DEEP BOTTLENECK FEATURES FOR I-VECTOR BASED TEXT-INDEPENDENT SPEAKER VERIFICATION

Sina Hamidi Ghalehjegh and Richard C. Rose

Department of Electrical and Computer Engineering, McGill University, Montreal, Canada

sina.hamidighalehjegh@mail.mcgill.ca, rose@ece.mcgill.ca

ABSTRACT

This paper describes the application of deep neural networks (DNNs), trained to discriminate among speakers, to improving performance in text-independent speaker verification. Activations from the bottleneck layer of these DNNs are used as features in an i-vector based speaker verification system. The features derived from this network are thought to be more robust with respect to phonetic variability, which is generally considered to have a negative impact on speaker verification performance. The verification performance using these features is evaluated on the 2012 NIST SRE core-core condition with models trained from a subset of the Fisher and Switchboard conversational speech corpora. It is found that improved performance, as measured by the minimum detection cost function (minDCF), can be obtained by appending speaker discriminative features to the more widely used mel-frequency cepstrum coefficients.

Index Terms— Speaker verification, deep neural network, i-vector.

1. INTRODUCTION

This paper presents an approach for applying deep neural networks (DNNs) to an i-vector based text-independent speaker verification in an attempt to improve its performance. In a text-independent speaker verification task, the users are allowed to utter any phrases with no constraints. This complicates the verification problem, since the phonetic contents of the reference and test utterances might be completely different. As a result, effective methods must be developed to account for phonetic variability.

Basically, any mismatch between the training and test conditions is regarded as session variability and is the major source of degradation in the performance of a speaker verification system [1]. Compensating for session variability can be performed in three spaces: feature space, model space and score space. In the feature space, the unwanted variabilites are removed from the acoustic features before model training or verification. This includes cepstral mean and variance normalizations [2] and feature mapping [3]. In the model space, methods are used to eliminate the irrelevant information during training the speaker model, such as speaker model synthesis [4] and joint factor analysis [5]. In the score space, the raw verification scores are shifted and scaled accordingly in an attempt to produce scores within a similar dynamic range so that a common threshold can be applied. Tnorm is an example of such methods [6].

In this paper, a feature space compensation method is proposed. The method is based on the DNN framework. Recently, DNN has demonstrated state-of-the-art performance for automatic speech recognition (ASR) tasks [7]. A phonetic discriminative DNN is used directly to estimate the probability distribution of context-dependent phones, given the input features [8]. DNN can also be used to improve the performance of conventional Gaussian mixture model (GMM)-based ASR systems by providing additional features [9]. Here, tandem features generated by DNN are appended to the original features.

It is often argued that phonetic discriminative DNNs are performing implicit feature normalization across speakers in ASR systems [10]. Similar argument might be true for speaker discriminative DNNs. In the other words, these kind of networks might be performing implicit feature normalization across phonetics. The goal in this paper is to investigate this conjecture. We use a speaker discriminative DNN to generate bottleneck features. We speculate that this will enhance the speaker discriminative ability of the features, while reducing irrelevant information such as phonetic variability. Concatenating the bottleneck and original features will produce features which are more specific for speaker verification task. The features will then be used in training i-vector based speaker verification system.

Over recent years, the i-vector has become the state-of-the-art technology for text-independent speaker verification tasks [11]. The technique is inspired by joint factor analysis (JFA) framework [5, 12], where the Gaussian mixture model (GMM) mean is factorized into the speaker and channel factors using separate subspaces. In the i-vector framework, however, a single subspace is defined which contains both the speaker and channel factors simultaneously.

There had been, of course, attempts in building neural net-

This work was supported by Google Graduate Award.

work based speaker verification before advent of DNN [13, 14, 15]. However, due to inadequate hardwares and learning algorithms for building big networks, the obtained gains were very limited. In [16], DNN is used as a feature extractor to improve the performance of a GMM-universal background model (GMM-UBM) based text-dependent speaker verification system.

The paper is organized as follows. Section 2 briefly describes the i-vector framework and deep neural network. Section 3 describes our proposed method. An experimental study is performed to evaluate the proposed approach in Section 4. Section 5 concludes the paper.

