Fast Adaptation of Deep Neural Network Based on Discriminant Codes for Speech Recognition

Article *in* IEEE/ACM Transactions on Audio, Speech, and Language Processing · December 2014

DOI: 10.1109/TASLP.2014.2346313

5 authors, including:

Ossama Abdel-Hamid
Apple Inc.
18 PUBLICATIONS **2,404** CITATIONS
SEE PROFILE

Hui Jiang
York University
157 PUBLICATIONS **4,623** CITATIONS
SEE PROFILE

Lirong Dai
University of Science and Technology of China
67 PUBLICATIONS **869** CITATIONS
SEE PROFILE

Fast Adaptation of Deep Neural Network based on Discriminant Codes for Speech Recognition

Shaofei Xue[1], *Ossama Abdel-Hamid*[2], *Hui Jiang*[2], *Lirong Dai*[1], *Qingfeng Liu*[1]

[1]National Engineering Laboratory of Speech and Language Information Processing,
University of Science and Technology of China, Hefei, P. R. China

[2]Department of Electrical Engineering and Computer Science, York University, Toronto, Canada

Email: xuesf@mail.ustc.edu.cn, ossama@cse.yorku.ca, hj@cse.yorku.ca, lrdai@ustc.edu.cn, qliu@iflytek.com

Abstract

Fast adaptation of deep neural networks (DNN) is an important research topic in deep learning. In this paper, we have proposed a general adaptation scheme for DNN based on discriminant condition codes, which are directly fed to various layers of a pre-trained DNN through a new set of connection weights. Moreover, we present several training methods to learn connection weights from training data as well as the corresponding adaptation methods to learn new condition code from adaptation data for each new test condition. In this work, the fast adaptation scheme is applied to supervised speaker adaptation in speech recognition based on either frame-level cross-entropy or sequence-level maximum mutual information training criterion. We have proposed three different ways to apply this adaptation scheme based on the so-called speaker codes: i) Nonlinear feature normalization in feature space; ii) Direct model adaptation of DNN based on speaker codes; iii) Joint speaker adaptive training with speaker codes. We have evaluated the proposed adaptation methods in two standard speech recognition tasks, namely TIMIT phone recognition and large vocabulary speech recognition in the Switchboard task. Experimental results have shown that all three methods are quite effective to adapt large DNN models using only a small amount of adaptation data. For example, the Switchboard results have shown that the proposed speaker-code-based adaptation methods may achieve up to 8-10% relative error reduction using only a few dozens of adaptation utterances per speaker. Finally, we have achieved very good performance in Switchboard (12.1% in WER) after speaker adaptation using sequence training criterion, which is very close to the best performance reported in this task [1].

Index Terms

Fast Adaptation, Deep Neural Network (DNN), Condition Code, Speaker Code, Cross Entropy (CE), Maximum Mutual Information (MMI),

EDICS number: SPE-RECO

I. INTRODUCTION

Speaker adaptation has been an important research topic in automatic speech recognition (ASR) for decades. Speaker adaptation techniques attempt to optimize ASR performance by transforming speaker-independent models towards one particular speaker or modifying the target speaker features to match a pre-trained speaker-independent model based on a relatively small amount of adaptation data from the target speaker. In the past, several successful speaker adaptation techniques have been proposed for the conventional Gaussian mixture hidden Markov model (HMM) in speech recognition. For example, the popular maximum a posteriori (MAP) method [2] has achieved very good performance when a large amount of adaptation data is available. On the other hand, maximum likelihood linear regression (MLLR) [3] and constrained MLLR (CMLLR) [4] have been reported to work much better when only a small amount of adaptation data is used. In these methods, all trained HMM parameters are transformed by one or a few linear functions that are learned from the available adaptation data. Moreover, the similar transformation functions may be directly applied in the feature space to transform speech features towards any given speaker-independent HMM models [5], [6]. In particular, for vocal track length normalization (VTLN) in [5], a parametric frequency warping function is used to normalize speech features among different speakers. This method is effective and robust since only one free parameter needs to be optimized per speaker. However, its performance is somehow limited by a manually designed frequency warping function.

As the hybrid deep neural networks (DNN) and HMM models have revived in acoustic modelling for large vocabulary continuous speech recognition (LVCSR) systems over the past years [7]–[10], it has become a very interesting research problem to perform effective speaker adaptation for DNN models. Obviously, some feature transformation techniques like VTLN can be directly used to normalize speech features prior to DNNs as in [10], [11]. In the literature, some speaker adaptation methods have been proposed to directly adapt neural networks (NN) towards target speakers. The most straightforward approach is to adjust all NN weights using the available adaptation data based on the standard error back-propagation (BP) training procedure. But this method is very prone to over-fitting especially when some class labels are missing in the adaptation data. Another more successful method is to add a linear input network (LIN) after the input as in [12]. During adaptation, only those weights of this linear layer are re-estimated based on the adaptation data. This method alleviates the over-fitting problem to some extent. In [13], the so-called linear hidden network (LHN) approach is proposed to add a linear transformation layer right before the output layer to process higher level features. In [14], linear output network (LON) is proposed to insert a linear transformation layer to directly transform the NN outputs. On the other hand, the retrained sub-set hidden units (RSHU) method in [15] tries to retrain only weights connected with active hidden nodes. In [16], Hermitian-based MLP (HB-MLP) method achieves the adaptive capability through the use of new orthonormal Hermite polynomials as activation functions in NN. All these methods depend on modifying part or all of the NN parameters using the limited adaptation data. Due to the large number of parameters to be re-estimated in adaptation, these methods typically require a relatively large amount of adaptation data to perform effective adaptation. Even in LIN or LHN, the number of parameters to be adapted cannot be easily controlled because it depends on the number of input

or output nodes in the original neural networks. More recently, feature discriminative linear regression technique in [11] and output-feature discriminative linear regression in [17] have also been proposed to perform speaker adaptation for DNNs. Furthermore, it has proposed to use Kullback-Leibler (KL) divergence as regularization in the adaptation criterion [18] since it forces the state distribution estimated from the adapted DNN to stay close enough to the original model to avoid over-fitting. In [19], it has explored how to adapt DNN to new speakers by other retraining and regularization tricks. In [20], it has investigated the performance of traditional speaker adaptation techniques on DNN-generated bottleneck features. In summary, in spite of these progresses, speaker adaptation remains a very challenging task for the hybrid DNN-HMM models in ASR, especially when only a very small amount of adaptation data is available per speaker, since adaptation of DNN is very prone to over-fitting due to a large number of model parameters in DNN.

In [21], [22], we have proposed a fast speaker adaptation method based on the so-called speaker codes for hybrid DNN/HMM models, which is capable of adapting large size DNNs using only a few adaptation utterances. This method relies on some speaker-specific discriminative codes, which are connected to a large speaker-independent neural network through a separate set of connection weights. These new connection weights and all speaker codes for all speakers in the training set can be jointly learned based on the available training data. Adaptation to a new speaker can be simply done by learning a new speaker code without changing other NN weights. This method is appealing because most newly introduced parameters can be reliably learned from the entire training data set while only a small speaker code is learned from adaptation data for each speaker. Moreover, the speaker code size can be freely adjusted according to the amount of available adaptation data. In [21], the speaker-code based adaptation has been found quite effective for fast speaker adaptation of hybrid DNN/HMM in small scale speech recognition tasks, like TIMIT. In [23], we have extended the idea of speaker-code based adaptation in [21] and proposed an alternative direct adaptation method that performs speaker adaptation in model space. This fast adaptation method has been shown to be quite effective in many large vocabulary speech recognition tasks, such as the Switchboard task. Note that a similar idea has been previously investigated for shallow neural networks in [24].

In this paper, we first extend our previous works scattered in [21]–[23] to propose a general adaptation scheme for DNNs based on discriminant condition codes, and then fully investigate how to apply this adaptation scheme to perform fast speaker adaptation for pre-trained speaker-independent DNNs based on the so-called speaker codes in both feature space and model space. In the feature space, the speaker codes are connected to another adaptation neural network, which is supposed to transform feature vectors prior to the original speaker-independent DNNs. The adaptation NN and all connection weights are estimated from the given training data. In this way, the learned adaptation NN serves as a nonlinear transformation to normalize each speaker's features into a generic speaker-independent feature space if a small speaker code is available for the speaker. In the model space, the basic idea is to connect speaker codes directly to all hidden and output layers of the original DNN through a set of new connection weights, which can be efficiently learned from all or part of training data along with additional information of speaker labels. In test stage, a new speaker code is similarly estimated for each new speaker from a small amount of adaptation data and the estimated speaker code is directly fed to the original DNN to form

a nonlinear transformation in model space. Moreover, we have also investigated several different training criteria to conduct supervised speaker adaptation for DNNs, including frame-level cross-entropy (CE) criterion as well as lattice-based sequence level maximum mutual information (MMI) criterion [25]. More importantly, in this paper, we have proposed a new speaker adaptive training strategy (SAT) similar to [26], [27] for DNNs based on the speaker codes. Instead of adapting a pre-trained speaker-independent DNN model based on adaptation data, the proposed SAT method can estimate more effective speaker-independent DNN models from the ground up in a single training procedure, by applying speaker normalization based on speaker codes. In this work, we have evaluated the proposed speaker-code based adaptation methods in two standard speech recognition tasks, namely the TIMIT phone recognition and the 320-hour Switchboard task. Experimental results have shown that the proposed speaker-code based methods are very effective to perform fast adaptation of DNNs for not only small ASR tasks but also the state-of-the-art large scale ASR tasks. For example, in the Switchboard task, the proposed speaker-code based adaptation methods may lead to up to 8-10% relative reduction in word error rate using only a few dozen of adaptation utterances per speaker.

