Improvements on Deep Bottleneck Network based I-Vector Representation for Spoken Language Identification

Yan Song[1],*Rui-Lian Cui*[1] *Ian McLoughlin*[2], *Li-Rong Dai*[1]

National Engineering Laboratory of Speech and Language Information Processing
University of Science and Technology of China, China[1]
School of Computing, University of Kent, Medway, UK[2]
{songy, lrdai}@ustc.edu.cn, cuirl@mail.ustc.edu.cn, ivm@kent.ac.uk

Abstract

Recently, the i-vector representation based on deep bottleneck networks (DBN) pre-trained for automatic speech recognition has received significant interest for both speaker verification (SV) and language identification (LID). In particular, a recent unified DBN based i-vector framework, referred to as DBN-pGMM i-vector, has performed well. In this paper, we replace the pGMM with a phonetic mixture of factor analyzers (pMFA), and propose a new DBN-pMFA i-vector. The DBN-pMFA i-vector includes the following improvements: (i) a pMFA model is derived from the DBN, which can jointly perform feature dimension reduction and de-correlation in a single linear transformation, (ii) a shifted DBF, termed SDBF, is proposed to exploit the temporal contextual information, (iii) a senone selection scheme is proposed to improve the i-vector extraction efficienvy. We evaluate the proposed DBN-pMFA i-vector on the most confused six languages selected from NIST LRE 2009. The experimental results demonstrate that DBN-pMFA can consistently outperform the previous DBN based framework [1]. The computational complexity can be significantly reduced by applying a simple senone selection scheme.

1. Introduction

The i-vector representation has achieved state-of-the-art performance for both language identification (LID) and speaker verification (SV) tasks [2, 3]. Generally, the i-vector procedure consists of (1) a front-end feature extraction stage, to extract low-level acoustic features from the given utterance, and (2) a back-end modeling stage, which constructs a low-dimensional compact representation via factor analysis (FA). Conventionally, a diagonal Gaussian mixture model (GMM) is used as the universal background model (UBM) for computing zeroth-order and first-order Baum-Welch statistics.

Recently, several works have used deep neural networks (DNNs) as front-end feature extractors. Given a pre-trained DNN structure, some systems exploit the output from an internal layer. In [4, 5, 6, 7, 8], the i-vector representations based on deep bottleneck features (DBF) have achieved significant performance gains in both clean and noisy conditions. Other works use transformations of the output phonetic posterior probabilities [9, 10]. For example, Diez *et al.* introduced phone log-likelihood ratios (PLLR) [9]. Ma *et al.* used the log of phoneme posteriors combined with PLP features [10]. These works demonstrated competitive performance as stand-alone systems for LID.

Supported by the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing (grant, 2015A15).

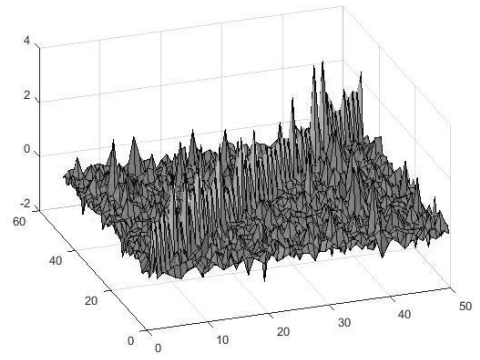

Figure 1: *A 3-D surface plot of the covariance matrix taken from a typical pGMM component, in which significant off-diagonal values indicate the correlation among different feature coefficients.*

In our previous work [1], a unified DBN based i-vector framework has been proposed, which can consistently outperform the GMM-UBM and DNN based i-vector systems. As shown in Fig. 2(a), the front-end DBF is taken from the internal BN layer, similar to [4, 5]. A diagonal phonetic GMM (pGMM), derived from the LID corpus, is used as UBM to calculate sufficient statistics needed for FA. For simplicity we refer to this framework as DBN-pGMM i-vector.

