Odyssey 2012
The Speaker and Language Recognition Workshop
25-28 June 2012, Singapore

Bottleneck Features for Speaker Recognition

Sibel Yaman[1], Jason Pelecanos[1], Ruhi Sarikaya[2]

[1] IBM T. J. Watson Research Labs, Yorktown Heights, NY
[2] Microsoft Corporation, Redmond, WA

{syaman, jwpeleca}@us.ibm.com, ruhi.sarikaya@microsoft.com

Abstract

Bottleneck neural networks have recently found success in a variety of speech recognition tasks. This paper presents an approach in which they are utilized in the front-end of a speaker recognition system. The network inputs are mel-frequency cepstral coefficients (MFCCs) from multiple consecutive frames and the outputs are speaker labels. We propose using a recording-level criterion that is optimized via an online learning algorithm. We furthermore propose retraining a network to focus on its errors when leveraging scores from an independently trained system. We ran experiments on the same- and different-microphone tasks of the 2010 NIST Speaker Recognition Evaluation. We found that the proposed bottleneck feature extraction paradigm performs slightly worse than MFCCs but provides complementary information in combination. We also found that the proposed combination strategy with re-training improved the EER by 14% and 18% relative over the baseline MFCC system in the same- and different-microphone tasks respectively.

1. Introduction

The speech recognition community has seen a recent boom in the ways neural networks are utilized. In particular, deep neural networks with multiple layers of hidden nodes are shown to provide significant improvements in speech recognition performance, for example, see [1]. Bottleneck networks (BNs) constitute one such class of deep neural networks that do not require conversion of network outputs into features. A BN with three hidden layers is depicted in Figure 1. Shown is an information bottleneck in a middle layer that provides features of desired dimensionality. BNs are shown to perform comparably with or better than other published MLP systems with fewer parameters [2, 3, 4, 5].

There is also an emerging interest in using neural networks for speaker recognition. In [6], a BN was trained to classify each sample that consists of MFCC features of nine consecutive frames into one of 34 speaker classes. The training was performed by minimizing the sample-level cross-entropy. In [7], a factor analysis technique for a mixture of auto-associative neural networks is proposed to learn a low-dimensional subspace in the supervector space of last layer weights.

[1]This work was supported in part by Contract No. D11PC20192 DOI/NBC under the RATS program and by Contract No. W911NF-10-C-0025 by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the Army Research Laboratory (ARL). All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of the DOI/NBC, IARPA, the ODNI, or the U.S. Government.

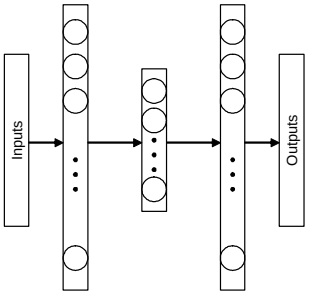

Figure 1: A bottleneck network topology with an information bottleneck in the middle layer.

In this paper, we use bottleneck features in the front-end of a state-of-the-art GMM-UBM speaker recognition system. We propose a recording-level training criterion and optimize it via an online learning algorithm. In our experiments, we found that a network trained with a recording-level training criterion outperforms one that is trained with a sample-level training criterion by up to 16% relative in terms of EER.

We also propose retraining an optimized network with additional scores from an independently trained system. The goal is to provide complementary information in the combination of the two. Our experiments indicate that the proposed combination technique can perform better than linear score combination of independently trained systems.

The organization of this paper is as follows: Section 2 presents our approach for training a bottleneck network for speaker recognition. Section 3 describes the proposed retraining technique for obtaining complementary information in combination. Our experimental findings are reported in Section 4. Finally, Section 5 summarizes our conclusions.

2. Bottleneck Features

BNs are deep neural networks with multiple hidden layers that create a bottleneck of information in a middle layer. The layers below the bottleneck focus on generating robust speaker-specific features while the upper layers focus on the discriminative learning of the speaker classes.