The remainder of this paper is organized as follows. In section II, we briefly review the standard hybrid DNN-HMM model and its training methods based on both CE and MMI criteria for ASR. In section III, we introduce a general fast adaptation framework for DNN. In section IV, we present three different methods to apply the general adaptation scheme to supervised speaker adaptation in ASR. In section V, we report and discuss our experimental results in TIMIT and Switchboard tasks. Finally, the paper is concluded with our findings in section VI.

II. REVIEW OF THE HYBRID DNN-HMM MODELS

In this section, we briefly review the hybrid DNN-HMM model as well as the standard training methods of DNNs for ASR. A deep neural network is a feed-forward, conventional multi-layer perceptrons with many hidden layers. In the hybrid DNN-HMM model for ASR, DNN is used to directly model the posterior probabilities of all tied states of triphone HMMs [8].

A. DNN-HMM

An $(L+1)$-layer DNN, consisting of L hidden nonlinear layers ($l = 1...L$) and one output layer ($l = L+1$), is used to model the posterior probability $\Pr(s|X)$ of an HMM state s given an observation vector X. Each hidden unit, j, of layer l typically uses the sigmoid activation function, denoted as $\sigma(\cdot)$, to map its total input from the layer below, a_j^l, to its output, h_j^l, which in turn is sent to the next layer as input. Meanwhile, the topmost output layer uses softmax operation to compute posteriors for all class labels. The main equations related to DNN are listed as follows:

$$\mathbf{h}^0 = X \tag{1}$$

$$\mathbf{a}^l = \mathbf{W}^l \mathbf{h}^{l-1} + \mathbf{b}^l \quad (1 \leq l \leq L+1) \tag{2}$$

$$\mathbf{h}^l = \sigma(\mathbf{a}^l) \quad (1 \le l \le L) \quad \text{with} \quad h_j^l = \sigma(a_j^l) = \frac{1}{1+e^{-a_j^l}} \quad (\forall j) \tag{3}$$

$$h_s^{L+1} = \Pr(s|X) = \text{softmax}_s(\mathbf{a}^{L+1}) = \frac{e^{a_s^{L+1}}}{\sum_{s'} e^{a_{s'}^{L+1}}} \tag{4}$$

where $\mathbf{W}^l$ and $\mathbf{b}^l$ represent weight matrix and bias vector for layer l, a_j^l and h_j^l denote the j-th component of vector $\mathbf{a}^l$ and vector $\mathbf{h}^l$ of layer l, respectively. We normally learn all parameters of DNN, i.e., $\{\mathbf{W}^l, \mathbf{b}^l \mid 1 \le l \le L+1\}$, using the standard stochastic gradient descent (SGD) algorithm [28] to optimize an objective function $F(\cdot)$. For notational simplicity, we may expand the output vector, $\mathbf{h}^l$, in each layer by adding an additional dimension of constant 1 to incorporate the bias vector, $\mathbf{b}^l$, into the weight matrix, $\mathbf{W}^l$. As a result, we only consider how to learn weight matrices, $\mathbf{W}^l$, in this paper. The iterative updating formula is shown as:

$$\mathbf{W}^l \leftarrow \mathbf{W}^l - \eta \frac{\partial F}{\partial \mathbf{W}^l}, \quad (1 \le l \le L+1) \tag{5}$$

where η is the learning rate. In each update, we update all DNN parameter towards negative direction of the current gradient.

After calculating the state posteriors $\Pr(s|X)$ from the DNN output layer, we convert these state posteriors to the so-called scaled likelihoods by dividing the priors [7], [8] and then use them to replace GMM likelihoods in a Viterbi decoder to search for speech recognition results.

B. Learning DNNs for Speech Recognition

In speech recognition, DNN acts as a discriminative acoustic model to distinguish different phonemes or different phone states. First of all, DNN can be trained to optimize a frame-level cross-entropy (CE) criterion, which measures the difference between the target posterior probability distribution of all HMM states and the actual posteriori probability distribution calculated by DNN. Secondly, DNN parameters can be further refined for ASR by optimizing sequence-level maximum mutual information (MMI) discriminative criterion that is more closely related to the actual decision rule used in speech recognition.

1) Frame-Level Cross-Entropy (CE) Criterion:

DNN-based acoustic models estimate the posterior probability for every HMM state at its output layer. DNN is trained to optimize a given objective function, such as cross entropy error between the actual output distribution and the desired target distribution, using the standard error back-propagation algorithm through SGD. For any input vector, i.e., X_{rt}, denoting t-th feature vector from r-th utterance, the output distribution of DNN is calculated using the softmax function as follows:

$$y_{rt}(s) = \Pr(s|X_{rt}) = \frac{e^{a_{rt}^{L+1}(s)}}{\sum_{s'} e^{a_{rt}^{L+1}(s')}} \quad (\forall\, s) \tag{6}$$

where $a_{rt}^{L+1}(s)$ denotes the activation signal at the output layer corresponding to state s at time t for utterance r. Assuming that the whole training set consists of all training samples and their corresponding target labels, as $\{X_{rt}, s_{rt} \mid 1 \le r \le R, 1 \le t \le T_r\}$, the CE objective function can be expressed as the following form:

$$\mathcal{F}_{CE}(\boldsymbol{\mathcal{W}}) = -\sum_{r=1}^{R}\sum_{t=1}^{T_r} \log \Pr(s_{rt}|X_{rt}) = -\sum_{r=1}^{R}\sum_{t=1}^{T_r} \log y_{rt}(s_{rt}) \tag{7}$$

where $\boldsymbol{\mathcal{W}}$ denotes all connection weights in DNN, y_{rt} is the output at the top sotftmax layer given the input vector X_{rt}.

In the standard back-propagation (BP) algorithm, the most important quantities to calculate are the gradients of the CE objective function with respect to the input activations, a_j^l, at each layer, which are normally called as *error signals*:

$$e_{rt}^l(j) \equiv \frac{\partial}{\partial a_j^l} \log \Pr(s_{rt}|X_{rt}). \tag{8}$$

For the CE objective function in eq.(7), the error signals at the output layer can be computed as:

$$e_{rt}^{L+1}(s) = \frac{\partial \log y_{rt}(s_{rt})}{\partial a_{rt}(s)} = y_{rt}(s) - \delta_{rt}(s) \tag{9}$$

where $\delta_{rt}(s) = 1$ if s equals to the target label s_{rt} and otherwise $\delta_{rt}(s) = 0$. And the error signals for all other layers ($1 \leq l \leq L$) can be calculated from e_{rt}^{l+1} following a standard error back-propagation procedure as follows:

$$\mathbf{e}_{rt}^l = (\mathbf{W}^{l+1})'\mathbf{e}_{rt}^{l+1} \cdot \mathbf{h}_{rt}^l \cdot (1 - \mathbf{h}_{rt}^l) \quad (l = L, ..., 1) \tag{10}$$

where the vector $\mathbf{e}_{rt}^l$ is constructed by concatenating all error signals in layer l, and $\cdot$ stands for the element-wise multiplication between two vectors with the same size.

Once we have all error signals for all nodes j in all layers l given each input vector, the gradients for all DNN weights can be easily derived from these error signals. In this case, the gradients with respect to all DNN weights can be easily obtained as:

$$\frac{\partial \mathcal{F}_{CE}(\boldsymbol{\mathcal{W}})}{\partial \mathbf{W}^l} = \mathbf{e}_{rt}^l(\mathbf{h}_{rt}^{l-1})' \quad (\forall\, l) \tag{11}$$

Substituting eq.(11) into eq.(5), we can update all DNN weights iteratively based on stochastic gradient descent algorithm.