Despite good performance, there are still several issues with the current DBN-pGMM based i-vector framework. Firstly, it does not take the correlation of DBF coefficients into consideration. For a typical mixture of pGMMs, there are several strong non-zero off-diagonal elements in the covariance structure. This is illustrated in Fig.1 which plots a full-covariance pGMM from the DBN. This indicates that a diagonal pGMM may not be optimal for describing the DBF distribution. In [7], it is shown that full-covariance GMM may consistently improve the performance for short duration test conditions. Secondly, it is known that using shifted delta cepstra (SDC) features may lead to significant performance gain over conventional mel-frequency cepstral coefficients (MFCC) and perceptual linear prediction (PLP) feature [11]. It is straightforward to extend the DBF to form a shifted DBF in a similar way. However, the high-

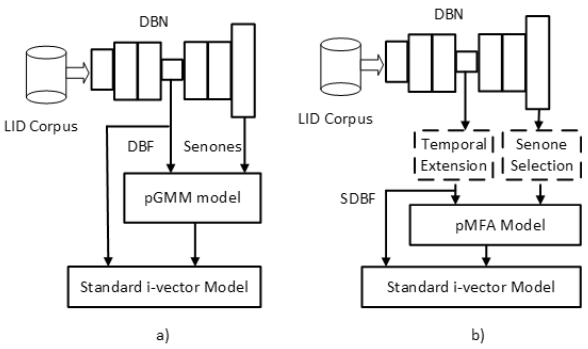

Figure 2: *A flowchart of DBN based i-vector frameworks. a) The existing DBN-pGMM i-vector framework [1]. b) The proposed DBN-pMFA i-vector framework, with several improvements shown in dashed boxes.*

dimensionality of the shifted DBF increases the computational complexity and memory requirements.

Finally, the pGMM is obtained by associating its mixtures to the senones in the DBN output layer. The senones are tied-states with context-dependent phones determined from an ASR decision tree. Generally, large numbers of senones may improve the discriminative capability of the extracted DBF. However, this may increase the size of the derived pGMM, leading to high-dimensionality issues. Furthermore, due to the lack of transcribed data, the DBN is effectively constrained to certain languages (*e.g.*, English and Mandarin). It is still unclear how to construct the LID-related senones for i-vector extraction.

We propose replacing the pGMM with a phonetic mixture of factor analyzers (pMFA) in the DBN based i-vector framework, which we will term DBN-pMFA i-vector. A flowchart of the proposed DBN-pMFA is shown in Fig.2(b). The DBN-pMFA i-vector framework is effectively a two-stage FA process. The pMFA applies first-stage FA on front-end features, which can jointly perform the dimension reduction, de-correlation via a single linear transformation. This can also address the high-dimensionality issue. The i-vector procedure is a second-stage FA, which projects the high-dimensional super-vector into a low-dimensional total variability space. In DBN-pMFA i-vector, a shifted DBF (which we will term SDBF), is further proposed to exploit the temporal contextual information. Finally, a senone selection scheme is proposed according to the collected zeroth-order statistics on the LID corpus, which makes i-vector training and extraction more efficient.

To evaluate the proposed DBN-pMFA based i-vector system, we conducted extensive experiments on the six high-confusable languages from the NIST LRE 2009 evaluations. The experimental results on 30s, 10s and 3s test conditions show the consistent and significant improvements over the previous DBN-based i-vector [1].

The rest of this paper is organized as follows. Section 2 briefly reviews the DBN-based i-vector framework, followed by the introduction of the proposed DBN-pMFA based i-vector in section 3. Section 4 discusses implementation details, including the senone selection scheme and SDBN. Section 5 presents experimental results and analysis, followed by the conclusion and future work in section 6.

