LEARNING FACTORIZED FEATURE TRANSFORMS FOR SPEAKER NORMALIZATION

Lahiru Samarakoon, Khe Chai Sim

School of Computing, National University of Singapore, Singapore

lahiruts@comp.nus.edu.sg, simkc@comp.nus.edu.sg

ABSTRACT

This paper proposes an approach to improve automatic speech recognition (ASR) by normalizing the speaker variability of a well trained Deep Neural Network (DNN) acoustic model using i-vectors. Our approach learns a speaker dependent transformation of the acoustic features combined with the standard speaker dependent bias, to minimize the mismatch due to the inter-speaker variability. Speaker normalization experiments on the Aurora 4 task show 10.9% relative improvement over the baseline. Moreover, the proposed approach reported 4.5% relative improvement over the standard i-vector based method where only a speaker dependent bias is used. Furthermore, we report an analysis to compare our approach with the Constrained Maximum Likelihood Linear Regression (CMLLR) method.

Index Terms— Automatic speech recognition, deep neural networks, speaker normalization.

1. INTRODUCTION

Recently, Deep Neural Network (DNN) based acoustic modeling has achieved state-of-the-art performance in ASR systems in comparison to the conventional Gaussian Mixture Model (GMM) based systems [1]. Increased computational power and the utilization of the Graphical Processing Units (GPUs) in computations have made the training of these complex models affordable. Moreover, advances in machine learning approaches in DNN training have contributed to the increased performance. DNN-HMM systems surpass the conventional GMM-HMM systems by using the superior representation learning power of the DNNs to model senone log-likelihoods, combined with the sequential modeling capability of HMMs to model speech signals.

DNNs, like all other machine learning techniques, are susceptible to performance degradation due to mismatch between the training and testing conditions. Adaptation techniques change the model to match the testing condition or change the inputs to match the model. In ASR, speaker adaptation techniques are used to optimize the performance by minimizing the training-testing mismatch introduced by the speaker variability. The two most successful ways of adapting a GMM-HMM model are to use the maximum a posteriori (MAP) [2] and maximum likelihood linear regression (MLLR) [3] techniques. In MAP, instead of using maximum likelihood for parameter estimation, model parameters are re-estimated by maximizing the posterior probability. In MLLR, a linear transformation of the model parameters are estimated to construct the adapted model. It is possible to take advantage of GMM-HMM adaptation techniques with tandem systems [4, 5] in which a DNN is trained to extract bottleneck features for a GMM-HMM system.

The adaptation techniques developed for generative GMMs cannot be directly utilized for discriminative DNNs. In addition, due to the large number of parameters in DNN-HMM systems, techniques developed for artificial neural network (ANN)-HMM hybrid systems [6] are prone to over-fitting when only a small amount of adaptation data is available. However, DNN adaptation is important as it reduces the error rates significantly [7–10]. A good adaptation technique should prevent over-fitting to the adaptation data. This is achieved by finding a compact representation of the model parameters or using a regularization based method to perform the adaptation conservatively. Moreover, it is desirable to perform adaptation in an unsupervised fashion, which is more realistic.

In this paper, we propose a modified DNN structure which is speaker adaptively trained using the i-vectors [11, 12]. The i-vectors can be considered as low dimensional representations of the speaker characteristics. In our method, we are creating a new subspace that learns a feature transformation based on i-vectors. A DNN is capable of using this extra information about the speakers to perform implicit speaker normalization.

The rest of the paper is organized as follows. In Section 2, a brief review of the DNN adaptation techniques is given. Section 3 describes our approach based on learning feature transformations. In Section 4, a comparison between our method and the CMLLR estimation is given based on least squares approximation. Experimental results are reported in Section 5 and we conclude our work in Section 6.

2. DNN ADAPTATION

DNN adaptation techniques can be categorized into three classes: linear transforms, regularization methods, and sub-

space methods.

Linear Transformation-based methods augment the original DNN model with a linear layer. Usually, the linear layer is initialized with an identity matrix and zero biases and is updated with the back-propagation (BP) algorithm using the adaptation data while keeping the weights of the original DNN fixed. In linear input network (LIN) [13, 14] and feature discriminative linear regression (fDLR) [15], a linear layer is inserted between the input layer and the first hidden layer. The intuition is similar to fMLLR [3], where speaker dependent (SD) features are linearly transformed to match the speaker independent (SI) model. When the linear transformation is applied to the softmax layer, the adaptation technique is known as linear output network (LON) [14]. The idea is to transform the last hidden layer's SD feature representation to match the average speaker. Depending on the number of output neurons, it is possible to apply the transformation before or after the softmax layer weights. When the linear transformation is applied to the hidden layers, it is known as a linear hidden network (LHN) [16].