2) Sequence-level Maximum Mutual Information Criterion:

Sequence training attempts to simulate the actual MAP decision rule in speech recognition by incorporating sequence level constraints from acoustic models, lexicon and language models. In this work, we study the sequence training of DNN based on the maximum mutual information (MMI) criterion [25], [29], [30].

Assuming $\mathbf{X}_r = \{X_{r1}, ..., X_{rT_r}\}$ denotes the observation sequence of utterance r, and P_r is its reference word sequence label, the MMI objective function criterion is represented as:

$$\mathcal{F}_{MMI}(\boldsymbol{\mathcal{W}}) = \sum_r \log \frac{p(\mathbf{X}_r|S_r)^k Pr(P_r)}{\sum_{P\in\Omega_r} p(\mathbf{X}_r|S_P)^k Pr(P)} \tag{12}$$

where $S_r = \{s_{r1}, ..., s_{rT_r}\}$ is the reference state sequence corresponding to P_r, k is the acoustic scaling factor, and in the denominator the summation is taken over all competing hypotheses P in a word graph, Ω_r. Differentiating the MMI objective function in eq.(12) with respect to log likelihood $\log p(X_{rt}|s)$ for each HMM state s, we obtain:

$$\frac{\partial \mathcal{F}_{MMI}(\boldsymbol{\mathcal{W}})}{\partial \log p(X_{rt}|s)} = k\big(\gamma_{rt}^{num}(s) - \gamma_{rt}^{den}(s)\big) \tag{13}$$

where $\gamma_{rt}^{num}(s)$ and $\gamma_{rt}^{den}(s)$ stand for the posterior probabilities of being in state s at time t, computed for utterance r from the reference state sequence $\mathbf{S}_r$ and the word graph Ω_r, respectively. They take the following forms:

$$\gamma_{rt}^{num}(s) = p(s_t = s|\mathbf{X}_r, P_r) \tag{14}$$

$$\gamma_{rt}^{den}(s) = \sum_{P\in\Omega_r} \frac{p(\mathbf{X}_r|S)^k P(P)}{\sum_{V\in\Omega_r} p(\mathbf{X}_r|S_v)^k P(V)} p(s_t = s|\mathbf{X}_r, P) \tag{15}$$

Therefore, the required error signals in the output layer are calculated as follows:

$$\begin{aligned} e_{rt}^{L+1}(s) &= \frac{\partial \mathcal{F}_{MMI}(\boldsymbol{\mathcal{W}})}{\partial a_{rt}(s)} \\ &= \sum_{s'} \frac{\partial \mathcal{F}_{MMI}(\boldsymbol{\mathcal{W}})}{\partial \log p(X_{rt}|s')} \frac{\partial \log p(X_{rt}|s')}{\partial a_{rt}(s)} \\ &= \frac{\partial \mathcal{F}_{MMI}(\boldsymbol{\mathcal{W}})}{\partial \log p(X_{rt}|s)} - p(s|X_{rt}) \sum_{s'} \frac{\partial \mathcal{F}_{MMI}(\boldsymbol{\mathcal{W}})}{\partial \log p(X_{rt}|s')} \end{aligned} \tag{16}$$

where s' sums over all HMM states in the model. After some minor manipulation, it is straightforward to derive that the second term in eq.(16) equals to zero. After substituting eq.(13), we derive the error signals of the MMI objective function at time t of utterance r for state s in the output layer as follows:

$$e_{rt}^{L+1}(s) = k\big(\gamma_{rt}^{num}(s) - \gamma_{rt}^{den}(s)\big) \tag{17}$$

This error can be propagated to the previous layers in the same way as eq. (10) to derive error signals in all layers. Finally, all weights can be updated iteratively in eq. (5) just like the frame level cross-entropy training.

III. DNN Adaptation based on Discriminant Condition Codes

In this section, we propose a general scheme to adapt deep neural networks (DNN) based on the so-called discriminant condition codes, which are directly associated with various modelling conditions with respect to adaptation. This adaptation approach is capable of effectively adjusting a quite large DNN in a very fast fashion based on a small amount of adaptation data.

A. Model Description

Assume that we have an $(L+1)$-layer DNN consisting of weight matrices, denoted as $\{\mathbf{W}^l \mid 1 \le l \le L+1\}$. We know that it is difficult to adapt DNN using just a small amount of adaptation data since DNN typically consists of a large number of weights. In this paper, instead of directly adjusting DNN weights, we propose to use a separate set of the so-called discriminant condition codes, $\{\mathbf{s}^{(c)}\}$, each of which is associated with a different modelling

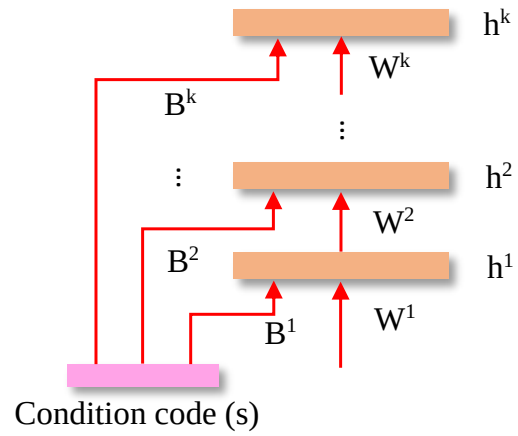


Fig. 1. Illustration of a general structure to adapt DNNs based on discriminant condition codes.

condition, c, relevant to adaptation [31]. As shown in Fig. 1, these condition codes are fed into some particular layers (either hidden or output layer) of DNN through another set of connection weights, denoted as $\mathbf{B}^l$ ($l \in \mathcal{L}$), where $\mathcal{L}$ stands for all layers of DNN that are connected to the condition codes. For any layer l ($l \in \mathcal{L}$), it receives input signals from both the lower layer $l-1$ and the condition code. Unlike eq.(3), the activation signals in these layers are computed as follows:

$$\mathbf{h}^l = \sigma\Big(\mathbf{W}^l\mathbf{h}^{l-1} + \mathbf{B}^l\mathbf{s}^{(c)}\Big) \quad (\forall l \in \mathcal{L}) \tag{18}$$

where $\mathbf{s}^{(c)}$ denotes the condition code associated with one particular modelling condition c. Assume the data come from C different modelling conditions in total, we should have C different condition codes, $\mathbf{s}^{(c)}$ ($1 \le c \le C$). Each condition code is simply a real vector, whose dimension can be freely adjusted based on the amount of available data under each condition.

B. Training and Adaptation

These condition codes along with their connection weights can also be learned from training data using the same error back-propagation (BP) algorithm. The gradient with respect to each element of the condition codes, $s_k^{(c)}$, can be easily calculated as follows:

$$\frac{\partial F}{\partial s_k^{(c)}} = \frac{1}{|\mathcal{L}|}\sum_{l\in\mathcal{L}}\sum_{j=1}^{J}\frac{\partial F}{\partial h_j^l}(1-h_j^l)h_j^l B_{jk}^l = \frac{1}{|\mathcal{L}|}\sum_{l\in\mathcal{L}}\sum_{j=1}^{J} e_j^l B_{jk}^l \tag{19}$$

where B_{jk}^l stands for an element in the connection weight matrix, $\mathbf{B}^l$, which connects k-th node in speaker code and j-th node in l-th layer, and e_j^l denotes the error signal derived for j-th node in l-th layer during the standard BP process. In the same way, we may calculate the gradient with respect to each element of the connection weight matrix, B_{jk}^l, as follows:

$$\frac{\partial F}{\partial B_{jk}^l} = \frac{\partial F}{\partial h_j^{(l)}}(1-h_j^{(l)})h_j^{(l)}s_k^{(c)} = e_j^l s_k^{(c)}. \tag{20}$$

In the learning process, both condition codes and their connection weight matrices are all randomly initialized. They are all learned in the BP procedure based on SGD. For each mini-batch in SGD, error signals are computed as usual and the corresponding derivatives are computed as in eqs.(19) and (20). For connection weight matrices $\mathbf{B}^l$, the gradients in eq.(20) are used to update all $\mathbf{B}^l$ in the same way as learning normal DNN weights. On the other hand, for condition codes, the gradients in eq.(19) are used to update different condition codes based on what condition each data sample comes from. As a result, we need to provide condition labels for all training samples when we learn the condition codes in the above procedure. For each update iteration in training, we use current model parameters to compute the gradients as in eqs.(19) and (20), which are in turn used to update the speaker codes and connection weights at the same time for the current iteration.

In the test stage, a new condition code, $\mathbf{s}^{(c)}$, is learned based on a small amount of adaptation data from the new test condition while we freeze all learned DNN weights, $\mathbf{W}^l$, as well as all connection weights, $\mathbf{B}^l$. In this paper, we use the supervised adaptation method to learn the new condition code, where a small set of labelled adaptation utterances are available for each new test condition. In this stage, the new condition code is first initialized to be zero and then it is updated based on the derivatives of the adaptation data as in eq.(19) until convergence. The learned condition code will be fed to DNN as in eq.(18) for testing purpose.