2. Review of DBN-pGMM i-vector [1]

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ be an utterance with T speech frames, $\mathbf{x}_t \in \mathcal{R}^d$ is DBF extracted on the t-th frame given a pre-trained DBN for ASR. Let $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_K\}$ be a diagonal pGMM, where $\lambda_k = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ describes the parameters of the k-th mixture. The pGMM is derived from the DBFs and senone posterior on target LID corpus. The zeroth-order and the centralized first-order statistics for k-th mixture, *i.e.* N_k and $\mathbf{F}_k$, are calculated as follows

$$\gamma_{k,t} = P(\lambda_k|\mathbf{x}_t) \quad (1)$$

$$N_k = \sum_{t=1}^{T} \gamma_{k,t} \quad (2)$$

$$\mathbf{F}_k = \sum_{t=1}^{T} \gamma_{k,t}(\mathbf{x}_t - \boldsymbol{\mu}_k) \quad (3)$$

where $\gamma_{k,t}$ is the posterior probability for $\mathbf{x}_t$ on the k-th pGMM component. The super-vector of $\hat{\mathbf{F}}$ is the concatenation of all the centralized first-order statistics $\hat{\mathbf{F}} = [\mathbf{F}_1^T, \ldots, \mathbf{F}_K^T]^T$. By applying factor analysis on the super-vector space, $\hat{\mathbf{F}}$ can be linearly projected into a low-dimensional total variability space as

$$\hat{\mathbf{F}} \rightarrow \mathbf{T}\mathbf{w} \quad (4)$$

where $\mathbf{w}$ is the R-dimensional i-vector with normal distribution $\mathcal{N}(0, \mathbf{I})$, $R << Kd$. The loading matrix $\mathbf{T} \in \mathcal{R}^{Kd \times R}$ is a low-rank rectangular matrix, which can be trained similarly to the eigen-voice method [12].

It is known that the i-vector extraction process is computational expensive [13]. Considering a K-component diagonal GMM and d-dimensional features, the R-dimensional i-vector $\mathbf{w}$ is computed as

$$\mathbf{w} = (\mathbf{I} + \mathbf{T}^T\boldsymbol{\Sigma}^{-1}\mathbf{N}\mathbf{T})^{-1}\mathbf{T}^T\boldsymbol{\Sigma}^{-1}\mathbf{N}\mathbf{F} \quad (5)$$

where $\mathbf{N}$ is a $Kd \times Kd$-dimensional diagonal matrix, with diagonal blocks $N_k\mathbf{I}, k = 1, \ldots, K$. $\boldsymbol{\Sigma}$ is a $Kd \times Kd$-dimensional diagonal covariance matrix. The computational complexity of eqn.(5) is about $O(R^3 + KR^2 + KdR)$. When K and d are large, the computational cost is huge.

Furthermore, existing DBN based i-vector framework does not consider the correlation between coefficients of DBF. In the following sections, we first derive a pMFA model from DBN as UBM for i-vector extraction. Compared to the diagonal pGMM model, a pMFA model is equivalent to a constrained full-covariance GMM model. By selecting appropriate latent factors, an improved i-vector representation can be obtained efficiently.

3. DBN-pMFA based i-vector

3.1. Mixture of Factor Analyzers

As shown in [14], MFA is a directed generative model, which approximates nonlinear manifolds under the local linear assumption. A MFA generally consists of K linear factor analyzers, which can be defined as

$$p(k) = \pi_k, \qquad \sum_{k=1}^{K} \pi_k = 1 \quad (6)$$

$$p(\mathbf{z}|k) = p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I}_{q_k}) \quad (7)$$

$$p(\mathbf{x}|\mathbf{z}, k) = \mathcal{N}(\mathbf{x}|\mathbf{W}_k\mathbf{z} + \boldsymbol{\mu}_k, \boldsymbol{\Psi}_k) \quad (8)$$

where k is the component indicator, and $\mathbf{z} \in \mathcal{R}^{q_k}$ denotes the q_k-dimensional latent variable, $\mathbf{I}_{q_k}$ is a $q_k \times q_k$ matrix. The parameters of k-th factor analyzers include the mixing proportion π_k, a factor loading matrix $\mathbf{W}_k \in \mathcal{R}^{d \times q_k}$, mean vector $\boldsymbol{\mu}_k$, and a diagonal matrix $\boldsymbol{\Psi}_k \in \mathcal{R}^{d \times d}$ that represents independent noise variance for each of the variables. In our implementation, we assume isotropic noise variance for simplicity, *i.e.* $\boldsymbol{\Psi}_k = \sigma_k^2 \mathbf{I}$, where σ_k^2 denotes the average noise power. This is actually a mixture of probabilistic principal component analyzers (PPCA).

