-domain) are the ramaining 126 speakers ($L = 126$). In positive verification, where $l = k$, and the k-th value on $\boldsymbol{O}(k,k)$, i.e. $\boldsymbol{O}_k(k,k)$ should be high; while in negative verification, where $l \in [1, L]$, and $\boldsymbol{O}_k(k,l)$ should be low. If

$$\boldsymbol{O}_k(k,k) > \text{any}(\boldsymbol{O}_k(k,l)), \quad l \in [1, L], \qquad (9)$$

then, the k-th speaker can be correctly verified. In our experiment, $\boldsymbol{O}(k,k)$ and $\boldsymbol{O}(k,l)$ are actually normalized over K output node dimension, and the normalized versions are:

$$\boldsymbol{O}'(k,k) = \frac{\boldsymbol{O}(k,k)}{\sum_{k=1}^{K} \boldsymbol{O}(k,k)}, \boldsymbol{O}'(k,l) = \frac{\boldsymbol{O}(k,l)}{\sum_{k=1}^{K} \boldsymbol{O}(k,l)}. \qquad (10)$$

It is found to achieve better verification accuracy by penalizing the ones with strong competing speakers. Fig. 6 shows the accuracy vs. number of testing files (up to 5 since there are 5 sentences from "SI" and "SA" categories). For example, the mean accuracy is 61.7% when speakers are tested with individual files and 85.25% when tested with a combination of two files ($\binom{5}{2} = 10$ cases). The sentences duration is about 2.5 seconds each, so it is similar to the accuracy with testing duration 2.5 seconds, 5 seconds, etc. For each out of the 200 client speakers, the accuracy is binary, either 1, i.e. Eq. (9) is satisfied, or 0 otherwise.
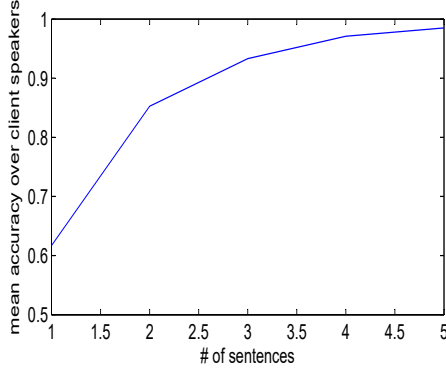
Fig. 6. Verification accuracy (1 in-domain client speaker vs. 126 out-of-domain imposters) vs. number of testing files, averaged over all 200 in-domain speakers in TIMIT.

B. Speaker Specific Thresholding

The accuracy measurement above will drop significantly when the imposter size is getting larger. In fact, it is merely an analysis to demonstrate the challenge to maintain high accuracy with a large imposter size which is rare in the real scenario. Next, the speaker-specific thresholds will be obtained by finding the Gaussian distributions of the positive (testing speaker is the client speaker) and negative (testing speakers is one of the imposters) samples, using Bayes rule.
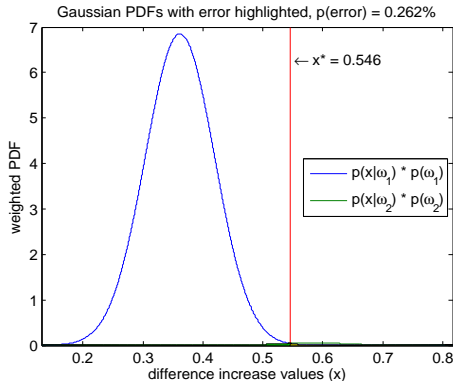

Fig. 7. Example of thresholding with 2 Gaussians distributions of positive and negative samples. Sample values are collected with combinations of 2 files (10 cases with ∼ 5 seconds in duration), i.e. 10 positives vs. 1260 negatives.

Since the positive and negative is extremely skewed with current 126 imposter size (i.e. positive:negative is 1:126), the distribution for the positive samples has a very low prior and almost invisible in Fig. 7. However, the estimated threshold, which is the intersection of the two Gaussians, can be still found by solving the Eq. (11) using the root finding method, which first reformats the Eq. (11) to quadratic function $ax^2 + bx + c = 0$, and then represents x by a, b, c.

$$\frac{p_1}{\sigma_1} e^{\frac{(x - u_1^2)}{2\sigma_1}} = \frac{1 - p_1}{\sigma_2} e^{\frac{(x - u_2)^2}{2\sigma_2}}. \quad (11)$$

C. Performance with Optimized Thresholds

With the speaker-specific thresholds T_k, $k \in [1, K]$, the output normalized prediction vector is shifted by

$$\boldsymbol{O}'(k, l) \to \boldsymbol{O}'(k, l) - T_k, l \in \{k, [1, L]\}. \quad (12)$$