IV. Fast Speaker Adaptation of DNN based on speaker codes

In this section, as several typical applications of the above adaptation strategy for DNN, we propose several different ways to perform fast speaker adaptation of DNN in speech recognition. In this case, each condition code is associated with one speaker in data, which is thus called speaker code for convenience. Each speaker has one's own speaker code and each speaker code is a very compact feature vector representing speaker-dependent information. In the first method, we propose to add an adaptation neural network prior to the pre-trained DNN and speaker codes are connected to the adaptation neural network. In this case, the adaptation neural network can be viewed as a nonlinear transformation controlled by speaker codes to normalize input features towards the given speaker-independent model. In the second method, we propose to use the above method to directly adapt a pre-trained speaker-independent DNN in model space based on speaker codes, which are connected to all layers of the pre-trained DNN. Finally, instead of adapting a given speaker-independent DNN, we propose a joint learning procedure where all DNN weights, speaker codes and connection weights are jointly learned from scratch. This can be viewed as speaker adaptive training of DNNs based on speaker normalization with the proposed speaker codes.

The advantage of these methods is that only a small speaker code needs to be estimated for each new speaker. This largely reduces the required amount of adaptation data per speaker particularly when a small speaker code is chosen for each speaker. As a result, it is possible to conduct very rapid speaker adaptation for the hybrid DNN-HMM model in speech recognition based on only a few utterances per speaker. On the other hand, if a large amount of adaptation data is available per speaker, the size of speaker code can be increased to allow a better representation for each speaker.

A. Nonlinear Feature Normalization based on Speaker Codes: fSA-SC

In this section, we first propose a fast speaker adaptation method for DNN in feature space, which is called feature space speaker adaptation based on speaker code, denoted as *fSA-SC* for short. This method relies on a joint training procedure to learn a generic adaptation NN from the whole training set as well as many small speaker codes, one of which is estimated for each speaker only using data from that particular speaker. The speaker code is fed to the generic adaptation NN to form an effective nonlinear transformation in feature space to normalize speaker variations. This transformation is controlled by the speaker code. During adaptation, a new speaker code for a new speaker is learned such that the performance of the new speaker is optimized on the adaptation data. This method is appealing because the large adaptation network can be reliably learned from the entire training data set while only a small speaker code is needed for each speaker. Moreover, the speaker code size can be freely adjusted according to the amount of available adaptation data. As a result, it is possible to conduct a very fast adaptation of the hybrid DNN/HMM model for each speaker based on only a small amount of adaptation data.

Assume a speaker-independent (SI) DNN has been trained following the standard procedure as in [8] without using any speaker label information. As shown in Fig. 2, the proposed speaker adaptation method relies on learning another generic adaptation NN as well as some speaker specific codes. The adaptation NN is inserted above the input layer of original DNN-HMM model and consists of weights matrixes $\mathbf{A}^l$ and $\mathbf{B}^l$ (for all l), where l stands for the l-th layer of the adaptation neural network. The top layer of the adaptation NN represents the transformed features and its size matches the input size. Each layer of the adaptation neural network receives all activation output signals of the lower layer along with a speaker code $\mathbf{s}^{(c)}$ as follows:

$$\mathbf{h}^l = \sigma(\mathbf{A}^l\mathbf{h}^{l-1} + \mathbf{B}^l\mathbf{s}^{(c)}) \quad (\forall\, l). \tag{21}$$

When we estimate the adaptation NN using the BP algorithm, the derivatives of the objective function are calculated with respect to all NN weights, the connection weight matrices and the speaker codes in the same way as in the above sections II and III. As a result, all of them can be learned from all training data based on SGD in eq. (5). For example, when we apply feature vectors from c-th speaker to update the adaptation NN in BP, we use the computed derivatives to update all weights, $\mathbf{A}^l$ and $\mathbf{B}^l$ ($\forall\, l$), and the speaker code $\mathbf{s}^{(c)}$ specific to the c-th speaker. In this way, we are able to benefit from speaker labels to learn a generic adaptation NN as well as a whole bunch of speaker codes at the end of the BP training process. Each speaker has one's own speaker code and each speaker code, $\mathbf{s}^{(c)}$, is a very compact feature vector representing speaker-dependent information. The speaker code is fed to the adaptation NN to control how each speaker's speech features are transformed into a general speaker-independent feature space.

Moreover, this model configuration provides a very effective way to conduct speaker adaptation for DNN. To adapt the model to a new speaker, c, only a new speaker code, $\mathbf{s}^{(c)}$, needs to be estimated without changing any weights in both original NN and adaptation NN in Fig. 2. The advantage of our proposed method is that only a small speaker code needs to be estimated for each new speaker but the generic adaptation NN is learned using all training data. This allows to build a large adaptation NN that is powerful enough to model a complex transformation

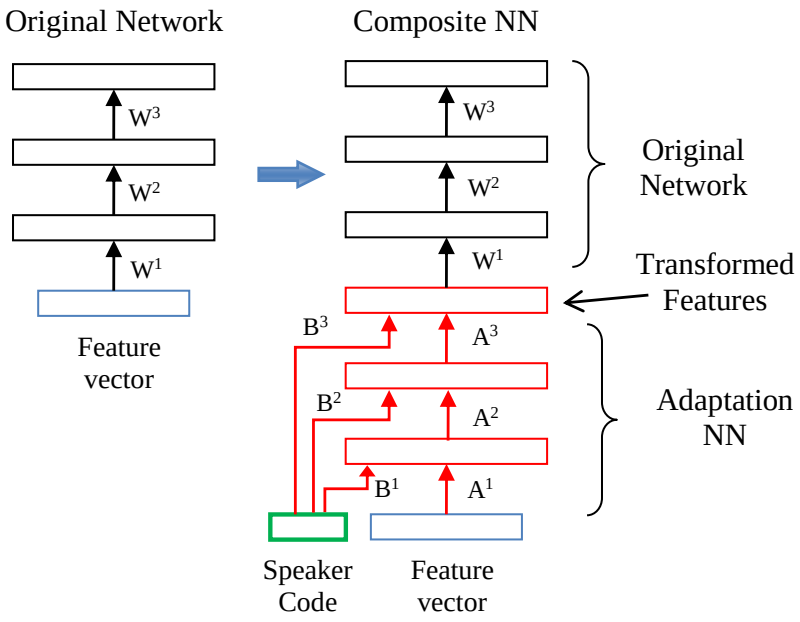


Fig. 2. Illustration of the proposed model structure for nonlinear feature normalization prior to DNN based on speaker codes, denoted as *fSA-SC*.

function between different feature spaces. This method is clearly superior to other speaker adaptation methods that learn an independent transform for each speaker, where each transformation needs to be linear.

During the training stage, we need to learn three sets of parameters: the original NN weights $\mathbf{W}^l$, adaptation NN weights $\mathbf{A}^l$ and connection weights $\mathbf{B}^l$, and all training speaker codes $\mathbf{s}^{(c)}$. First of all, the original DNN weights, $\mathbf{W}^l$, are learned without inserting the adaptation weights in the same way as a standard hybrid DNN-HMM model with no speaker information. This results in a speaker independent DNN-HMM model. Secondly, the adaptation layers are inserted and all adaptation weights, including $\mathbf{A}^l$ and $\mathbf{B}^l$, and speakers codes $\mathbf{s}^{(c)}$ for all speakers in the training set, are all randomly initialized and then jointly learned in the standard back-propagation algorithm using the derivatives w.r.t. $\mathbf{A}^l$ as in eq.(11), and the derivative w.r.t. $\mathbf{B}^l$, as in eq.(20). Similarly, we compute the derivatives with respect to each element of all speaker codes as in eq.(19), which are used to update the speaker code $\mathbf{s}^{(c)}$ based on all data from this speaker. In this stage, we have several different methods to deal with the original DNN weights when training the adaptation NN. For example, we may simply keep all DNN weights, $\mathbf{W}^l$, unchanged during this stage. Of course, all or part of the original NN weights can be further fine-tuned when learning the adaptation NN to further optimize the whole network because speaker labels are considered in this phase.

After learning all adaptation NN weights using all training data as above, adaptation to a new speaker is done by learning a new speaker code for each new speaker who is not observed in the training set. In this work, we use the supervised adaptation method where a small set of labelled adaptation utterances are available for each new speaker. The state level alignments of the adaptation data are first obtained from the trained HMM model using the forced alignment method. After that, we use the same BP training procedure as in the above except that only

the speaker code $\mathbf{s}^{(c)}$ is updated based on its derivatives while all other NN weights are kept unchanged. After the new speaker code is learned from a small amount of adaptation data, the speaker code will be fed to the adaptation NN along with new test data for recognition as in Fig. 2.