The adaptation of all the parameters is more powerful and more effective than the linear transformations. However, this may lead to over-fitting since the amount of adaptation data is limited. Conservative training methods address this issue by adding a regularization term to the adaptation criterion. In [7], a KL divergence based method is used to force the distribution of the adapted model to be closer to that of the SI model. The estimated distribution is a linear interpolation between the target distribution (derived using alignments) and the distribution of the SI model. Another popular approach is the L_2 regularization [17], which aims to keep the parameters of the adapted model closer to that of the SI model. However, speaker personalization of all the parameters increases storage costs, which necessitates the employment of techniques that reduce the per-speaker footprint [18]. Therefore, some approaches perform the adaptation on a subset of parameters, including the last hidden layer [19], output layer biases [20], or more active hidden units of the network [19]. Another effective model adaptation technique is known as learning hidden unit contributions (LHUC) [21, 22], which learns speaker dependent hidden unit contributions during adaptation.

It is also possible to find a speaker subspace and perform the adaptation as a point in the subspace. In [23], principal component analysis (PCA) is performed on a set of adaptation matrices to get eigenvectors. Transformations for test speakers can then be estimated as a linear combination of these eigenvectors. Coefficients for each speaker are estimated using the BP procedure. This method can also be used with the LIN, LHN and LON techniques [14]. In addition, recently, cluster adaptive training (CAT) has been applied for speaker normalization [24, 25]. In CAT DNN approaches, a set of bases are estimated during training and followed by an interpolation vector estimation to combine the bases during testing. Another popular subspace method is to feed the features for speaker variability with acoustic features. These methods are discussed in Section 3 along with comparisons to the techniques we propose in this paper.

3. PROPOSED METHOD

A DNN can be viewed as a model that learns a feature representation as well as a classifier. Each hidden layer learns a more abstract representation ($\mathbf{h}_l$) from the lower layer's representation ($\mathbf{h}_{l-1}$), which can be shown as:

$$\mathbf{h}_l = \sigma(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l), \tag{1}$$

where σ is the sigmoid activation function. $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weight matrix and the bias vector for layer l, respectively.

In the proposed method, we incorporate speaker information in addition to the standard acoustic features during training. This approach is known as speaker-aware training.

3.1. Speaker-aware Training (SaT)

The intuition behind SaT is that a DNN is capable of exploiting the supplementary information about speakers to adjust the model parameters for speaker normalization. The first step in SaT is the speaker information estimation where techniques like i-vectors [8, 9, 26] and bottleneck features [27] are commonly used. In this paper, i-vectors are used as the speaker information; however it is also possible to replace the i-vectors with the bottleneck features.

The simplest approach in SaT is to concatenate the acoustic features with the i-vector of the speakers before DNN training. In that case, speaker information can be considered as a bias as given below.

$$\mathbf{h}_l = \sigma(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l^s), \tag{2}$$

where $\mathbf{b}_l^s$, the SD bias for layer l, is given by

$$\mathbf{b}_l^s = \mathbf{U}_l \mathbf{v}^{(s)} + \mathbf{b}_l, \tag{3}$$

$\mathbf{v}^{(s)}$ is the speaker representation (i-vector) and $\mathbf{U}_l$ is the speaker representation transformation weight matrix for layer l, respectively.

In our approach, in addition to the SD bias we propose to learn a SD feature transformation based on i-vectors as given below.

$$\mathbf{h}_l = \sigma(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{U}_1 \mathbf{D}^{(s)} \mathbf{U}_2 \mathbf{h}_{l-1} + \mathbf{b}_l^s), \tag{4}$$

where $\mathbf{D}^{(s)} = diag(\mathbf{v}^{(s)})$, and the $\mathbf{U}_1$ and $\mathbf{U}_2$ are weight matrices for SD transformation.

Formulation of our approach is similar to that of the CMLLR [28] estimation for features (also known as fMLLR). In fMLLR, SD transformation ($\mathbf{A}^{(s)}$) as well as a SD bias ($\mathbf{b}^{(s)}$) is estimated using Maximum Likihood criterion (equation 5). In the next section, a comparison of the modeling capacity of

our approach to fMLLR is given using a least square approximation.

$$\hat{\mathbf{x}} = \mathbf{A}^{(s)}x + \mathbf{b}^{(s)} \tag{5}$$

3.2. Training Configuration

All the models with speaker features are trained using a warm-start configuration to reduce the training time. In warm start, the weights of the initial DNN are directly used and the weights for the additional connections are randomly initialized. Then, the newly added random weights are fine-tuned while keeping the rest of the weights fixed.