Then, ROC curve is computed to find the Equal Error Rate (EER), which is a common performance indicator to evaluate biometric systems. EER equal to False Positive Rate (FPR), when $FPR + TPR = 1$. Fig. 8 demonstrates the ROC curve, when verifying with length of 2 files (∼ 5 seconds). By offsetting outputs with speaker-specific thresholds, the EER is reduced from 14.9% to 5.9%. Another metric Area Under Curve (AUC) is 98.05%, and the global threshold corresponding to this best EER is -0.0941.
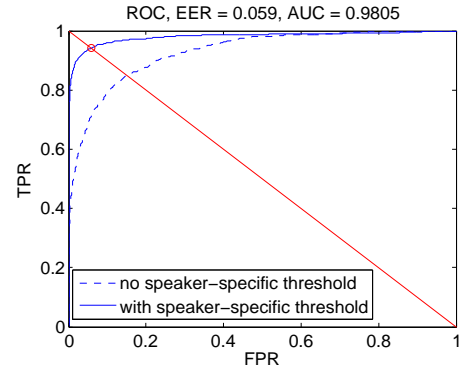

Fig. 8. ROC when verifying with length of 2 files (∼ 5 seconds), with or without speaker-specific thresholds.

V. CONCLUSION AND FUTURE WORK

This work demonstrated a novel neural net framework for speaker classification and verification with enhanced features. The performance is tested using TIMIT corpus with 8K sampling rate. For speaker classification, 200 speakers can be classified correctly with data no more than 1.18 seconds; For speaker verification, the EER is 5.9%, when verifying 200 in-domain speakers with 126 imposters, using speech about 5 seconds long (2 TIMIT files). Though the performance of speaker classification and verification systems is difficult to compare, due to various database condition, and enrollment and testing scenarios [10], 100% classification rate using about 1 second audio and less than 6% EER using 5 seconds data in speaker verification, is still among one of the very competitive performances in most of the cases [11].

This is achieved by combining all the essential components, including 1) feature engineering, such as VAD/silence removal, speaker-level MVN, feature concatenation to capture transitional information, etc., 2) neural network setup, model parameter optimization, training with dynamically reduced regularization parameter in speaker classification, and 3) output score normalization and speaker-specific thresholding in speaker verification.

There is still much room for potential improvement. First, the enrollment process is typically one-by-one, rather than enrolling a group of speakers as a whole, so the recursively model training and updating need to be addressed. Second, more challenging and noisy database should be considered to added in, in order to deal with channel normalization and system robustness. Third, combining current neural network approaches with other state-of-the-art methods, such as GMM-UBM [5] and i-vector [7], [8] is also desired.

REFERENCES

[1] Alan L Higgins, "Speaker verifier using nearest-neighbor distance measure," Aug. 16 1994, US Patent 5,339,385.

[2] Frank K Soong, Aaron E Rosenberg, Bling-Hwang Juang, and Lawrence R Rabiner, "Report: A vector quantization approach to speaker recognition," *AT&T technical journal*, vol. 66, no. 2, pp. 14–26, 1987.

[3] David Snyder, Daniel Garcia-Romero, and Daniel 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.

[4] Kevin R Farrell and Richard J Mammone, "Speaker identification using neural tree networks," in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*. IEEE, 1994, vol. 1, pp. I–165.

[5] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1, pp. 19–41, 2000.

[6] Patrick Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13*, 2005.

[7] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[8] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas A Reynolds, and Reda Dehak, "Language recognition via i-vectors and dimensionality reduction.," in *INTERSPEECH*, 2011, pp. 857–860.

[9] William M Campbell, Douglas E Sturim, and Douglas A Reynolds, "Support vector machines using gmm supervectors for speaker verification," *IEEE signal processing letters*, vol. 13, no. 5, pp. 308–311, 2006.

[10] Douglas Reynolds, "An overview of automatic speaker recognition," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)(S. 4072-4075)*, 2002.

[11] Benoît GB Fauve, Driss Matrouf, Nicolas Scheffer, Jean-François Bonastre, and John SD Mason, "State-of-the-art performance in text-independent speaker verification through open-source software," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 1960–1968, 2007.

[12] Theodoros Giannakopoulos, "A method for silence removal and segmentation of speech signals, implemented in Matlab," *University of Athens, Athens*, 2009.

[13] Daniel P. W. Ellis, "PLP and RASTA (and MFCC, and inversion) in Matlab," 2005, online web resource.

[14] Andrew Ng, "Neural networks: learning (lecture week 5 and lab 4), from coursera course: Machine learning," https://www.coursera.org/learn/machine-learning/home/info, Accessed: 2016-07-21.

[15] Sargur Srihari, "Neural network training (lecture 5.3), from university of buffalo course: Machine learning and probabilistic graphical models," http://www.cedar.buffalo.edu/~srihari/CSE574/, Accessed: 2016-07-21.

[16] Carl Edward Rasmussen, "Gaussian processes for machine learning," 2006.