2. BACKGROUND

2.1. i-Vector Framework

The essence of the i-vector framework is to define a single subspace which represents the speaker and channel variabilities simultaneously [11]. A given speaker- and channel-dependent GMM supervector $\mathbf{s}$ can be modeled as:

$$\mathbf{s} = \mathbf{m} + \mathbf{T}\mathbf{w}, \tag{1}$$

where $\mathbf{m}$ is the UBM speaker- and channel-independent mean supervector. The low-rank matrix $\mathbf{T}$ is referred to as total variability matrix and consists of bases representing main directions of speaker and channel variabilities and the vector $\mathbf{w}$ has a prior standard normal distribution. The i-vector $\hat{\mathbf{w}}$ is the MAP point estimate of the variable $\mathbf{w}$. In the other words, the i-vector is the mean of posterior probability of $\mathbf{w}$, $p(\mathbf{w}|\mathbf{X})$, for a given set of features $\mathbf{X}$.

The i-vector framework can be regarded as feature extractor rather than a modeling framework. A fast scoring method can then be used to determine if test and target i-vectors are from the same speaker. It is believed that the magnitude of i-vector conveys non-speaker information [11]. Hence, only the angle between two i-vectors can be considered as the score:

$$\text{score}(\hat{\mathbf{w}}_{\text{target}}, \hat{\mathbf{w}}_{\text{test}}) = \frac{\langle \hat{\mathbf{w}}_{\text{target}}, \hat{\mathbf{w}}_{\text{test}} \rangle}{\|\hat{\mathbf{w}}_{\text{target}}\| \|\hat{\mathbf{w}}_{\text{test}}\|}. \tag{2}$$

2.2. Deep Neural Network

Deep neural networks (DNNs) are a generalization of single hidden layer feed forward neural networks, in that they are generally assumed to include multiple hidden layers [7]. DNNs have been widely used in many classification problems in machine learning. For example, DNNs are used to predict state observation probabilities in context dependent phone models for hybrid DNN-hidden Markov models (DNN-HMMs). ASR systems based on these models have demonstrated state-of-the-art performance on a variety of speech recognition tasks [8]. A DNN is parameterized by the weights associated with its input, hidden and output layers. Each layer consists of a number of units where each unit contains a non-linear activation function and a bias vector. The outputs of each layer are multiplied by a weight matrix and presented as inputs to the following layer. For the hidden layers, the sigmoid function $\sigma(z_j) = 1/(1 + \exp(-z_j))$ is often used as the activation function. For the output layer, which is the classification layer, the softmax non-linearity, $\text{softmax}(z_j) = \exp(z_j)/\sum_j \exp(z_j)$, is used as the activation function. The parameters of the DNN model, θ, i.e. the set of weight matrices and bias vectors, can be estimated by optimizing a cross entropy based criterion:

$$C(\theta) = -\sum_j d_j \log P(j|\mathbf{x}, \theta), \tag{3}$$

where $P(j|\mathbf{x}, \theta)$ is the posterior probability of class j and $\mathbf{x}$ is the input feature vector to the DNN. The target value, d_j, for class j takes the value of one if j is the target class and zero otherwise. These target values are used as supervision during DNN training. A stochastic gradient descent (SGD) based algorithm, referred to as the back-propagation (BP) algorithm [17], is used to minimize (3). The gradient of the cost function is computed over a randomized minibatch of training data and used in updating the model parameters. Generally, gradient based algorithms are susceptible to local optima. Initialization of DNN parameters seems to be a practical solution for this problem. Unsupervised restricted Boltzmann machine (RBM) pre-training is used to initialize the DNN parameters before performing the BP algorithm [7, 8].

3. DEEP BOTTLENECK FEATURES

Speaker verification involves automatically verifying the claimed identity of an individual given an utterance spoken by that individual. However, human speech utterances exhibit many sources of variability that are not relevant to determining a speaker's identity. Depending on the application, such as robustness and speaker discrimination, different feature engineering methods can be used to characterize the speech utterances [18]. The spectral-based features, including Mel-frequency cepstrum coefficient (MFCC) and perceptual linear predictive (PLP), are the most widely used features in practice. However, these features are mainly engineered for speech recognition tasks and were originally developed for discriminating the phonetic information, rather than speaker information.