Let $\mathbf{W}^{(\ell)}$ denote the weight matrix and the scalar-valued bias of the ℓ^{th} layer of an L-layer network. Also let $\mathbf{\Theta}$ denote the set of all the weights and biases to be estimated during training, i.e., $\mathbf{\Theta} = \{\mathbf{W}^{(1)}, ..., \mathbf{W}^{(L)}\}$. Let $\mathbf{x}_{ijk}$ be the k^{th} feature vector from the j^{th} recording of the i^{th} speaker. For simplicity of notation, we will denote $\mathbf{x}_{ijk}$ as $\mathbf{x}$. It represents a concatenation of mean and variance normalized raw MFCC features of multiple adjacent frames and we refer to each $\mathbf{x}$ as a *sample*. For

each $\mathbf{x}$, each hidden node computes an output and propagates it to its next layer.

Each hidden and output node performs a linear combination followed by a non-linear operation (such as a sigmoid non-linearity) on its inputs. The output of the first layer (i.e., the input layer) is equal to the input itself, i.e., $\boldsymbol{\sigma}^{(1)} = \mathbf{x}$. The inputs to the ℓ^{th} layer are first linearly combined as $\mathbf{u}^{(\ell)} = \mathbf{W}^{(\ell-1)}\boldsymbol{\sigma}^{(\ell-1)}$ for $\ell = \{2, 3, ..., L\}$. A non-linearity (such as a sigmoid) is applied on $\mathbf{u}^{(\ell)}$, which yields outputs

$$\boldsymbol{\sigma}^{(\ell)}(\boldsymbol{\Theta}) = \frac{1}{1 + \exp[-\mathbf{u}^{(\ell)}]} \quad (1)$$

for $\ell = \{2, 3, 4, ..., L-1\}$. The output at the n^{th} hidden node of the output layer is typically calculated by performing a softmax-operation on its input, i.e., by computing

$$\sigma_n^{(L)}(\boldsymbol{\Theta}) = \frac{exp(u_n^{(L)})}{\sum_i exp(u_i^{(L)})} \quad (2)$$

where $\sigma_n^{(L)}$ can be interpreted as the *a posteriori* probability of the associated speaker label n.

The traditional criterion adapted to train neural networks for classification tasks is the cross-entropy criterion. It is formulated as

$$J_{XH}(\boldsymbol{\Theta}) = \sum_n t_n \log \sigma_n^{(L)}(\boldsymbol{\Theta}) \quad (3)$$

where t_n stands for the 0/1-valued target output at the n^{th} node. During training, the weights $\mathbf{W}^{(\ell)}$ are updated in proportion to the derivative of J_{XH} with respect to $\mathbf{W}^{(\ell)}$.

2.1. Training a Bottleneck Network

This paper proposes using a bottleneck network in the front-end of a speaker recognition system. The inputs are spectral features and the outputs are the labels assigned with the S speakers. In the speaker recognition application, the scores generated at the last layer represent one target score and S-1 non-target scores for each given recording. The features extracted from the bottleneck layer are decorrelated with a whitening transform and used as inputs to a GMM-UBM speaker recognition system.

The fact that the network outputs $\{\sigma_n^{(L)}\}$ are normalized over all speakers can be suboptimal for speaker recognition scenarios, which aim to avoid dependence on non-target speakers. Therefore, we propose using scores $\{u_n^{(L)}\}$ which do not require information from other speakers. The goodness of these scores can be evaluated with a log-likelihood ratio-based cost function, J_{LLR}, defined as [8]

$$J_{LLR} = \alpha \sum_T \log(1 + e^{-u_T - c}) + \beta \sum_N \log(1 + e^{u_N + c}) \quad (4)$$

where u_T and u_N denote target and non-target speaker scores, respectively, computed at the last layer. The two summations in (4) are over all target and non-target scores.

The term c is a constant and is equal to $\log(\frac{\pi}{1-\pi})$. The parameter π depends on the *a priori* target speaker probability, P_{target}, and the costs associated with miss and false-alarm errors (C_{miss} and C_{FA}). It is equal to $\frac{P_{target} \times C_{miss}}{P_{target} \times C_{miss} + (1 - P_{target}) \times C_{FA}}$. The weights α and β are equal to $\frac{\pi}{N_{target}}$ and $\frac{1-\pi}{N_{nontarget}}$, respectively, where N_{target} and $N_{nontarget}$ are the number of target and non-target trials. Following the specification of the 2008 NIST SRE, we used $C_{miss} = 10, C_{FA} = 1$ and $P_{target} = 0.01$.