4. COMPARISON WITH FMLLR

The fMLLR transformation is a speaker-specific linear transform and DNNs are capable of modeling highly non-linear functions. However, DNNs are not modeling speaker normalizing transformations that are similar to fMLLR during training. The significant differences between the performances of DNNs trained on features before and after the fMLLR transforms are evident for that. Furthermore, the standard i-vector based technique with the SD bias is not as robust as the fMLLR technique for speaker normalization. Therefore, in this section, we present a least square approximation method to compare the fMLLR technique with our proposed method.

The features are normalized before feeding into the DNN training. Therefore, in our analysis, the normalization is taken into consideration. In our experiments, the fMLLR feature transforms are estimated on top of the Linear Discriminant Analysis (LDA) features. The normalized fMLLR and LDA features are given in equations 6 and 7 respectively.

$$\mathbf{y}_1(t) = \mathbf{N}_1(\mathbf{A}^{(s)}\mathbf{x}(t) + \mathbf{b}^{(s)} + \mathbf{k}_1). \tag{6}$$

$$\mathbf{y}_2(t) = \mathbf{N}_2(\mathbf{x}(t) + \mathbf{k}_2). \tag{7}$$

The difference between fMLLR and LDA features is given by:

$$\begin{aligned}\mathbf{C}(t)^{(s)} &= \mathbf{y}_2(t) - \mathbf{y}_1(t)\\ &= (\mathbf{N}_2 - \mathbf{N}_1\mathbf{A}^{(s)})\mathbf{x}(t)\\ &+ \mathbf{N}_2\mathbf{k}_2 - \mathbf{N}_1(\mathbf{b}^{(s)} + \mathbf{k}_1)\\ &= \mathbf{P}^{(s)}\mathbf{x}(t) + \mathbf{r}^{(s)}.\end{aligned} \tag{8}$$

Using our model (4) to reduce this difference

$$\mathbf{e}(t)^{(s)} = \mathbf{C}(t)^{(s)} + \mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x}(t) + \mathbf{U}_3\mathbf{v}^{(s)}. \tag{9}$$

Then, the total error is calculated for all the speakers (s) and for all the frames (t) of that speaker. The formula can be calculated as given in equation 10.

$$Q = \sum_{s,t} \mathbf{e}(t)^{(s)\top}\mathbf{e}(t)^{(s)}. \tag{10}$$

In order to calculate this error, we need to estimate $\mathbf{U}_1$, $\mathbf{U}_2$ and $\mathbf{U}_3$ parameters. We do these estimations iteratively. First, initialize $\mathbf{U}_1 = \mathbf{U}_2 = \mathbf{0}$ and estimate $\mathbf{U}_3$. Then $\mathbf{U}_2$ is estimated by assigning $\mathbf{U}_1 = \mathbf{U}_3$. Next, $\mathbf{U}_1$ is estimated. After that parameters are estimated iteratively one by one. Therefore, it is important to formulate the equation in a manner where it is not necessary to go through the data at each iteration. The statistics should be collected at speaker-level. We have formulated estimation equations such that $(\sum_t \mathbf{x}(t))$ and $(\sum_t \mathbf{x}(t)\mathbf{x}(t)^\top)$ statistics per speaker are sufficient. The rest of this section details the estimation formulas for $\mathbf{U}_1$, $\mathbf{U}_2$ and $\mathbf{U}_3$.

4.1. Estimation of $\mathbf{U}_1$

The $\mathbf{U}_1$ that minimizes Q can be estimated as given below:

$$\begin{aligned}\frac{\partial Q}{\partial \mathbf{U}_1} &\Rightarrow \sum_{s,t} \mathbf{e}(t)^{(s)}(\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x})^\top = 0,\\ \mathbf{U}_1 &= -\mathbf{K}_1\mathbf{G}_1^{-1},\end{aligned}$$

where $\mathbf{K}_1 = (\sum_{s,t}(\mathbf{C}(t)^{(s)} + \mathbf{U}_3\mathbf{v}^{(s)})\mathbf{x}^\top\mathbf{U}_2^\top\mathbf{D}^{(s)})$ and $\mathbf{G}_1 = (\sum_{s,t}\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x}\mathbf{x}^\top\mathbf{U}_2^\top\mathbf{D}^{(s)})$.