By integrating out the latent factor $\mathbf{z}$, MFA is equivalent to a GMM model $\boldsymbol{\lambda}_G = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

$$p(\mathbf{x}|k) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z}, k) p(\mathbf{z}|k) d\mathbf{z} = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (9)$$

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (10)$$

where $\boldsymbol{\Sigma}_k = \mathbf{W}_k \mathbf{W}_k^T + \boldsymbol{\Psi}_k$ is the constrained full-covariance matrix of the k-th Gaussian component.

3.2. From full covariance pGMM to pMFA

The pGMM with full-covariance matrix $\boldsymbol{\Sigma}_k, k = 1, \ldots, K$ can be calculated from the DBN, using the DBFs $\mathbf{x}_t$ and corresponding senone posteriors $\gamma_{k,t}$, $t = 1, \ldots, T$.

$$N_k = \sum_{t=1}^{T} \gamma_{k,t} \quad (11)$$

$$\pi_k = \frac{N_k}{\sum_{j=1}^{K} N_j} \quad (12)$$

$$\boldsymbol{\mu_k} = \frac{1}{N_k} \sum_{t=1}^{T} \gamma_{k,t} \mathbf{x}_t \quad (13)$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{t}^{T} \gamma_{k,t} \mathbf{x}_t \mathbf{x}_t^T - \boldsymbol{\mu_k} \boldsymbol{\mu_k^T} \quad (14)$$

According to the equivalence between MFA and full-covariance GMM, the mixing proportion π_k and mean $\boldsymbol{\mu}_k$ can be determined using eqns. (12), (13).

We will derive the noise variance matrix Ψ_k and factor loading matrix $\mathbf{W}_k$ as follows. First, the noise subspace of the k-th factor analyzer can be determined by the value of q_k, which defines the number of principal axes to be selected. And the average noise power σ_k^2 can be obtained via

$$\sigma_k^2 = \frac{1}{d - q_k} \sum_{i=q_k+1}^{d} \lambda_{k,i} \quad (15)$$

where $\lambda_{k,i}, i = q_k + 1, \ldots, d$ are the smallest eigenvalues of the covariance matrix $\boldsymbol{\Sigma}_k$. Then the factor loading matrix can be obtained via maximum likelihood estimation as shown in [15].

$$\mathbf{W}_k = \mathbf{U}_{q_k} (\boldsymbol{\Lambda}_{q_k} - \sigma_k^2 \mathbf{I})^{\frac{1}{2}} \quad (16)$$

where $\mathbf{U}_{q_k}$ is the eigenvector matrix corresponding to the largest q_k eigenvalues, and Λ_{q_k} is the diagonal matrix of eigenvalues.

3.3. pMFA based i-vector

Given the pMFA model, the zeroth-oder and first-order Baum-Welch statistics are needed for i-vector extraction, as shown in Section 2. The zeroth-order statistics can be directly computed using eqn. (2) according to the senone posterior $\gamma_{k,t}$ obtained from the DBN. And for first-order statistics, the posterior mean of latent factor $\mathbf{z}_t$ of the k-th component is used instead of $\mathbf{x}_t$, that is

$$E\{\mathbf{z}_t|\mathbf{x}_t, k\} = \mathbf{A}_k^T (\mathbf{x}_t - \boldsymbol{\mu}_k) \stackrel{\triangle}{=} \mathbf{z}_{t,k} \quad (17)$$

where

$$\mathbf{A}_k^T = \boldsymbol{\Lambda}_{q_k}^{-1} (\boldsymbol{\Lambda}_{\boldsymbol{q_k}} - \sigma_k^2 \mathbf{I})^{\frac{T}{2}} \mathbf{U}_{q_k}^T \quad (18)$$