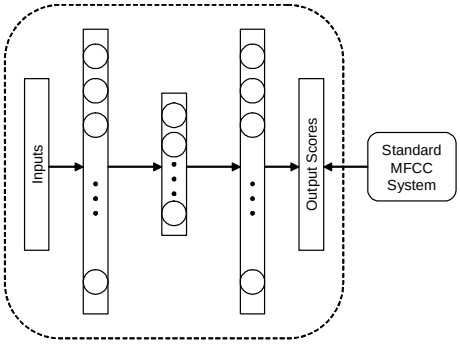

Figure 2: Scores generated by an independently trained MFCC system are leveraged to re-train a network.

2.2. A Recording-Level Training Objective

Optimizing J_{LLR} requires computing and accumulating error functions for each input sample. Despite the use of multi-frame features in the input, the training algorithm does not impose *any* global constraint on the classification decision of the entire speech segment. Therefore, the use of only local information gives the neural network a disadvantage over state-of-the-art counterparts that use both local information from multiple frames as well as global information from the entire recording.

To resolve these issues, we re-formulate J_{LLR} as a *recording-level* training objective. The time-average of the input at the last layer is computed as

$$\mathbf{u} = \mathbb{E}_C \left[\mathbf{u}^{(L)} \right] \quad (5)$$

where $\mathbb{E}_C$ denotes the expected value over all samples of the recording C. These time-averaged scores replace the u_T and u_N in the training criterion in Equation (4).

3. Boosting Network Performance with a State-of-the-Art System

In this section, we propose a network *re-training* technique to improve system performance using the knowledge of scores generated by an independently trained system. Figure 2 depicts the proposed system combination technique, where the scores generated by the bottleneck network $\{u_n\}$ are leveraged with scores from a standard system.

More specifically, a standard speaker recognition system generates a set of scores (denoted as $\{u_n^M\}$) for each given speaker and recording pair constituting a trial. The idea is to use these scores to help the network identify the 'easy' and 'difficult' trials. The network training criterion then focuses on correcting the difficult cases.

The training criterion to perform the proposed re-training is a reformulation of J_{LLR} in which $\{u_n(\boldsymbol{\Theta})\}$ are replaced with

$$u_n'(\boldsymbol{\Theta}) = \omega_B u_n(\boldsymbol{\Theta}) + \omega_M u_n^M + \beta \quad (6)$$

where ω_B and ω_M are the importance weights for the bottleneck and MFCC system scores and β is an offset. Note that the scores $\{u_n^M\}$ do not depend on $\boldsymbol{\Theta}$ but $\boldsymbol{\Theta}$ does depend on the scores $\{u_n^M\}$.

There is one important difference between the scores generated by the bottleneck network and those generated by a standard system: The scores $\{u_n\}$ have the interpretation of being

log-likelihood ratio (LLR) scores. However, the scores $\{u_n^M\}$ can be arbitrary scores. Therefore, before being used in training, $\{u_n^M\}$ need to be properly calibrated.

The calibration is achieved by estimating the scaling factors ω_B and ω_M, and the offset β that optimize J_{LLR} with the network parameters fixed. This is equivalent to solving the following problem:

$$\omega_B^*, \omega_M^*, \beta^* = \arg\min_{\omega_B, \omega_M, \beta} J_{LLR}(\omega_B, \omega_M, \beta | \mathbf{\Theta}_{fixed}). \quad (7)$$

The resulting ω_B^*, ω_M^*, and β^* produces a more compatible combination of the scores generated at the output layer of the neural network with scores generated by an independently trained system. Starting with this initialization, the scores at the output layer are refined at each iteration by solving the optimization problem

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} J_{LLR}(\mathbf{\Theta} | \omega_B, \omega_M, \beta). \quad (8)$$

The combination weights ω_B, ω_M, and β can be updated at each iteration (by solving (7)) to provide a better combination or can be held fixed at their initial estimates to help reduce computation time. This combination strategy is more powerful than the standard way of system combination since the bottleneck feature-based system was exposed to information from the state-of-the-art system during its training.