$\mathbf{K}_1$ and $\mathbf{G}_1$ can be simplified to:

$$\begin{aligned}\mathbf{K}_1 &= \sum_s(\mathbf{P}^{(s)}(\sum_t \mathbf{x}\mathbf{x}^\top) + \mathbf{r}^{(s)}(\sum_t \mathbf{x}^\top)\\ &\quad + \mathbf{U}_3\mathbf{v}^{(s)}(\sum_t \mathbf{x}^\top))\mathbf{U}_2^\top\mathbf{D}^{(s)}.\\ \mathbf{G}_1 &= \sum_s \mathbf{D}^{(s)}\mathbf{U}_2(\sum_t \mathbf{x}\mathbf{x}^\top)\mathbf{U}_2^\top\mathbf{D}^{(s)}.\end{aligned}$$

4.2. Estimation of $\mathbf{U}_2$

The $\mathbf{U}_2$ that minimizes Q can be estimated by taking $\frac{\partial Q}{\partial \mathbf{U}_2} = 0$:

$$\begin{aligned}\frac{\partial Q}{\partial \mathbf{U}_2} &\Rightarrow \sum_{s,t}(\mathbf{x}\otimes\mathbf{D}^{(s)}\mathbf{U}_1^\top)\mathbf{e}(t)^{(s)} = 0,\\ \sum_{s,t}\mathbf{E}\mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x} &= -\sum_{s,t}\mathbf{E}\mathbf{F},\end{aligned}$$

where $\otimes$ is the Kronecker product, $\mathbf{E} = \mathbf{x}\otimes\mathbf{D}^{(s)}\mathbf{U}_1^\top$ and $\mathbf{F} = \mathbf{C}(t)^{(s)} + \mathbf{U}_3\mathbf{v}^{(s)}$.

$$\begin{aligned}\sum_{s,t}\mathrm{Vec}(\mathbf{E}\mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x}) &= -\sum_{s,t}\mathbf{E}\mathbf{F},\\ \sum_{s,t}(\mathbf{x}^\top\otimes\mathbf{E}\mathbf{U}_1\mathbf{D}^{(s)})\,\mathrm{Vec}(\mathbf{U}_2) &= -\sum_{s,t}\mathbf{E}\mathbf{F},\end{aligned}$$

where $\mathrm{Vec}()$ is the vectorization operator. Then, the expression for $\mathbf{U}_2$ is formulated as below:

$$\mathrm{Vec}(\mathbf{U}_2) = -(\sum_{s,t}(\mathbf{x}^\top \otimes \mathbf{E}\mathbf{U}_1\mathbf{D}^{(s)}))^{-1}(\sum_{s,t}\mathbf{E}\mathbf{F}),$$
$$\mathrm{Vec}(\mathbf{U}_2) = -\mathbf{G}_2^{-1}\mathbf{K}_2.$$

$\mathbf{G}_2$ can be estimated as below:

$$\mathbf{G}_2 = \sum_{s,t}(\mathbf{x}^\top \otimes \mathbf{x} \otimes \mathbf{D}^{(s)}\mathbf{U}_1^\top)\mathbf{U}_1\mathbf{D}^{(s)},$$
$$= \sum_s(\sum_t \mathbf{x}\mathbf{x}^\top) \otimes \mathbf{D}^{(s)}\mathbf{U}_1^\top\mathbf{U}_1\mathbf{D}^{(s)}.$$

$\mathbf{K}_2$ can be estimated as below:

$$\mathbf{K}_2 = \sum_{s,t}(\mathbf{x} \otimes \mathbf{D}^{(s)}\mathbf{U}_1^\top)(\mathbf{P}^{(s)}\mathbf{x} + \mathbf{r}^{(s)} + \mathbf{U}_3\mathbf{v}^{(s)}),$$
$$= \sum_s \mathrm{Vec}(\mathbf{D}^{(s)}\mathbf{U}_1^\top\mathbf{P}^{(s)}\sum_t \mathbf{x}\mathbf{x}^\top)$$
$$+ \sum_s(\sum_t \mathbf{x}) \otimes (\mathbf{D}^{(s)}\mathbf{U}_1^\top\mathbf{r}^{(s)})$$
$$+ \sum_s(\sum_t \mathbf{x}) \otimes (\mathbf{D}^{(s)}\mathbf{U}_1^\top\mathbf{U}_3\mathbf{v}^{(s)}).$$

4.3. Estimation of $\mathbf{U}_3$

The $\mathbf{U}_3$ that minimizes Q can be estimated as given below:

$$\frac{\partial Q}{\partial \mathbf{U}_3} \Rightarrow \sum_{s,t}\mathbf{e}(t)^{(s)}\mathbf{v}^{(s)\top} = 0,$$
$$\mathbf{U}_3 = -\mathbf{K}_3\mathbf{G}_3^{-1}.$$

The formulation of $\mathbf{K}_3$ and $\mathbf{G}_3$ is shown below. $\alpha^{(s)}$ is the number of feature frames for speaker s.