As shown in [16], eqn. (17) can be considered as a feature transformation process of $\mathbf{x}_t$, which can perform feature dimension reduction, de-correlation, and enhancement simultaneously. The first-order statistics $\mathbf{F}_k^{'}$ is extracted in terms of $\mathbf{z}_{t,k}$

$$\begin{aligned} \mathbf{F}_k^{'} &= \sum_{t=1}^{T} \gamma_{k,t} \mathbf{z}_{t,k} \\ &= \sum_{t=1}^{T} \gamma_{k,t} \mathbf{A}_k^T \big(\mathbf{x}_t - \boldsymbol{\mu}_k\big) \\ &= \mathbf{A}_k^T \big(\mathbf{F}_k - N_k \boldsymbol{\mu_k}\big) \end{aligned} \quad (19)$$

The super-vector $\mathbf{F}^{'}$ is the concatenation of the centralized first-order statistics $\mathbf{F}^{'} = [\mathbf{F}_1^{'T}, \ldots, \mathbf{F}_K^{'T}]^T$. The i-vector training and extraction can then be processed as illustrated in Section 2.

It is worth noting that the dimension of super-vector $\mathbf{F}^{'}$ is $K \times q_k$. When $q_k << d$, the computational complexity and memory requirement of i-vector extraction will be greatly reduced.

4. Implementation details

In this section, we will describe the details of DBN-pMFA i-vector framework, including the DBN structure, the SDBF configuration, the senone selection scheme, and the post-processing after i-vector extraction for LID.

DBN structure. The DBN structure we used has 7 layers comprising 1 input layer, 5 hidden layers, and 1 output layer, which is configured as: $21 \times 48 - 2048 - 2048 - 50 - 2048 - 2048 - 1536$. For each speech frame, a 48 dimensional feature is first extracted, which comprises 39-dimensional dimensional PLP +ΔPLP+$\Delta\Delta$PLP and 9 dimensional pitch features, and their 1st and 2nd derivatives. Then a $21 \times 48 = 1008$ input feature is obtained by concatenating 21 frames centered around the current one. The output layer contains 1536 senones automatically determined by a decision tree using maximum likelihood [17]. The DBN is trained on about 300 hours of Switchboard corpus, using the Kaldi speech recognition toolkit [18]. For each utterance in the target LID corpus, we extract the T DBFs x_t and corresponding senone posterior probability $\gamma_{k,t}$, $t = 1, \ldots, T, k = 1, \ldots, 1536$ from the pre-trained DBN.

In related work [1, 19], the DBFs extracted from different DBN structures were evaluated, indicating that better LID performance can be obtained with larger number of output senones.

Temporal extension. The temporal extension of DBF, termed SDBF is obtained similarly to SDCs, and is configured using parameters $N - d - P - k$ defined in [11]. In experiments, we evaluate the 200 dimensional SDBF, configured as $50-1-3-3$.

Table 1: *The target languages along with available training data channel sources and the number of evaluation segments.*

Language	Training data source	Number of tests		
		30s	10s	3s
Dari	VOA	375	392	400
Farsi	CTS	389	383	387
Russian	CTS&VOA	510	486	490
Ukrainian	VOA	368	405	391
Hindi	CTS&VOA	662	618	642
Urdu	CTS&VOA	367	385	381

Table 2: *Evaluation of DBN-pGMM i-vector and DBN-pMFA i-vector with on 50 dimensional DBF on the 6 high-confusable languages from LRE09. The performance is evaluted in terms of EER (%)*

		30s	10s	3s
DBN-pGMM-DBF(1536)		6.56	8.43	15.20
DBN-pMFA-DBF	q_k = 45	5.68	7.91	15.35
	q_k = 40	**5.65**	7.87	**14.57**
	q_k = 35	5.69	**7.84**	14.68
	q_k = 30	5.84	8.09	15.50

Senone Selection. We select the LID-related senones based on the zeroth-order statistics $\mathcal{N}_k, k = 1, \dots 1536$ computed from senone posteriors on the LID corpus. We sort $\mathcal{N}_k$ in descending order, and select the first $N, N < 1536$ senones as the most related components for the target LID task. The resulting pMFA is derived from the full-covariance pGMM as illustrated in Section 3.