In this section, a DNN is used as a feature generator to provide features which are more relevant to the speaker verification task. The tandem approach is the most common approach in extracting features from a DNN [19]. In this approach, the activations of a hidden layer or output layer are considered as high-level features. If the hidden layer is narrow, i.e. its number of units is less than other hidden layers, the features are called bottleneck features.

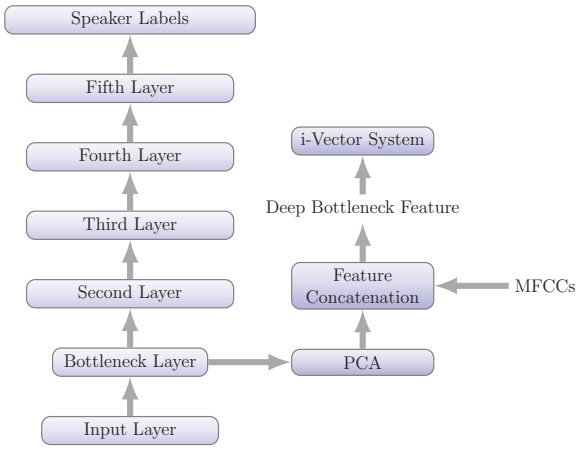

Fig. 1. Deep bottleneck features for speaker verification. The example network consists of an input layer, five hidden layers and an output layer. The first hidden layer is the bottleneck layer.

DNNs used in hybrid HMM-DNN speech recognition systems are trained to discriminate between state labels associated with context dependent phonetic classes. However, the bottleneck DNNs used in this work are trained using speaker labels as targets. It is hoped that the activations obtained from the bottleneck layer of these networks will provide an embedded representation that is in some way optimal for speaker discrimination. After training the DNN, the dimensionality of bottleneck features are reduced using principal component analysis (PCA). They are then appended to the original MFCC features. The combined feature set are referred to here as deep bottleneck features. These features can be used in any speaker verification system including systems based on GMM-UBM, joint factor analysis (JFA) and i-vector. Figure 1 displays the proposed deep bottleneck features for speaker verification where the first hidden layer is considered as the bottleneck layer. Here, the deep bottleneck features are being fed into an i-vector based speaker verification system.

As we will discuss in Section 4, the amount of speech data available in this work for speaker verification training is limited. As a result, we are facing with data scarcity problem. This problem, however, is interesting since for some languages it is difficult to obtain large speech corpora for configuring speaker verification systems for that language. The proposed method provides a solution for this problem by extracting additional information from existing speech data, which has effectively no cost.

4. EXPERIMENTAL STUDY

This section presents an experimental study evaluating the performance of the approach described in Section 3. Performance is reported as the equal error rate (EER) and minimum detection cost function (minDCF) on the female part of Fisher and Switchboard conversational speech corpora. minDCF is defined as the point where the detection threshold minimizes the cost function $0.1 \times P(\text{False Rejection}) + 0.99 \times P(\text{False Acceptance})$. After introducing the task domain and describing how the baseline text-independent speaker verification system is trained, we will present the speaker verification results using the proposed approach. Acoustic feature extraction was done using the HTK toolkit [20] and all the speaker verification training was done using ALIZE toolkit [21]. The DNN training was carried out on CUDA-capable GPU hardware[1]. A Python-based DNN trainer[2] based on Gnumpy [22] was adapted for this work. The evaluation of the system, such as plotting the DET curves, is done using the DETware toolbox (for MATLAB) by NIST[3].

4.1. Baseline System

The baseline text-independent gender-dependent system is constructed using the female part of Fisher English conversational speech corpus part 1. This corpus is the first half of conversational telephone speech (CTS) dataset. It consists of 5850 two-sided speech conversations, each with a duration of up to 10 minutes. The corpus is collected from 4174 female speakers, with approximately 328 hours of data. The evaluation of the verification system is done using the Switchboard1-Phase1 corpus. This corpus consists of 1022 5-minute two-sided speech conversations from 88 female speakers. We used only a subset of this data which does not suffer from the crosstalk problem. The subset consists of 60 female speakers with 531 utterances.