4. Experiments

We ran experiments on the same and different microphone tasks of the NIST 2010 SRE. We used 40K recordings in estimating the parameters of the standard speaker recognition system. These recordings came from the NIST SRE 2004-2006 and Switchboard Phases II and III datasets. We trained standard MFCC-based baseline systems that used either 42 or 60 dimensional features as input for the clean signal. We also simulated a noise condition by adding 4dB additive white Gaussian noise to each recording.

Table 1 reports our experimental results with the standard MFCC-based baseline system. The results reported are in terms of 1000 times the minimum detection cost function (minDCF) as defined in the NIST SRE 2008 (denoted as D08), and the equal-error rate (EER).

As Table 1 reports, the performance in terms of EER does not change much as we change the feature dimensionality. Under the noisy condition, the performance in terms of D08 and EER is about 2.5 times worse for the same microphone task and about 3 times worse for the different microphone task.

4.1. Training a Bottleneck Network

Since our focus in this paper is on the microphone tasks, the NIST SRE 2004-2006 microphone recordings were used in training a bottleneck network. These recordings came from a total of 173 speakers. We included five recordings from each speaker in our validation set and the remaining 4,341 recordings for training.

For the results reported in this paper, the first hidden layer had 1,000 nodes and the third hidden layer had 500 nodes. We report our experimental findings for the bottleneck feature vector sizes of 42, 60, and 100. Our bottleneck network implementation was based on [9].

For input features to the bottleneck network, we extracted MFCCs for all (including speech and non-speech) frames of a given recording. The raw MFCC features are mean- and variance-normalized over a sliding window of 3 seconds. The resulting normalized MFCC features of multiple (for example, 21) consecutive frames are concatenated and used as the input. We used one in every 10 frames to downsample our training data. We had approximately 7 million samples for training and 3 million samples for validation.

Table 1: The speaker recognition performance of a standard MFCC-based system with 42 and 60 dimensional features under clean and noisy conditions.

MFCC	Same mic		Different mic	
Train/Test	D08	EER	D08	EER
Clean/Clean (42)	15.0	2.8	24.8	5.0
Clean/Clean (60)	14.9	3.0	27.5	5.5
Noisy/Noisy (42)	37.3	6.9	74.5	16.5
Noisy/Noisy (60)	36.9	8.1	73.1	16.2

The minDCF performance metric evaluated on the validation data determined the stopping criterion for network training as well as the choice of the network size. The performance was estimated in each training iteration using the following technique: The score generated for a given recording at the output node is taken as the speaker verification score. Therefore, the score generated at the node that corresponds to the correct speaker label is used as a target score and the rest are used as non-target scores. These scores were used to calculate the minDCF and EER within the network without requiring an explicit UBM-GMM-PLDA modelling paradigm. The best validation set peformance was obtained with a configuration where each mini-batch consisted of 50,000 samples and the conjugate gradient algorithm used 25 line search iterations.

4.2. Performance of the Bottleneck Feature-Based System

We evaluated the performance of the bottleneck features within the UBM-GMM-PLDA paradigm. The whitened bottleneck features were used to train a 1024 component GMM-UBM and to generate MAP-adapted speaker models. The GMM mean-based supervectors were used to estimate the PCA, LDA, and WCCN transformations. We first reduced the dimensionality to 800 with a PCA transform. Following this, an LDA transform reduced the dimensionality to 250 for clean conditions and to 150 for noisy conditions.

Our results with the proposed recording-level training criterion are reported in Table 2 in terms of the D08 and EER for three feature dimensions, denoted as 42, 60, and 100. Despite achieving the best results with 100 dimensions, we focus on 42 and 60 dimensional features, which are commonly adopted in state-of-the-art systems.

The D08 and EER of the 42 dimensional bottleneck feature-based system are both about 30% worse relative than the standard MFCC-based system with the same feature vector size in the same microphone task. When the feature vector size is 60, the relative degradation in terms of D08 and EER drops to 27% and 23%, respectively. The performance gap is significantly less for the 4dB AWGN condition. Similar trends are observed for the different microphone task.

Table 2: The speaker recognition performance of a bottleneck feature-based system using recording-level training.