$$\mathbf{K}_3 = \mathbf{C}(t)^{(s)} + \mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x})\mathbf{v}^{(s)\top},$$
$$= \sum_s \mathbf{P}^{(s)}(\sum_t \mathbf{x})\mathbf{v}^{(s)\top} + \mathbf{r}^{(s)}\mathbf{v}^{(s)\top} * \alpha^{(s)}$$
$$+ \mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2(\sum_t \mathbf{x})\mathbf{v}^{(s)\top}.$$
$$\mathbf{G}_3 = \sum_s \mathbf{v}^{(s)}\mathbf{v}^{(s)\top}\alpha^{(s)}.$$

4.4. Re-estimation of $\mathbf{v}^{(s)}$

In our analysis, we use two approaches to approximate fMLLR error; in the first case, we only estimate U_1,U_2 and U_3 parameters iteratively. In the second case, in addition to those parameters, the i-vectors are re-estimated ($\mathbf{v}^{(s)}$).

$$\frac{\partial Q}{\partial \mathbf{v}^{(s)}} \Rightarrow \sum_t(\mathbf{U}_1\mathbf{D}^{(\mathbf{U}_2\mathbf{x})} + \mathbf{U}_3)^\top\mathbf{e}(t)^{(s)} = 0,$$
$$\mathbf{v}^{(s)} = -\mathbf{G}_4^{-1}\mathbf{K}_4,$$

where $\mathbf{G}_4 = \sum_t(\mathbf{U}_1\mathbf{D}^{(\mathbf{U}_2\mathbf{x})} + \mathbf{U}_3)^\top(\mathbf{U}_1\mathbf{D}^{(\mathbf{U}_2\mathbf{x})} + \mathbf{U}_3)$ and $\mathbf{K}_4 = \sum_t(\mathbf{U}_1\mathbf{D}^{(\mathbf{U}_2\mathbf{x})} + \mathbf{U}_3)^\top\mathbf{C}(t)^{(s)}$.

For efficient implementation, $\mathbf{G}_4$ and $\mathbf{K}_4$ are formulated as below:

$$\mathbf{G}_4 = \mathbf{U}_2(\sum_t \mathbf{x}\mathbf{x}^\top)\mathbf{U}_2^\top \star \mathbf{U}_1^\top\mathbf{U}_1$$
$$+ \mathbf{D}^{(\mathbf{U}_2(\sum_t \mathbf{x}))}\mathbf{U}_1^\top\mathbf{U}_3$$
$$+ \mathbf{U}_3^\top\mathbf{U}_1\mathbf{D}^{(\mathbf{U}_2(\sum_t \mathbf{x}))} + \mathbf{U}_3^\top\mathbf{U}_3\alpha^{(s)},$$
$$\mathbf{K}_4 = diag(\mathbf{U}_2(\sum_t \mathbf{x}\mathbf{x}^\top)\mathbf{P}^{(s)\top}\mathbf{U}_1)$$
$$+ \mathbf{D}^{(\mathbf{U}_2(\sum_t \mathbf{x}))}\mathbf{U}_1^\top\mathbf{r}^{(s)}$$
$$+ \mathbf{U}_3^\top\mathbf{P}^{(s)}(\sum_t \mathbf{x}) + \mathbf{U}_3^\top\mathbf{r}^{(s)}\alpha^{(s)},$$

where $\star$ is the element-wise multiplication.

5. EXPERIMENTS

5.1. Experimental Setup

In this paper, all the experiments are performed on the Aurora 4 corpus. The multi-condition training set of 83 speakers is used for training and the development set of 10 speakers is used as the validation set. We report the results on the test set of 8 speakers.

First, MFCC features are extracted from speech using a 25-ms window and a 10-ms frame-shift. Cepstral mean normalization (CMN) per speaker is then applied to the MFCCs. LDA features are obtained by first splicing 7 frames of 13-dimensional MFCCs and then projecting downwards to 40 dimensions using LDA. A global semi-tied covariance (STC) transformation [29] is applied on top of the LDA features. The GMM-HMM system for generating the alignments for DNN training is built on top of these 40 dimensional LDA features. In addition, we apply a speaker specific feature-space maximum likelihood linear regression (fMLLR) transform on top of the LDA features to create speaker normalized fMLLR features.

The initial DNN-HMM baseline is trained on the LDA features that span a context of 11 neighboring frames. Before being presented to the DNN, cepstral mean and variance normalization (CMVN) is performed on the features globally. To train the network, we used layer by layer-wise discriminative pre-training. The initial DNN has 7 sigmoid hidden layers with 2048 units per layer, and 2031 senones as the outputs. All the DNNs are trained to optimize the cross-entropy criterion with a mini-batch size of 256. CNTK [30] is used to train the DNNs. The Kaldi toolkit [31] is used to build the GMM-HMM systems and for the i-vector extraction. The i-vectors are trained on top of the same 40 dimensional LDA features. The UBM consist of 128 gaussians. We extracted i-vectors that are of 100 dimensions. The estimation of the fMLLR

transforms and the i-vector extraction used all the test speaker data in unsupervised fashion. In all our experiments, speaker-level i-vectors are used. All the decodings are performed with the WSJ0 bigram language model.