B. DNN Adaptation in Model Space: mSA-SC

As a more straightforward way to apply the above adaptation strategy to speaker adaptation in speech recognition, we propose an adaptation structure as in Fig. 3 to adapt a pre-trained speaker-independent DNN. This method is named as model space speaker adaptation based on speaker code, denoted as *mSA-SC*. The basic idea of mSA-SC is to connect speaker codes directly to all hidden and output layers of the original DNNs through a set of new connection weights, which can be efficiently learned from all training data using additional information of speaker labels. In test stage, a new speaker code is estimated for each new speaker from a small amount of adaptation data and the estimated speaker code is directly fed to the original DNN to form a nonlinear transformation in model space. Since there is no need to estimate another adaptation neural network, the additional training time prior to adaptation may be reduced significantly.

As show in Fig. 3, instead of stacking an adaptation NN below the initial speaker-independent DNN and normalizing speech features in feature space, we propose to feed the speaker codes directly to the hidden layers and the output layer of the initial DNN through a set of new connection weights $\mathbf{B}^l$. In this way, speaker codes are directly used to adapt the speaker-independent DNNs towards any new target speaker. A main advantage of this new adaptation scheme is that the computation complexity is dramatically reduced in training because we have no need to learn another set of weight matrices, $\mathbf{A}^l$, from training data. In many cases, $\mathbf{A}^l$ is significantly bigger than $\mathbf{B}^l$ in size. Moreover, learning is easier for deeper networks since error signals are back-propagated through a smaller number of hidden layers before updating either the speaker code or the connection weights $\mathbf{B}^l$.

Following the same method in section III, we estimate all connection weights, $\mathbf{B}^l$, and speaker codes, $\mathbf{s}^{(c)}$, from training data for this new adaptation scheme without modifying the original DNN weights, $\mathbf{W}^l$. In the training stage, we first randomly initialize all $\mathbf{B}^l$ and $\mathbf{s}^{(c)}$. Next, we run several epochs of stochastic gradient descents over the training data to update $\mathbf{B}^l$ and $\mathbf{s}^{(c)}$ based on the gradients computed in eq.(20) and eq.(19). For speaker codes, $\mathbf{s}^{(c)}$ is only updated by data from c-th speaker. At the end, we have learned all weight matrices $\mathbf{B}^l$ which are capable of adapting the speaker-independent DNN to any new speaker given a suitable speaker code.

The next step in adaptation is to learn a speaker code for each new target speaker. During this phase, we still use the supervised adaptation method, where only the speaker code is estimated using the gradients in eq.(19) for the new speaker from a small number of adaptation utterances and their labels (all $\mathbf{B}^l$ and $\mathbf{W}^l$ remain unchanged). After the speaker code is learned for each test speaker, the speaker code is imported into the neural network through $\mathbf{B}^l$ as in eq.(18) to compute posterior probabilities of test utterances for final recognition.

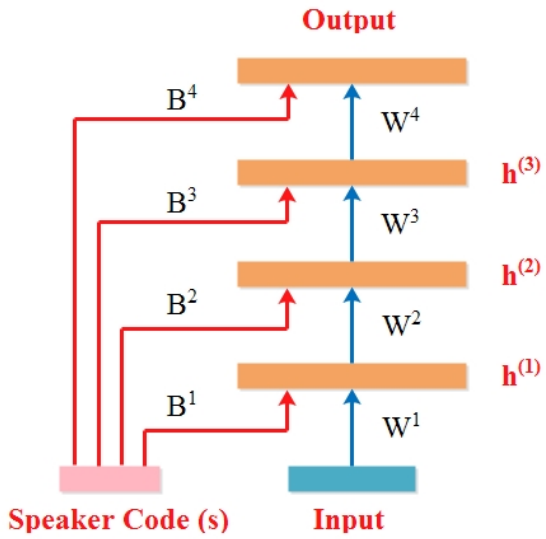


Fig. 3. Illustration of the proposed model structure for direct adaptation of DNNs based on speaker codes, denoted as *mSA-SC*.

C. Joint Speaker Adaptive Training based on Speaker Codes: SAT-SC

Both fSA-SC and mSA-SC are proposed to adapt a pre-trained speaker-independent DNN in feature space and model space respectively. In fSA-SC, a separate adaptation NN controlled by speaker codes is learned to normalize speech features prior to the speaker-independent DNN. In mSA-SC, a set of connection weights as well as speaker codes are learned from training data to directly adapt the speaker-independent DNN in model space. Generally speaking, mSA-SC is more suitable for large scale speech recognition tasks since it has significantly reduced extra training time prior to speaker adaptation.

In this section, we propose a new training strategy to train speaker-dependent DNN as well as all adaptation parameters for even better training efficiency, which is called joint speaker adaptive training based on speaker codes, denoted as *SAT-SC*.[1] In SAT-SC, instead of training a speaker-independent DNN beforehand and performing adaptation on it, we propose to feed the speaker codes directly to the hidden layers and the output layer of the neural network through a set of new connection weights from the beginning. Assume we adopt the model structure of mSA-SC in Fig. 3. In this case, the baseline DNN weights $\mathbf{W}^l$ are initialized randomly or by pre-training, and other adaptation parameters, including, $\mathbf{B}^l$ and $\mathbf{s}^{(c)}$, are first randomly initialized. After that, all of these parameters are jointly learned based on their gradients using the standard BP algorithm, where speaker labels are provided for training speaker codes, $\mathbf{s}^{(c)}$, for all speakers in the training set. This method can be viewed as a new speaker adaptive training (SAT) strategy [26], [27] for DNN based on speaker normalization controlled by speaker codes. Obviously, this SAT method further decreases the total training time to generate the generic speaker-independent DNN model

[1]In fSA-SC and mSA-SC, part of the model is trained with speaker labels, which may be viewed as a partial speaker adaptive training (SAT). On the other hand, in SAT-SC, the entire model is fully learned from scratch in a stricter sense of SAT.

as well as the related adaptation parameters since it has combined the training process of $\mathbf{W}^l$ and $\mathbf{B}^l$ into a single stage. Moreover, this SAT-SC method may further improve the performance of the generic speaker-independent DNN over the previous mSA-SC method [26]. In mSA-SC, $\mathbf{W}^l$ and $\mathbf{B}^l$ are trained separately in two different steps, where we first train all $\mathbf{W}^l$ and then fix them to learn $\mathbf{B}^l$. However, in SAT-SC, all model parameters are jointly learned based on the provided training data as well as speaker label information, which presumably generates better and more compact speaker-independent models.

After learning all $\mathbf{W}^l$ and $\mathbf{B}^l$ using all training data, the adaptation to a new speaker is done in the same way as mSA-SC. We use the supervised adaptation method where a small set of labelled adaptation utterances are available for each new speaker. And the BP training algorithm is applied to only learn a new speaker code $\mathbf{s}^{(c)}$ based on the derivatives while all the other model parameters are kept unchanged. After the speaker code is learned from a small amount of adaptation data, the speaker code will be fed to the model structure in Fig. 3 along with speech features for final recognition.

V. EXPERIMENTS

In this section, we first evaluate the proposed adaptation methods for rapid speaker adaptation on the small-scale TIMIT phone recognition task based on the frame-level CE training criterion. Afterwards, we use the standard large scale 320-hr Switchboard task to evaluate the proposed fast speaker adaptation methods for both the frame-level CE training criterion as well as the MMI-based sequence training.

A. TIMIT Phone Recognition

In all TIMIT phone recognition experiments, we use the standard 462-speaker training set. A separate development set of 50 speakers is used for tuning all of the meta parameters in training. The final recognition results are reported using the standard core test set consisting of 24 new speakers with no overlap with those in the development set and the training set. Each speaker in the test set has eight test utterances.

In feature extraction, speech is analyzed using a 25-ms Hamming window with a 10-ms fixed frame rate. The speech feature vector is generated by a Fourier-transform-based filter-banks which include 40 coefficients distributed on the Mel scale and energy, along with their first and second temporal derivatives. This leads to a 123-dimension feature vector per speech frame. All speech data are normalized by averaging over all training samples so that all feature vectors have zero mean and unit variance. We use 183 target class labels (i.e., 3 states for each one of the 61 phones) for DNN training. After decoding, these 61 phone classes are mapped to a set of 39 classes for the final scoring purpose as in [32]. In our experiments, a bi-gram language model at phone level, estimated from all transcripts in the training set, is used for recognition. When training speaker-independent DNN, we use an RBM-based pre-training method in [33], [34] to initialize all DNN weights prior to the BP-based fine-tuning.