DBN	Same mic		Different mic	
Train/Test	D08	EER	D08	EER
Clean/Clean (42)	21.4	4.0	39.0	7.6
Clean/Clean (60)	20.5	3.9	32.2	6.3
Clean/Clean (100)	19.2	3.7	32.2	6.0
Noisy/Noisy (42)	45.5	8.7	78.5	18.2
Noisy/Noisy (60)	42.8	8.3	76.7	17.4
Noisy/Noisy (100)	38.4	8.1	71.1	16.1

Table 3: The effect of the recording-level and sample-level training criterion on the performance.

DBN	Same mic		Different mic	
Train/Test	D08	EER	D08	EER
fDCF (42D)	25.9	4.7	46.0	9.1
DCF (42D)	21.4	4.0	39.0	7.6

4.3. Effect of the Training Criterion

In this section, we compare the effectiveness of the proposed recording-level training criterion, J_{LLR}, to its sample-level version. We keep the network architectures the same for fair evaluation. The results are reported in Table 3. The results indicate a significant improvement with the use of a recording-level instead of a sample-level training criterion. This gain can be attributed to the use of a training objective that is better aligned with the final performance evaluation metric.

The performance difference between the two training techniques is even more significant on the validation data performance estimated within the network during training: The estimated D08 and EER were 13.3% and 2.2%, respectively, when recording-level J_{LLR} was used. When the sample-level J_{LLR} was adopted, the D08 and EER were 24.8% and 9.0% respectively.

4.4. Effect of the Combination Strategy

We also investigated the effectiveness of using the standard MFCC-based system speaker recognition scores in training a bottleneck network to boost its performance. The results of these experiments are reported in Table 4. For comparison purposes, results for score-level fusion are also provided. The MFCC and bottleneck feature-based systems were combined with a linear weighting as determined on a held-out set. As the results show, the proposed combination strategy, that uses MFCC scores in retraining the network, provided an incremental gain in the D08 (for which the training objective J_{LLR} was tuned for) for the two tasks.

5. Conclusions

This paper proposed an approach to using a bottleneck neural network to provide features to a speaker recognition system. We described a network training technique that optimizes a recording-level criterion rather than a sample-level training objective that exploits long-range information. Our experimental results indicated that a recording-level training criterion provides significant gains over a sample-based criterion.

Table 4: Combination of the bottleneck feature-based system with a standard MFCC-based system.

MFCC+DBN	Same mic		Different mic	
Clean/Clean	D08	EER	D08	EER
Score-level (42D)	13.7	2.4	22.9	4.4
Boosted (42D)	13.4	2.4	21.8	4.1

We furthermore proposed leveraging speaker recognition scores from another system to retrain a well-trained network to focus on remaining errors. Our experiments showed that this method provided additional benefit when bottleneck feature-based system was combined with the MFCC-based baseline.

6. Acknowledgements

The authors thank David Nahamoo, Bhuvana Ramabhadran, and Tara Sainath for their insightful discussions on neural networks.

7. References

[1] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Interspeech*, 2011.

[2] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky, "Probabilistic and bottleneck features for LVCSR of meetings," in *International Conference on Acoustics, Speech, and Signal Processing*, 2007.

[3] F. Grezl and P. Fousek, "Optimizing bottleneck features for LVCSR," in *International Conference on Acoustics, Speech, and Signal Processing*, 2008.

[4] F. Grezl, M. Karafiat, and L. Burget, "Investigation into bottleneck features for meeting speech recognition," in *Annual Conference of the International Speech Communication Association*, 2009.

[5] C. Plahl, R. Schluter, and H. Ney, "Hierarchical bottleneck features for LVCSR," in *Interspeech*, 2010.

[6] Y. Konig, L. Heck, and K. Sonmez, "Nonlinear discriminant feature extraction for robust text-independent speaker recognition," in *RLA2C, ESCA workshop on Speaker Recognition and its Commercial and Forensic Applications*, 1998.

[7] S. Garimella and H. Hermansky, "Factor analysis of mixture of auto-associative neural networks for speaker verification," in *Odyssey*, 2012.

[8] N. Brummer and et al., "Fusion of heterogeneous speaker recognition systems in the STBU submission for the NIST speaker recognition evaluation 2006," *IEEE Transactions On Audio, Speech and Language Processing*, vol. 15, 20067.

[9] G. Hinton, "http://www.cs.toronto.edu/hinton," Accessed April, 2012.