Post processing. After the i-vector representation, two intersession compensation techniques are applied. The first step is within-class covariance normalization (WCCN), which normalizes the i-vector with the inverse of the within-class covariance. The second step is linear discriminative analysis (LDA), a popular dimension reduction techniques for removing noise. After these two steps, the language model can be represented as the center of the corresponding i-vectors. Given a test utterance, the confidence score is calculated as the cosine distance from each model.

5. Experiments

We conduct extensive experiments on the selected six most confused languages from NIST LRE2009, *i.e.* Dari, Farsi, Russian, Ukrainian, Hindi and Urdu. The training dataset is mainly collected from the conversational telephone speech (CTS) and Voice of America (VOA) radio broadcasts. The evaluations are performed on closed-set test, which comprise 2671, 2669 and 2691 evaluation segments for 30s, 10s and 3s test conditions respectively. The performance is measured using equal error rate (EER). Table 1 summarizes the training and evaluation data.

Three experiments are conducted to evaluate the effectiveness and efficiency of the proposed DBN-pMFA i-vector. The DBN-pMFA i-vector [1] is taken as the baseline system In all experiments, we fix the i-vector dimension to be 400 for fair comparison.

5.1. Evaluation of DBN-pMFA i-vector against DBN-pGMM i-vector using DBF

The experiments in this section compare the proposed DBN-pMFA i-vector against the DBN-pGMM i-vector with DBF [1]. In DBN-pGMM, the dimensionality of the front-end DBF is $d = 50$, and the derived pGMM is used as UBM for collecting the required sufficient statistics The number of UBM mixtures is determined by the senones in the pre-trained DBN, *i.e.* $K = 1536$. In DBN-pMFA, the pGMM is replaced by a pMFA model. The front-end feature is the mean of latent factors with q_k dimensions from the k-th pMFA component. We evaluated the performance of the i-vector with $q_k = 45, 40, 35, 30 and k = 1, \dots, K$, yielding compression ratios of $0.9, 0.8, 0.7, 0.6$, respectively. For simplicity, we heuristically set the same q_k for all k.

The results are shown in Table 2. For the 30s and 10s test conditions, DBN-pMFA-DBF consistently outperforms the DBN-pGMM-DBF. For the 3s test condition, the performance of DBN-pMFA-DBF is comparable. When q_k is selected appropriately, such as $q_k = 40, 45$, a slight improvement can be achieved. When $q_k = 30$, the EERs are 5.84%, 8.09% and 15.5% for 30s, 10s and 3s test conditions respectively. The performance is nearly saturated when we increase q_k.

It is interesting to see that when $q_k = 45$ (0.9 of DBF dimensionality), the LID performance slightly degrades. This indicates that when the dimensionality of i-vector R is fixed, there exists a trade-off between the first-stage and second-stage FA in the DBN-pMFA i-vector. If q_k is small, some language dependent information is discarded in pMFA, which degrades the performance. On the contrary, when q_k is large, the information loss occurs on the second-stage FA, which projects the $K \times q_k$-dimensional super-vector to a R-dimensional total variability space.

Specifically, when $q_k = 40$, the performance of DBN-pMFA for 30s, 10s and 3s test conditions are 5.65%, 7.87% and 14.57%, with relative improvements 16%, 7%, and 4% respectively. And according to eqn. (5), the computational complexity of i-vector extraction is slightly reduced due to the smaller q_k.

5.2. Evaluation of DBN-based i-vector using SDBF

The experiments in this section evaluate the DBN-based i-vector system using front-end features with temporal contextual information, *i.e.* SDBF. We configured SDBF as $50-1-3-3$, parameterized by $(N-d-p-k)$ as described in [11]. In DBN-pGMM-SDBF, the dimensionality of front-end feature SDBF is $d = 200$, and the number of pGMM mixtures is $K = 1536$. Similar as section 5.1, the performance of DBN-pMFA-SDBF is evaluated with $q_k = 160, 140, 120, 100, 80, 60, 40$.

The results are shown in Table. 3. Firstly, we can see that DBN-pGMM-SDBF consistently outperforms DBN-pGMM-DBF in 30s, 10s and 3s test conditions. This can be attributed to the introduction of temporal contextual information in S-DBF. However, the computational complexity of DBN-pGMM-SDBF i-vector is much higher with larger d.