In all fSA-SC experiments, learning rate annealing and early stopping strategies are utilized as in [35] to train the weights of the original DNN as well as in the adaptation of DNN. The NN input layer includes a context window of 15 consecutive frames. During adaptation, we use a fixed learning rate of 0.1 for sigmoid layers and 0.025 for

linear layers to learn speaker code for each target speaker, where the momentum is always kept as 0.9. In fSA-SC, the mini-batch size is set to 128 and speaker code is 50 in size. The number of epochs is determined using the development set and it may vary for different sizes of adaptation data set. Similarly, in all mSA-SC experiments, we use the same early stopping strategies as above with an initial learning rate of 0.5 to learn the weight matrices $\mathbf{B}^l$, where the momentum is also kept as 0.9. The mini-batch size is set to 128 but we use larger speaker codes in mSA-SC, each of which is 300 in size.

Since each test speaker has eight utterances in total. Adaptation and testing is conducted for each speaker based on a cross validation method. In each run, for each speaker, eight utterances are divided into n_a utterances for adaptation and the remaining $8 - n_a$ utterances for test. This is repeated eight times for each speaker. Each time, different adaptation and test utterances are randomly selected in such a way that each utterance is assigned the same times for both adaptation and testing. The overall recognition performance is the average among all eight runs.

1) Performance of fSA-SC on different adaptation set sizes: In the first set of experiments, we measure the performance of the proposed fSA-SC method using different amounts of adaptation data. In this experiment, a baseline NN with two hidden layers is first trained. We use an adaptation NN as shown in Fig. 2. The adaptation NN also contains two hidden layers in addition to the output layer which has a linear activation function. All hidden layers have 1000 nodes each. We used speaker code size of 50 in fSA-SC.

In Fig. 4, we have shown the adaptation performance using different numbers of adaptation utterances in the above cross-validation setting (varying n_a from 1 to 7). It shows that even with one adaptation utterance, we can achieve performance gain of about 1.3% (about 5.7% relative) reduction in error rate. After using seven utterances for adaptation, phone error rate (PER) drops from 22.83% to 20.5%. The *dummy* adaptation means we simply add adaptation NN without using any speaker codes, which is equivalent to increasing DNN from 2 hidden layers to 5 hidden layers. The results in Fig. 4 show that it only achieves very minor improvement by simply adding these two dummy layers. This confirms that the gain is obtained mainly from adaptation not from just adding more layers. The performance using zero adaptation utterances means that the connection weights are learned for the two adaptation layers in training but the speaker code is not learned in adaptation for each new speaker and it is simply set as all zeros for recognition. It indicates that the proposed fSA-SC may be a little helpful even without learning a new speaker code during the testing stage. The Oracle score is the PER when we use the same eight utterances per speaker for both adaptation and testing.

2) Performance fSA-SC and mSA-SC on TIMIT: In this set of experiments, we compare the performance of fast speaker adaptation on the TIMIT database in feature space (using fSA-SC) and model space (using mSA-SC). For the TIMIT database, we only consider DNN training and adaptation based on the frame-level cross-entropy criterion. In this experiment, we have trained two baseline speaker-independent (SI) DNNs, i.e., one with 2 hidden layers of 1000 nodes and the other with 5 hidden layers of 1000 nodes. From results shown in Table I, the deeper model achieves slightly better recognition performance on TIMIT, i.e., 21.6% vs. 22.8% in phone error rate (PER). If we apply fSA-SC based speaker normalization in feature space to both SI DNNs using only 7 utterances per speaker, fSA-SC improves recognition performance to 20.5% for 2-hidden-layer DNN and 20.7% for 5-hidden-layer DNN,

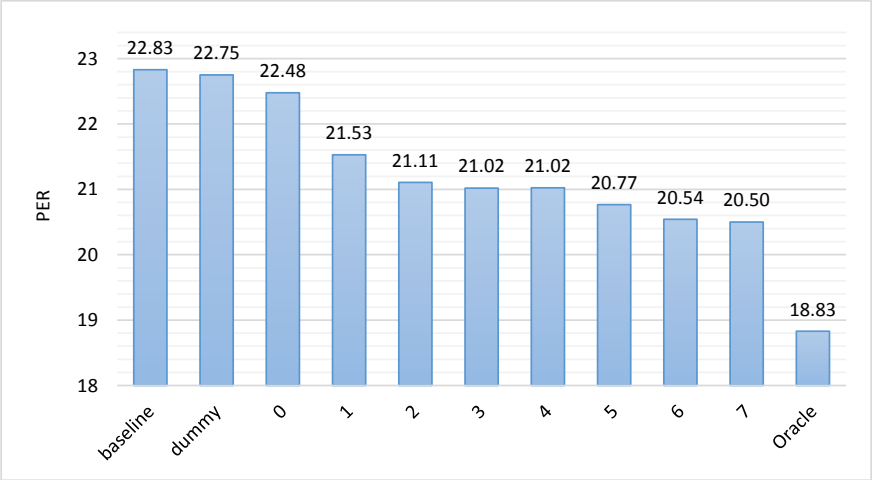


Fig. 4. Adaptation performance (phone error rate in %) as a function of number of adaptation utterances in TIMIT.

accounting for 10.2% and 4.2% relative improvements. On the other hand, if we apply mSA-SC based speaker adaptation in model space to both DNNs using the same 7 utterances per speaker, mSA-SC improves recognition performance to 21.6% for 2-hidden-layer DNN and 20.3% for 5-hidden-layer DNN, accounting for about 5.4% and 6.2% relative improvements. The results also suggest that fSA-SC works the best for relative smaller and more shallow baseline DNN but mSA-SC yields significantly better performance for larger and deeper DNN baseline. This can be easily explained because in learning of speaker codes in fSA-SC as in Fig. 2, error signals need to be propagated through the baseline DNN and they may get relatively weak after a very deep model. Meanwhile, for mSA-SC in Fig. 2, learning of speaker codes can directly receive error signals even from the output layer, which makes the learning quite effective even for very deep baseline model.

TABLE I

TIMIT adaptation performance (phone recognition rate in %) on two different DNN baselines (one with 2 hidden layers of 1000 and the other with 5 hidden layers of 1000) using fSA-SC and mSA-SC. All results are shown for the case using 7 adaptation utterances per speaker.

	PER	relative error reduction
DNN (2x1000)	22.8%	-
DNN (5x1000)	21.6%	-
DNN (2x1000) + fSA-SC	20.5%	10.2%
DNN (5x1000) + fSA-SC	20.7%	4.2%
DNN (2x1000) + mSA-SC	21.6%	5.4%
DNN (5x1000) + mSA-SC	20.3%	6.2%

B. Switchboard

The Switchboard (SWB) training data set consists of 309 hour Switchboard-I data set and 20 hour Call Home English set, including about 1540 speakers in total. In this work, we use the standard NIST 2000 Hub5e set (containing 1831 utterances from other 40 new speakers) as the evaluation set.

For all Switchboard experiments, we use PLP features (static, first and second derivatives) that are pre-processed with cepstral mean and variance normalization (CMVN) per conversation side. The baseline GMM-HMM (with 8,991 tied states and 40 Gaussians per state) is first trained based on maximum likelihood estimation (MLE) and then discriminatively trained using the MPE criterion. A trigram language model (LM) is trained using 3M words of the training transcripts and 11M words of the Fisher English Part 1 transcripts. The baseline tri-phone GMM-HMMs model is used to obtain the state level alignment labels for both training and evaluation sets for DNN training and adaptation. The baseline DNNs are trained as described in [10], [36], [37] with the RBM-based pre-training and the BP-based fine-tuning using the frame-level CE criterion. In this experiment, we have built three baseline speaker-independent DNNs with various model sizes: i) 3 hidden layers with 1024 nodes in each hidden layer; ii) 3 hidden layers with 2048 nodes in each hidden layer; iii) 6 hidden layers with 2048 nodes in each hidden layer. The baseline performance of these speaker independent DNNs is listed in the second column in Table II, which is comparable with other results reported in the Switchboard task [7], [9], [38].

In the evaluation set (Hub5e00), each test speaker has different number of utterances. We use two different cross-validation (CV) configurations for adaptation and test. Firstly, in each CV run, a fixed number of utterances (to say 10, 20) is used as adaptation data and the remaining utterances from the same speaker is used to evaluate recognition performance. The process is rotated for many runs until all test utterances are covered. The overall recognition performance is computed as the average of all runs. Secondly, we also consider to use maximum number of utterances per speaker for adaptation, called *max adaptation*. For every test utterance in Hub5e00, we use all remaining utterances from the same speaker to adapt DNNs that is in turn used to recognize only this test utterance. The process is repeated for all utterances in Hub5e00. Since the number of utterances is different for each speaker in the test set, the number of adaptation utterances used in this case varies from minimum 25 utterances to maximum 67 utterances per speaker (46 utterances per speaker in average). In all the following experiments, when we use the proposed methods to adapt speaker-independent DNN in the adaptation stage to learn a new speaker code for each new speaker, the learning rate is set as 0.02 and the mini-batch size is set as 128 and the learning is conducted for 5 epochs.