5.2. Results

Table 1 presents the results of various DNN models trained on top of the LDA features. It can be clearly seen that adding speaker information consistently improves the performance. A relative improvement 9.2% is reported over the baseline when both the bias and the transform ($+\mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x} + \mathbf{U}_3\mathbf{v}^{(s)}$) is used, which is also a 3.6% relative improvement over the standard i-vector baseline ($+\mathbf{U}_3\mathbf{v}^{(s)}$). The best performance (10.6%) is reported when three i-vector based feature transforms are estimated at the three lowest hidden layers. Therefore, speaker normalization of the bias and transform methods complement each other.

Table 1. Word Error Rate (WER %) of various DNN models on the test set. Relative improvement over the baseline is given in brackets.

Model	Test Set
LDA Baseline	11.9
LDA $+\mathbf{U}_3\mathbf{v}^{(s)}$	11.2 (5.9)
LDA $+\mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x}$	11.1 (6.7)
LDA $+\mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x} + \mathbf{U}_3\mathbf{v}^{(s)}$	10.8 (9.2)
LDA model with 3 layers of SD transforms	10.6 (10.9)

Figure 1 shows the WER (%) for various models with different numbers of i-vector based feature transforms, which include learning them on top of hidden layer representations. Moreover, these i-vector based feature transforms are added from the bottom (input layer) of the network. For instance, the model with two layers of SD transforms contains another i-vector based feature transform for the first hidden layer representations, in addition to the i-vector based transform in the input layer. Furthermore, all the models in this figure have a SD bias connected to the first hidden layer. It can be clearly seen that to reach the best performance, three layers of i-vector based transforms are sufficient. This suggests that lower layers are more important in speaker normalization.

Table 2 presents the results of various DNN models trained on top of the fMLLR features. A GMM-HMM system on top of fMLLR features is trained to obtain the training alignments for these models. Furthermore, i-vectors used in this experiment are trained using fMLLR features. The standard i-vector baseline ($+\mathbf{U}_3\mathbf{v}^{(s)}$) and the model with the additional SD transform ($+\mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x} + \mathbf{U}_3\mathbf{v}^{(s)}$) reported the same result (9.0). This is because fMLLR features are already transformed (using $\mathbf{A}^{(s)}$) per speaker . The best performance (8.8%) is reported when three i-vector based

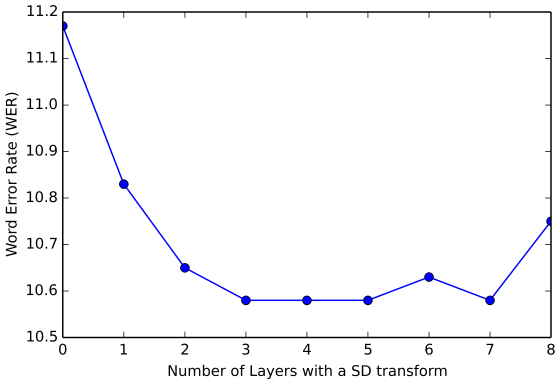

Fig. 1. WER (%) for various models against the number of layers with SD transform. Layer 8 is the output layer.

Table 2. WER (%) of various DNN models trained on fMLLR features. Relative improvement over the baseline is given in brackets.

Model	Test Set
fMLLR Baseline	9.5
fMLLR $+\mathbf{U}_3\mathbf{v}^{(s)}$	9.0 (5.3)
fMLLR $+\mathbf{U}_1\mathbf{D}^{(s)}\mathbf{U}_2\mathbf{x} + \mathbf{U}_3\mathbf{v}^{(s)}$	9.0 (5.3)
fMLLR model with 3 layers of SD transforms	8.8 (7.4)

feature transforms are estimated at lower layers, which is a 7.4% relative improvement over the baseline. However, the improvement over the standard i-vector based system is relatively smaller compared to that of DNNs trained on LDA features. This is simply because fMLLR features are already normalized using a speaker dependent transform to reduce the mismatch due to speaker variability.

5.2.1. *Least Squares Analysis*

First, we used the least squares based analysis to compare the standard i-vector technique ($+\mathbf{U}_3\mathbf{v}^{(s)}$) with the fMLLR method.