In both corpora, the speech is parameterized using 19 MFCCs, normalized energy and the first and second differences of these parameters to give a 60 dimensional acoustic vectors. An energy detector is used to determine the speech segments. Zero mean and unit variance normalizations are then performed on speech segments of each conversation. A UBM of 2048 Gaussians with diagonal covariances is trained using the Fisher data. The UBM is then used to train a 100-dimensional i-vector extractor. Consine distance between the i-vectors of target and test speakers is used as the score.

Evaluation is performed using the 2012 NIST SRE core-core condition. Here, the entire speech utterance is used to train the target i-vector. However, to train the test i-vector, we carry out the experiments using three different speech durations including 10 seconds, 30 seconds and entire utterance. We call these conditions as core-10sec, core-30sec and core-core during reporting the results. These conditions comprise 4959, 1510 and 471 files, respectively. Table 1 summarizes speaker verification results. The first row of the table shows

[1] http://www.calculquebec.ca/index.php/en/
[2] http://www.cs.toronto.edu/~gdahl/gdbn.tar.gz
[3] http://www.itl.nist.gov/iad/mig/tools/DETware_v2.1.targz.htm

Table 1. Speaker verification performance in terms of EER (%) and minDCF on 2012 NIST SRE core-core condition. The bottleneck features are obtained from the first hidden layer with 512 units. BL and BN stand for baseline and bottleneck, respectively.

	core-10sec		core-30sec		core-core	
System	EER	DCF	EER	DCF	EER	DCF
BL	**5.59**	2.50	**5.10**	2.21	**4.67**	1.98
BN	7.70	2.74	6.71	2.31	5.94	2.25
BN+MFCC	6.56	**2.36**	5.76	**2.05**	4.88	**1.93**

the baseline system performance for three test conditions.

It is important to note that the size of training corpus in our experiments is considered very small compared to those which are used nowadays to configure a speaker verification. As a result, the baseline EER and minDCF are higher than those reported elsewhere for this core task. Of course, it is difficult to assess whether the trends in performance obtained for the techniques evaluated here would also be observed with a larger training corpus. However, one of the motivations of this work is to determine whether speaker discriminative DNNs can be used to improve detection performance with limited data, and without the use of any additional data being collected for training DNNs. Hence, the goal is, not only to produce a feature set that is more robust with respect to known sources of variability, but also to achieve better performance with less training data.

4.2. Evaluation of Deep Bottleneck Features

To evaluate the proposed deep bottleneck features, a speaker-discriminative DNN is trained. The DNN consists of an input layer, 5 hidden layers and an output layer. A context window of 7 frames is used over 60-dimensional MFCC features to yield input features of size 420. All the 5 hidden layers, except the bottleneck layer, have 1024 units each. The position of the bottleneck layer and its number of units are determined empirically. It will be shown in this section that having the first hidden layer as the bottleneck layer with 512 units achieves the best performance. For output layer, 2000 speakers (out of 4174 speakers) are used as the output classes. The speakers used as the output classes contain the majority of the Fisher training data which is used in DNN training. DNN is initialized using RBM pre-training with a learning rate of 0.004. The learning rate of 0.08, along with a L2-norm regularization of coefficient 10^{-6} and a momentum of 0.9 are used for DNN fine-tuning. The DNN training is converged when the cross entropy cost on the validation set becomes flat over multiple epochs.

Once the DNN training is finished, the original MFCC features are passed through the network and bottleneck features are extracted. A zero mean normalization is performed on the bottleneck features and principal component analysis (PCA) is used to reduce the dimensions to 60 as the original MFCC features. Finally the features are unit variance normalized and appended to the original MFCC features to form the deep bottleneck features of size 120. These features are then used to train an i-vector based speaker verification system as explained in Section 4.1.

Second and third rows of Table 1 display the speaker verification results when bottleneck features and deep bottleneck features are used, respectively. We report the results in terms of EER and minDCF. The minDCF measure is used by the NIST as the primary evaluation figure, since EER uses an arbitrary detection threshold, which is not practically applicable [18]. We can see that an i-vector based speaker verification system trained on deep bottleneck features provides better performance in terms of minDCF in all test conditions. However, its performance is worse than the baseline in terms of EER.