1) Performance of fSA-SC on SWB: We first examine the proposed feature normalization method with fSA-SC in section IV-A on the Switchboard task. In this experiment, we use an adaptation NN consisting of 2 hidden layers of 2048 nodes and one linear output layer. During the adaptation, we use the whole training set to learn the adaptation NN as well as to fine-tune only the first layer of the baseline speaker-independent DNN. In this stage, we choose an initial learning rate of 0.1 and the similar learning rate annealing strategy and early stopping strategy as above. Furthermore, the momentum is kept as 0.9, the mini-batch size is set to 1024. In this experiment, we choose to use

a speaker code with 500 in dimension for each speaker. The adaptation normally converges after about 10 epochs. Experimental results shown in Table II clearly demonstrate that the fSA-SC scheme is quite effective to perform fast speaker adaptation for all three DNN models. The baseline performance of speaker-independent DNNs can be significantly improved by using only a small number of adaptation utterances per speaker. In particular, fSA-SC is quite suitable for adapting small and shallow DNNs. For example, fSA-SC improves a smaller DNN (with 3 hidden layers of 1024 nodes each) from 18.9% in word error rate (WER) to 17.5% by using only 10 adaptation utterances per speaker, accounting for about 7.4% relative error reduction. Furthermore, fSA-SC improves to 17.0% (about 10% relative error reduction) using the max adaptation strategy. On the other hand, fSA-SC yields much less performance gain in adapting large and deep DNN. For instance, fSA-SC improves WER from 16.2% in WER to 15.4% (about 4.9% relative error reduction) in the max adaptation scheme for a larger DNN (with 6 hidden layers of 2048 nodes each).

TABLE II

Switchboard performance comparison (WER in %) in feature space normalization using fSA-SC on 3 different speaker-independent DNNs based on 10, 20 and maximum number adaptation utterances per speaker. Both baseline training and fSA-SC adaptation of DNNs are based on the frame-level cross-entropy criterion. Numbers in bracket denote relative gain over the baseline.

DNN	baseline	adaptation utterances per speaker		
		10	20	max
3layer*1024node	18.9	17.5	17.3	17.0 (10.0%)
3layer*2048node	17.4	16.3	16.1	16.0 (8.0%)
6layer*2048node	**16.2**	15.7	15.5	**15.4** (4.9%)

2) Performance of mSA-SC on SWB: In this section, we evaluate the mSA-SC method in section IV-B for fast speaker adaptation of baseline DNNs in the Switchboard task. When we learn connection matrices $\mathbf{B}^l$ and speakers codes $\mathbf{s}^{(c)}$ in mSA-SC, we use an initial learning rate of 0.5 and it is halved every epoch after the first three epochs. The momentum is kept as 0.9. The mini-batch size is set to 1024. In mSA-SC, we choose to use larger speaker codes, each of which is 1000 in size. The learning process typically converges after only 4-6 epochs. In this experiment, we apply the proposed mSA-SC in Fig. 3 to the three speaker-independent baseline DNNs. The results in Table III have shown that the mSA-SC scheme is also quite effective to rapidly adapt all baseline DNNs using only a small number of adaptation utterances per speaker. Particularly, the results show that mSA-SC significantly outperforms fSA-SC in adapting large and deep models. For example, for a large baseline DNN (with 6 hidden layers of 2048 node each), mSA-SC can improve the baseline performance from 16.2% in WER to 14.9% (about 8% relative error reduction).

For comparison, we have also investigated the effect of model size on recognition performance. For this purpose, we have built another bigger DNN (with 6 hidden layers of 2690 nodes each layer), which has roughly the same number of model parameters (61 millions) as our mSA-SC method (after adding all $\mathbf{W}^l$ and $\mathbf{B}^l$ together). The performance of this larger DNN is shown in the last row of Table III, which is 15.9% in WER (only slightly better than the baseline). This indicates that the gain of mSA-SC mainly comes from speaker adaptation not from

more model parameters. Furthermore, we have also evaluated how different SC sizes and varied SC connection schemes may affect recognition performance in mSA-SC. In Table IV, we have listed the performance of mSA-SC using different sizes of speaker code (varying from 500 to 1500). The results have clearly shown that adaptation performance is not very sensitive to SC size but the speaker code size of 1000 yields the best performance. As a result, we use this size for all of our Switchboard experiments. Finally, as shown in Table V, we have shown the performance of mSA-SC using three other SC connection schemes: i) Speaker code is only connected to the first hidden layer; ii) Speaker code is only connected to one middle hidden layer; iii) Speaker code is only connected to the output layer. The results have shown that connecting the speaker code to the middle hidden layer is better than other two schemes but it is still not as good as connecting the speaker code to all the layers, which is used as the standard connection scheme in mSA-SC.

TABLE III

Switchboard performance comparison (WER in %) in model space using mSA-SC on 3 different speaker-independent DNNs based on 10, 20 and maximum number adaptation utterances per speaker. Both baseline training and mSA-SC adaptation of DNNs use the frame-level cross-entropy criterion. Numbers in bracket denote relative gain over the baseline.

DNN	baseline	adaptation utterances per speaker		
		10	20	max
3layer*1024node	18.9	17.8	17.8	17.4 (7.9%)
3layer*2048node	17.4	16.4	16.1	15.9 (8.6%)
6layer*2048node	**16.2**	15.2	15.2	**14.9** (8.0%)
6layer*2690node	15.9	-	-	-

TABLE IV

*Switchboard performance comparison (WER in %) in model space using mSA-SC on DNNs (6layer*2048node) with different speaker code sizes based on 10, 20 and maximum number adaptation utterances per speaker. Both baseline training and mSA-SC adaptation of DNNs use the frame-level cross-entropy criterion.*

SC size	baseline	adaptation utterances		
		10	20	max
500	16.2	15.5	15.4	15.2
800	16.2	15.3	15.3	15.1
1000	16.2	**15.2**	**15.2**	**14.9**
1200	16.2	15.3	15.2	15.0
1500	16.2	15.4	15.3	15.0

In addition, we also extend the mSA-SC method to MMI-based sequence training experiments. We first use the 6-layer DNN baseline (with 16.2% in WER) to conduct forced-alignment to generate new state level labels for both training and test set. The re-aligned training data is used to re-train another baseline speaker-independent DNN with 6 hidden layers of 2048 nodes. As shown in Table VI, the re-aligned DNN slightly improves the baseline performance from 16.2% to 15.9%. Furthermore, we conduct sequence training based on the MMI criterion and it

TABLE V

*Switchboard performance comparison (WER in %) in model space using mSA-SC on DNNs (6layer*2048node) with different connection scheme based on 10, 20 and maximum number adaptation utterances per speaker. Both baseline training and mSA-SC adaptation of DNNs use the frame-level cross-entropy criterion.*

Connecting SC to	baseline	adaptation utterances		
		10	20	max
all layers (mSA-SC)	16.2	15.2	15.2	**14.9**
first hidden layer	16.2	15.6	15.4	15.3
4-th hidden layer	16.2	15.5	15.3	15.1
output layer	16.2	16.0	15.9	15.7

has shown that the sequence training further improves the baseline performance to 14.0% in WER. In Table VI, we have shown more results to apply mSA-SC to these two baseline speaker-independent DNNs. For the realigned DNN (retrained by realigned state labels based on the cross-entropy criterion), we apply mSA-SC based on the cross-entropy criterion and experimental results show that mSA-SC can significantly improve the performance from 15.9% in WER to 14.2% (about 10.7% relative error reduction) in the max adaptation scheme. Furthermore, we apply mSA-SC to adapt the MMI-trained DNN. In this case, we have two different ways to learn adaptation parameters, $\mathbf{B}^l$, for mSA-SC: i) using frame-level cross entropy (CE) criterion; ii) using CE followed by sequence-level MMI criterion. Results in Table VI have shown that both adaptation criteria can significantly improve MMI-trained baseline DNN but the MMI-based adaptation yield slightly better performance. For example, using the MMI-based adaptation, mSA-SC may improve WER from 14.0% to 12.8% in the max adaptation scheme, about 8.6% relative error reduction.