As shown in the first row of Table 3, the relative reduction of the error when compared with the full fMLLR transform is only 0.68%. In addition, the i-vector based bias method is not capable of approximating the error introduced by the transform $\mathbf{A}^{(s)}$ as shown in the second row. As predicted, it is capable of approximating the effect of the fMLLR bias ($\mathbf{b}^{(s)}$) as shown in the last row. However, generally, i-vectors provide a better bias shift than that of the $\mathbf{b}^{(s)}$ for speaker normalization.

Figure 2 shows the behavior of the average error (Q) by the number of iterations. As can be clearly seen, the error

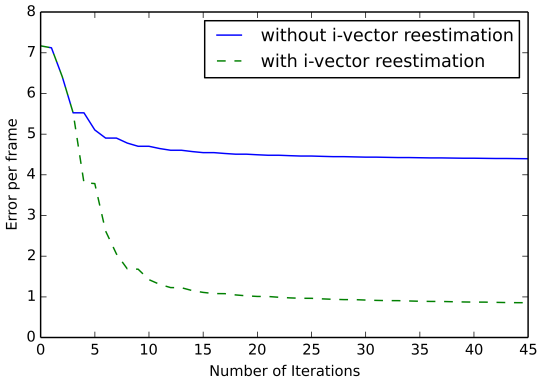

Fig. 2. The error (Q) for the least square approximation. Q is calculated per-frame. i-vector dimension is 100.

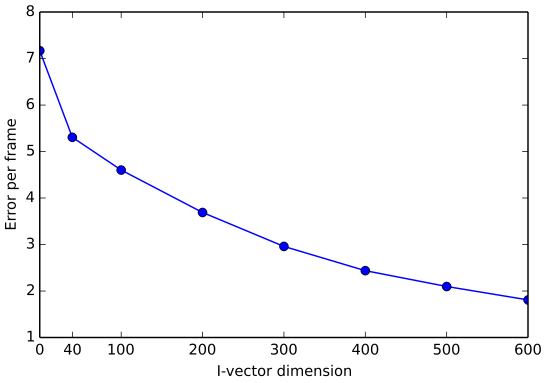

Fig. 3. Average error variation with the dimensionality of i-vectors.

Table 3. Least square approximation with the standard i-vector technique.

Scenario	Relative Reduction
Full transform ($\mathbf{A}^{(s)}x + \mathbf{b}^{(s)}$)	0.68%
Linear Transform ($\mathbf{A}^{(s)}x$)	0.02%
Bias shift ($\mathbf{b}^{(s)}$)	99.9%

reduction is better when the i-vectors are re-estimated. For instance, the relative error reduction is 38.7% when i-vectors are not re-estimated. However, relative error reduction is 88.1% when i-vectors are re-estimated.

Next, we used the re-estimated i-vectors in DNN training. This slightly improved the performance of both the i-vector baseline and our approach. These results are given in Table 4. To estimate the new i-vectors for the test and development sets, we used the parameter values of $\mathbf{U}_1$, $\mathbf{U}_2$ and $\mathbf{U}_3$ estimated from the training data.

Table 4. WER (%) of models trained with re-estimated 100 dimensional i-vectors ($\hat{\mathbf{v}}^{(s)}, \hat{\mathbf{D}}^{(s)} = diag(\hat{\mathbf{v}}^{(s)})$).

Model	Test Set
Baseline	11.9
$+\mathbf{U}_3\hat{\mathbf{v}}^{(s)}$	11.1 (6.7)
$+\mathbf{U}_1\hat{\mathbf{D}}^{(s)}\mathbf{U}_2\mathbf{x} + \mathbf{U}_3\hat{\mathbf{v}}^{(s)}$	10.7 (10.1)

Figure 3 shows the relationship between the average error and the dimensionality of the i-vector. The error decreases gradually as the i-vector dimension is increased. This behavior can be explained by the fact that fMLLR transforms have 1640 parameters (i.e., 40*41, since the feature size is 40), and therefore increasing the number of SD parameters aids approximation. That said, simply increasing the dimensionality of the i-vectors in DNN training may not always improve performance, because this can degrade the quality of the estimated i-vectors. In addition, the DNN models may be more prone to over-fitting due to the increased number of parameters.

In future work, we will use the derivations presented in this paper to investigate more suitable speaker representations for our approach. One of our aims is to find low dimensional speaker representations that can approximate the fMLLR transforms better than the i-vectors.