Figure 2 displays the detection error trade-off (DET) plot for the same three test conditions displayed in Table 1 for baseline and deep feature systems. The plot shows that using deep features provides a significant improvement in performance at lower false alarm rates (FARs). One can speculate that this occurs because the cost function used for training the DNN and extracting bottleneck features is specially designed to perform classification. It can be argued that optimizing the speaker discriminative DNN for closed set speaker classification may not be a good criterion when the final task is actually open set speaker verification. Since the cost function associated with minDCF emphasizes operating points with very low false alarm rates, using deep features results in better minDCF performance compared to baseline. On the other hand, since the EER falls on the regions where the deep features perform more poorly, we do not get an improvement in terms of EER.

4.3. Evaluation of Different Bottleneck Layer Position

It is believed that each hidden layer in a DNN captures a particular level of feature representation, depending on its position. The higher layers, i.e. layers close to output layer, produce more abstract representations compared to lower layers. It means that the features obtained from higher layers are more suitable for classification purposes. In this section, we examine the effect of bottleneck layer position in the performance of speaker verification system. Table 2 displays the result of this experimentation. The results show that when the bottleneck layer is situated at the first hidden layer it gives the best performance. Even though the results seem in contradiction with the feature representation property of DNN, however, the conclusion is consistent with the one obtained in [16]. We believe that this is because different training and test datasets have been used for this experiment and the speaker populations do not overlap. Therefore, none of the

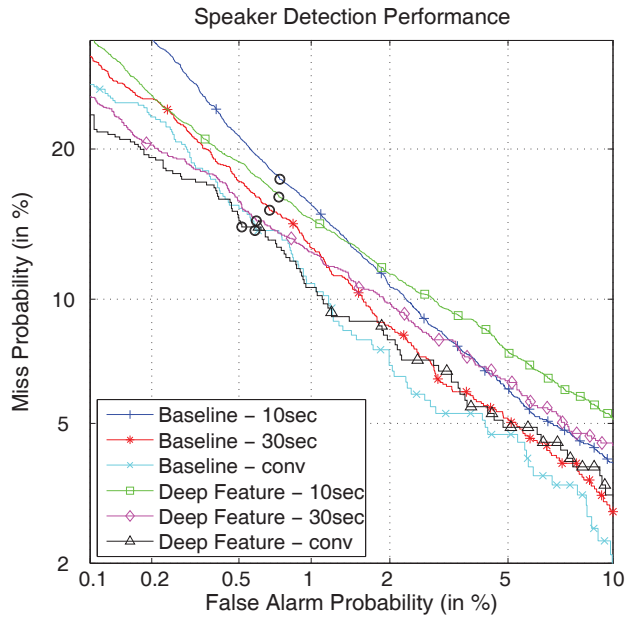

Fig. 2. Comparison of detection error trade-off (DET) plot for different test conditions. The circles determine the minDCF points.

Table 2. Speaker verification performance for different bottleneck layer position in terms of EER (%) and minDCF on 2012 NIST SRE core-core condition. In all experiments the number of bottleneck layer units is 512.

	core-10sec		core-30sec		core-core	
Position	EER	DCF	EER	DCF	EER	DCF
1st Layer	**6.56**	**2.36**	**5.76**	**2.05**	**4.88**	**1.93**
2nd Layer	7.76	2.75	7.09	2.65	6.79	2.69
3rd Layer	8.05	3.06	7.68	2.99	7.86	3.02
4th Layer	8.57	3.31	8.34	3.19	8.70	3.28
5th Layer	9.22	3.53	8.91	3.44	8.92	3.64

speakers in the test corpus has representative in the output layer. As a result, going higher in the network might harm rather than help.

4.4. Evaluation of Different Number of Bottleneck Layer Units

Table 3 summarizes the results of speaker verification system for different number of bottleneck layer units. The effect of different units number ranging from 60 to 1024 have been examined. The results show that varying the units number has a minor impact on the verification performance.