The comparison between DBN-pMFA-SDBF and DBN-pGMM-SDBF has the similar conclusion as section 5.1. In most cases, the DBN-pMFA-SDBF outperforms DBN-pGMM-SDBF, except that for 3s test conditions. It is worth noting that when $q_k = 40$, the performance is still better than DBN-pGMM-DBF i-vector. This validates the effectiveness of incorporating the temporal contextual information.

However, even with the reduced dimension, the i-vector extraction in DBN-pMFA-SDBF is computational expensive.

We further propose senone selection scheme, which aims to improve the efficiency of DBN-pMFA-SDBF, with slight performance loss.

Table 3: *Evaluation of DBN-pGMM i-vector and DBN-pMFA i-vector based on SDBF configured as* $50-1-3-3$*, on the six high-confusable languages from LRE09. The performance is evaluated in terms of EER (%)*

		30s	10s	3s
DBN-pGMM-DBF(1536)		6.56	8.43	15.20
DBN-pGMM-SDBF(1536)		6.17	7.86	14.64
DBN-pMFA-SDBF	q_k = 160	5.57	7.68	14.90
	q_k = 140	5.54	7.57	14.79
	q_k = 120	5.68	7.61	14.57
	q_k = 100	5.35	7.41	**14.57**
	q_k = 80	5.47	7.79	15.01
	q_k = 60	**5.32**	**7.38**	14.83
	q_k = 40	5.35	7.73	14.82

5.3. Evaluation of senone selection

As aforementioned, the DBN is pre-trained for ASR. The senones are generally determined by the decision tree of certain language, *i.e.* English or Mandarin. That is to say, the senones are not directly LID-related. In the DBN-based i-vector framework, the number of mixtures is generally determined by the number of senones. In experiments, we want to study whether all the senones in certain languages is informational for LID. The evaluations of senone selection schemes are conducted based on the previous optimal experimental setting, *i.e.* DBN-pMFA-SDBF60, the DBN-pMFA with $q_k = 60$ The number of senones to be selected is varied as $K = 600, 800, 100, 1200$.

The results are shown in Table. 4. It is shown that according to a simple criterion in Section 4, the DBN-pMFA-SDBF60 with $K = 600$ can achieve the comparable performance, with relative improvements over the baseline DBN-pGMM-DBF of 22% and 8% for 30s and 10s test conditions. Increasing K does not effectively alter the LID performance, except for the 3s conditions.

We also report the average computational complexity of i-vector extraction in terms of seconds per 100 samples on a server with Intel i7-2600K, CPU and 32G memory. For DBN-pMFA-SDBF with $K = 1536$ mixtures, the average time for extracting 100 400-dimensional i-vectors is about 37.5 seconds. For DBN-pMFA-SDBF60 with $K = 600$ mixtures, the average time is about 10.6 seconds. This is even more efficient than the baseline DBN-pGMM-DBF, in which a 15.4 second average is obtained.

6. Conclusion and Future work

This paper presented an improved DBN-based i-vector framework which we term the DBN-pMFA i-vector system, in which the pGMM is replaced by pMFA in the i-vector procedure. DBN-pMFA is actually a two-stage FA. The pMFA is performed on the front-end feature space, while i-vectors are the second-stage FA performed on the super-vector space. Compared to the pGMM model, the pMFA model is shown to be equivalent to a constrained full-covariance GMM, which can better describe the correlation among the DBF coefficients.

Table 4: *Evaluation of senone selection scheme using DBN-pMFA-SDBF60. The performance is evaluated in terms of EER (%).*

		30s	10s	3s
DBN-pGMM-DBF(1536)		6.56	8.43	15.20
DBN-pGMM-SDBF(1536)		6.17	7.86	14.64
DBN-pMFA-SDBF60(1536)		**5.32**	**7.38**	**14.83**
Mixtures	K = 600	5.35	7.80	15.20
	K = 800	5.57	7.98	15.53
	K = 1000	5.61	7.83	14.83
	K = 1200	5.31	7.65	15.16

Furthermore, pMFA can perform dimension reduction and decorrelation in a single linear transformation. This makes the application of high-dimension SDBF (a shifted version of DBF) possible, which can effectively outperform the existing methods. Finally, the LID-related senones are selected for deriving the pMFA model, which can improve the efficiency and effectiveness of the DBN-based i-vector for LID.