6. CONCLUSIONS

In this paper, we proposed a technique that incorporates speaker information to normalize the inter-speaker variability of the DNN based acoustic models. Our method relies on learning a separate subspace of feature transformations based on i-vectors. The speaker normalization experiments on the Aurora 4 task show relative improvements of 10.9% and 7.4% on top of the baseline system trained using the LDA and fMLLR features, respectively. We also compared our method with the fMLLR technique using a least square approximation. As future work, we will use the derivations presented in Section 4 to investigate more suitable representations for Speaker-aware Training (SaT) approaches. We believe it is important to find a low dimensional representation per speaker that can approximate the effect of fMLLR transforms more accurately than speaker-level i-vectors.

7. ACKNOWLEDGMENTS

This research was partially supported by the Google Faculty Research Award.

8. REFERENCES

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

[2] J. Gauvain and C.H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.

[3] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech and Language*, vol. 9, no. 2, pp. 171–186, 1995.

[4] H. Hermansky, D.P.W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. ICASSP*, 2000, pp. 1635–1638.

[5] P. Bell, P. Swietojanski, and S. Renals, "Multi-level adaptive networks in tandem and hybrid ASR systems," in *Proc. ICASSP*, 2013, pp. 6975–6979.

[6] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden markov models," in *Proc. ICASSP*, 1990, pp. 413–416.

[7] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Proc. ICASSP*, 2013, pp. 7893–7897.

[8] V. Gupta, P. Kenny, P. Ouellet, and T. Stafylakis, "I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription," in *Proc. ICASSP*, 2014, pp. 6334–6338.

[9] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proc. ASRU*, 2013, pp. 55–59.

[10] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *Proc. ICASSP*, 2013, pp. 7942–7946.

[11] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech and Language Processing*, 2010.

[12] O. Glembek, L. Burget, P. Matejka, M. Karafiat, and P. Kenny, "Simplification and optimization of i-vector extraction," in *Proc. ICASSP*, 2011, pp. 4516–4519.

[13] V. Abrash, H. Franco, A. Sankar, and M. Cohen, "Connectionist speaker normalization and adaptation," in *Proc. Eurospeech*, 1995, pp. 2183–2186.

[14] B. Li and K. C. Sim, "Comparison of discriminative input and output transformation for speaker adaptation in the hybrid nn/hmm systems," in *Proc. Interspeech*, 2010, pp. 526–529.

[15] F. Seide, Gang Li, Xie Chen, and Dong Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011, pp. 24–29.

[16] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. D. Mori, "Adaptation of hybrid ANN/HMM models using linear hidden transformations and conservative training," in *Proc. ICASSP*, 2006, pp. 1189–1192.

[17] X. Li and J. Bilmes, "Regularized adaptation of discriminative classifiers," in *Proc. ICASSP*, 2006, vol. 1, pp. I–I.

[18] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *Proc. ICASSP*, 2014, pp. 6359–6363.

[19] J. Stadermann and G. Rigoll, "Two-stage speaker adaptation of hybrid tied-posterior acoustic models," in *Proc. ICASSP*, 2005, pp. 977–980.

[20] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *Proc. SLT*, 2012, pp. 366–369.

[21] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proc. SLT*, 2014, pp. 171–176.

[22] P. Swietojanski and S. Renals, "Differentiable pooling for unsupervised speaker adaptation," in *Proc. ICASSP*, 2015, pp. 4305–4309.

[23] S. Dupont and L. Cheboub, "Fast speaker adaptation of artificial neural networks for automatic speech recognition," in *Proc. ICASSP*, 2000, pp. 1795–1798.

[24] T. Tian, Q. Yanmin, Y. Maofan, Z. Yimeng, and K. Yu, "Cluster adaptive training for deep neural network," in *Proc. ICASSP*, 2015, pp. 4325–4329.

[25] C. Wu and M. J. F. Gales, "Multi-basis adaptive neural network for rapid adaptation in speech recognition," in *Proc. ICASSP*, 2015, pp. 4315–4319.

[26] A. Senior and I. Lopez-Moreno, "Improving dnn speaker independence with i-vector inputs," in *Proc. ICASSP*, 2014, pp. 225–229.

[27] H. Huang and K. C. Sim, "An investigation of augmenting speaker representations to improve speaker normalization for DNN-based speech recognition," in *Proc. ICASSP*, 2015, pp. 4610–4613.

[28] M.J.F. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[29] M.J.F. Gales, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 272–281, 1999.

[30] D. Yu, A. Eversole, M. Seltzer, K. Yao, Z. Huang, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, H. Wang, et al., "An introduction to computational networks and the computational network toolkit," Tech. Rep., Tech. Rep. MSR, Microsoft Research, 2014, http://codebox/cntk, 2014.

[31] D. Povey, A. Ghoshal, G. Boulianne, N. Goel, M. Hannemann, Y. Qian, P. Schwarz, and G. Stemmer, "The kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.