Table 3. Speaker verification performance for different number of bottleneck layer units in terms of EER (%) and minDCF on 2012 NIST SRE core-core condition. In all experiments the bottleneck layer is situated at the first hidden layer.

	core-10sec		core-30sec		core-core	
#units	EER	DCF	EER	DCF	EER	DCF
60	7.36	2.74	6.69	2.49	6.00	2.53
100	6.80	2.41	6.03	2.11	5.10	1.96
200	6.84	2.47	5.89	2.19	5.15	2.03
348	**6.52**	2.39	**5.50**	2.05	5.10	1.98
448	6.65	2.40	5.76	2.11	5.10	1.93
512	6.56	**2.36**	5.76	**2.05**	**4.88**	**1.93**
768	6.53	2.45	5.56	2.18	5.10	2.01
1024	6.55	2.41	5.76	2.18	5.07	2.04

5. CONCLUSIONS

Deep bottleneck features were proposed for improving performance in i-vector based text-independent speaker verification. A speaker-discriminative deep neural network with a bottleneck layer was used to extract features. The features were then concatenated with MFCC features. Several experiments were performed to examine the effect of multiple positions of the bottleneck layer and multiple numbers of units in the bottleneck layer. We observed that when the first hidden layer was used as the bottleneck layer with 512 units, the speaker verification system provided the best performance. The minDCF results for this configuration outperformed the baseline i-vector based speaker verification system.

The ultimate goal was to produce feature vectors which are more relevant for speaker verification task by reducing the impact of sources of undesired variability. The hope was also to circumvent the issue of data scarcity by providing additional information to MFCC features. Since the bottleneck features were extracted from a DNN which is trained to discriminate between speakers, using deep features provided better performance in regions of the operating characteristic curve with low false alarm rates. Future work will investigate whether it is possible to obtain more uniform improvements in detection performance by using an alternative optimization criterion for training the speaker discriminative DNN. One solution is that instead of maximizing the classification rate, one can directly maximize the detection rate [23].

6. REFERENCES

[1] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Speaker and session variability in GMM-based speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1448–1460, 2007.

[2] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 2, pp. 254–272, 1981.

[3] D. Reynolds, "Channel robust speaker verification via feature mapping," in *ICASSP*. IEEE, 2003, vol. 2, pp. 53–56.

[4] R. Teunen, B. Shahshahani, and L. Heck, "A model-based transformational approach to robust speaker recognition.," in *InterSpeech*, 2000, pp. 495–498.

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

[6] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, no. 1, pp. 42–54, 2000.

[7] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[8] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[9] H. Hermansky, D. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *ICASSP*. IEEE, 2000, vol. 3, pp. 1635–1638.

[10] A. Mohamed, G. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," in *ICASSP*. IEEE, 2012, pp. 4273–4276.

[11] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[12] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[13] K. Farrell, R. Mammone, and K. Assaleh, "Speaker recognition using neural networks and conventional classifiers," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, pp. 194–205, 1994.

[14] Rita H Wouhaybi and M Adnan Al-Alaoui, "Comparison of neural networks for speaker recognition," in *The 6th IEEE International Conference on Electronics, Circuits and Systems*, 1999, vol. 1, pp. 125–128.

[15] S. Kishore, B. Yegnanarayana, and S. Gangashetty, "Online text-independent speaker verification system using autoassociative neural network models," in *International Joint Conference on Neural Networks*, 2001, vol. 2, pp. 1548–1553.

[16] T. Fu, Y. Qian, Y. Liu, and K. Yu, "Tandem deep features for text-dependent speaker verification," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[17] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, 1988.

[18] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech communication*, vol. 52, no. 1, pp. 12–40, 2010.

[19] N. Morgan, "Deep and wide: Multiple layers in automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 7–13, 2012.

[20] S. J. Young, G. Evermann, M. J. F. Gales, D. Kershaw, G. Moore, J. J. Odell, D. G. Ollason, D. Povey, V. Valtchev, and P. C. Woodland, "The HTK book (for HTK version 3.4)," 2006.

[21] A. Larcher, J.-F. Bonastre, and H. Li, "ALIZE 3.0-Open-source platform for speaker recognition," *IEEE SLTC Newsletter*, 2013.

[22] T. Tieleman, "Gnumpy: an easy way to use GPU boards in Python," *Department of Computer Science, University of Toronto*, 2010.

[23] Joseph Keshet, David Grangier, and Samy Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, no. 4, pp. 317–329, 2009.