The experimental results on selected highly confusable languages in NIST LID 09 show that (i) The DBN-pMFA i-vector can consistently outperform the previous DBN-pGMM except for the 3s test condition [1], (ii) The SDBF can achieve more significant performance gains, which validates that the temporal contextual information may be beneficial to LID, and (iii) We can select partial LID-related senones to compute the i-vector efficiently with acceptable performance loss.

In the future, we will evaluate the proposed DBN-pMFA i-vector framework on the full benchmark NIST LRE09 evaluations. Furthermore, in current DBN-pMFA i-vector, the two-stage FA is performed separately. We will try to find a joint optimization framework.

7. References

[1] Yan Song, Xinhai Hong, Bing Jiang, Ruilian Cui, Ian McLoughlin, and Lirong Dai, "Deep bottleneck network based i-vector representation for language identification," in *Proc. of Interspeech*, 2015.

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

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

[4] Bing Jiang, Yan Song, Si Wei, Ian McloughLin, and Li-Rong Dai, "Task-aware deep bottleneck features for spoken language identification," in *Proc. of Interspeech 2014*, 2014, pp. 3012–3016.

[5] Yuning Song, Bo Jiang, YeBo Bao, Shaojun Wei, and Li-Rong Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013.

[6] Pavel Matejka, Le Zhang, Tim Ng1, Sri Harish Mallidi, Ondrej Glembek, Jeff Ma, and Bing Zhang, "Neural network bottleneck features for language identification," in *Proc. of Odyssey*, 2014.

[7] Radek Fer, Pavel Matejka, Frantisek Grezl, Oldrich Plchot, and Jan Honza Cernocky, "Multilingual bottleneck features for language recognition," in *Proc. of Interspeech*, 2015.

[8] Wang Geng, Jie Li, Shanshan Zhang, Xinyuan Cai, and Bo Xu, "Multilingual tandem bottleneck feature for language identification," in *Proc. of Interspeech*, 2015.

[9] Mireia Diez, Amparo Varona, Mikel Penagarikano, L Rodriguez-Fuentes, and Germán Bordel, "Dimensionality reduction of phone log-likelihood ratio features for spoken language recognition," in *Proc. of InterSpeech*, 2013.

[10] Jeff Ma, Bing Zhang, Spyros Matsoukas, Sri Harish Mallidi, Feipeng Li, and Hynek Hermansky, "Improvements in language identification on the rats noisy speech corpus," in *Proc. of Interspeech*, 2013.

[11] M. A. Kohler and M. Kennedy, "Language identification using shifted delta cepstra," in *Proc. of the IEEE International Midwest Symposium on Circuits and Systems*, 2002, pp. 69–72.

[12] Patrick Kenny, Gilles Boulianne, and Pierre Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Trans Speech Audio Process*, vol. 13, no. 3, pp. 345–354, 2005.

[13] O. Glembek, L. Burget, P. Matejka, M. Karafiat, and P. Kenny, "Simplification and optimization of i-vector extraction," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2011, pp. 4516–4519.

[14] Z Ghahramani and G. E. Hinton, "The em algorithm for mixtures of factor analyzers," in *Technical Report CRG-TR-96-1, University of Toronto, 1996*.

[15] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," pp. 443–482, 1999.

[16] Taufiq Hasan and John H. L. Hansen, "Acoustic factor analysis for robust speaker verification," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 21, no. 4, pp. 842–853, 2013.

[17] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proc. of HLT94*, 1994, p. 307312.

[18] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlcek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *Proc. of IEEE ASRU 2011*, 2011.

[19] Ma Jin, Yan Song, Ian Mcloughlin, Li-Rong Dai, and Zhong-Fu Ye, "Lid-senone extraction via deep neural networks for end-to-end language identification," in *Proc. of Odyssey 2016 (Accepted)*.