TABLE VI

*Switchboard performance comparison (WER in%) of CE or MMI based mSA-SC on DNNs (6layer*2048node) re-trained by realigned state labels as well as MMI-based sequence training based on 10, 20 and maximum number adaptation utterances per speaker.*

DNN	baseline WER	adaptation criterion	adaptation utterances per speaker		
			10	20	max
CE-Realigned	**15.9**	CE	15.0	14.7	**14.2** (10.7%)
MMI-trained	**14.0**	CE	13.6	13.3	**13.1** (6.4%)
		+ MMI	13.4	13.2	**12.8** (8.6%)

3) Performance of SAT-SC on SWB: In this section, we evaluate the performance of joint speaker adaptive training method (SAT-SC) in the Switchboard task. In this experiment, we adopt the same model structure as mSA-SC in Fig. 3. For the joint SAT training, we always initialize DNN weights $\mathbf{W}^l$ using the standard RBM-based pre-training method, and examine two different initialization methods for the connection matrices $\mathbf{B}^l$: i) initialize them randomly; ii) initialize them with the final $\mathbf{B}^l$ in mSA-SC. In the joint SAT training, we use an initial learning rate of 0.1 and it is halved every epoch after the first six epochs. We stop the SAT training process after 10 epochs

in the same way as the training of speaker-independent DNN model. In SAT, the momentum is kept as 0.9, and the mini-batch size is set to 1024 and the speaker code size is 1000. In SAT training, we first run 10 epochs of CE training and also two epochs of MMI-based sequence training to further refine the joint model as described in [25].

In Table VII, we have shown recognition performance of all SAT-trained models combined with a new speaker code that is estimated from a number of adaptation utterances (10, 20 or max) for each target speaker. Results have shown that the SAT-SC with random initialization of connection weights, $\mathbf{B}^l$, gives similar performance as mSA-SC in Table VI. On the other hand, if all connection weights, $\mathbf{B}^l$, are initialized with the final value of $\mathbf{B}^l$ obtained in mSA-SC for each target speaker, SAT can further improve adaptation performance over mSA-SC. For example, SAT with MMI yields 12.1% in WER under the max adaptation scheme, which is significantly better than 12.8% in WER obtained by mSA-SC under the same condition. Moreover, it also accounts for totally 23.9% relative error reduction over the initial CE-training speaker-independent DNN baseline (15.9% in WER).

TABLE VII

*Switchboard performance (WER in%) of joint speaker adaptive training (SAT-SC) on DNN (6layer*2048node) using 10, 20 and maximum number adaptation utterances per speaker.*

initialization method	training criterion	adaptation utterances per speaker		
		10	20	max
random	CE	14.5	14.1	13.8
	+ MMI	13.3	13.1	**12.7**
using mSA-SC	CE	13.9	13.5	13.3
	+ MMI	12.6	12.4	**12.1**

4) Total training time of different adaptation methods: In this section, we compare total training time for various adaptation methods. In Table VIII, we have listed the total training time (in hours) to perform various speaker adaptation methods on the baseline DNN (6layer*2048node) using one core of a GTX690 GPU board. For fSA-SC, it includes the training time (only BP-based fine-tuning) for the baseline DNN and the extra training time to estimate adaptation NN and connection weights. For mSA-SC, it includes the same training time for the baseline DNN as well as the extra training time to estimate connection weights only. For SAT-SC, if the model is randomly initialized, the total training time only includes the joint SAT training. From the results in Table VIII, fSA-SC is quite slow to train deep DNN model for large data set and since it needs to re-estimate an extra adaptation NN using the whole training set. The mSA-SC method can significantly reduce the total training time from 275 hours down to 210 hours (about 24% faster than fSA-SC). Moreover, SAT-SC can further expedite the training process by getting rid of the baseline DNN training and the total training time for SAT-SC is about 178 hours, which is about 35% faster than that of fSA-SC. As shown in Table VIII, the training time of SAT is only slightly more than the time to train the baseline DNN. These results suggest that both mSA-SC and SAT-SC are better suited for fast adaptation of large and deep models using big data sets.

TABLE VIII

*Total training time (in hours) of different adaptation methods (fSA-SC, mSA-SC and SAT-SC) on 6-layer baseline DNN (6layer*2048nodes).*

DNN models	baseline	extra training	total
fSA-SC	150	+ 125	275
mSA-SC	150	+ 60	210
SAT-SC	-		178

5) Comparison with other adaptation methods: In this section, we compare our speaker-code based adaptation methods with other adaptation methods proposed for neural network in the literature. First of all, we compare our SC methods with previous methods as mentioned in section I, such as LIN [12], LHN [13], LOH [14] and fMLLR [39], in a supervised adaptation setting with 10 or 20 utterances per speaker. The results in Table IX have shown that these simpler speaker adaptation methods cannot bring significant improvement with a small amount of adaptation data when dealing with deep and large neural networks while our proposed methods are much more effective.

More recently, in [40], it has proposed to use i-vectors to replace the speaker codes in our previous work [21]. In that method, i-vectors are separately trained from a standard procedure for all speakers in both training and test sets as in [41]. The estimated i-vectors are directly used as speaker representations in training to estimate connection weights, $\mathbf{B}^l$. During the test stage, the pre-trained i-vectors are directly used without re-estimation. In this experiment, we compare our proposed mSA-SC method with the i-vector based approach in [40], where i-vectors are connected to all DNN layers in the same way as mSA-SC in Fig. 3. As shown in Table X, i-vector based adaptation achieves slightly better performance than our mSA-SC method, where all speaker codes are randomly initialized in both training and adaptation. However, if we use the pre-estimated i-vectors as initialization for all speaker codes, our mSA-SC method consistently outperforms the i-vector method that does not optimize speaker representations based on DNN models.

TABLE IX

*Switchboard performance comparison (WER in %) of different speaker adaptation methods on DNNs (6layer*2048node) based on 10, 20 adaptation utterances per speaker.*

method	baseline	adaptation utterances	
		10	20
fSA-SC	16.2	15.7	15.5
mSA-SC	16.2	**15.2**	**15.2**
LIN [12]	16.2	15.9	15.8
LHN [13]	16.2	16.1	15.9
LON [14]	16.2	16.2	16.1
fMLLR [39]	16.2	16.7	16.0

TABLE X

*Switchboard performance comparison (WER in %) between mSA-SC (using random initialization or iVectors as initialization for speaker codes) and the iVector method [40] on DNNs (6layer*2048node) based on 10, 20 and max adaptation utterances per speaker. One 1000-dimension iVector is estimated for each speaker in training and testing sets.*

method	baseline	adaptation utterances		
		10	20	max
iVector [40]	16.2	15.0	14.9	14.7
mSA-SC (random)	16.2	15.2	15.2	14.9
mSA-SC (iVector)	16.2	**14.8**	**14.7**	**14.4**

VI. Conclusion

In this paper, we have proposed a general method to rapidly adapt deep neural networks (DNN) using a small amount adaptation data. The fast adaptation method relies on some condition codes that are fed into different layers of pre-trained DNN. In this work, this adaptation scheme is applied to fast speaker adaptation of DNNs for speech recognition. We have proposed three different methods to perform supervised speaker adaptation based on the so-called speaker codes: i) nonlinear feature normalization in feature space; ii) direct DNN model adaptation in model space; iii) joint speaker adaptive training. All these three methods have been evaluated in two standard speech recognition tasks, namely TIMIT phone recognition and large vocabulary speech recognition in the Switchboard task. Experimental results have shown that all of these methods are quite effective to adapt very large DNN models based on only a small number of adaptation utterances per speaker. For example, the Switchboard results have shown that the proposed speaker-code-based adaptation can achieve about 8-10% relative reduction in word error rate with only a few dozen of adaptation utterances per speaker. In the Switchboard task, we have achieved a very low word error rate (12.1%) after speaker adaptation using sequence training, which is very close to the best performance reported in this task [1].

Acknowledgment

This work was partially funded by the National Nature Science Foundation of China (Grant No. 61273264) and the National 973 program of China (Grant No. 2012CB326405), as well as a discovery grant from Natural Sciences and Engineering Research Council of Canada (NSERC). The authors would like to thank Jia Pan, Cong Liu from iFlytek Research, Anhui USTC iFLYTEK Corporation Limited, Hefei, China for sharing their full sequence training codes with us for some experiments. The authors also want to thank Changqing Kong from National Engineering Laboratory of Speech and Language Information Processing, University of Science and Technology of China for his help in some experiments.

References

[1] T. N. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. of IEEE Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[2] J. L. 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. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995.

[4] V. Digalakis, D. Rtischev, and L. Neumeyer, "Speaker adaptation using constrained estimation of gaussian mixtures," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 5, pp. 357–366, 1995.

[5] L. Lee and R.C. Rose, "Speaker normalization using efficient frequency warping procedures," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 1996, vol. 1, pp. 353–356.

[6] H. Jiang, F. Soong, and C.-H. Lee, "Hierarchical feature compensation and its application to hands-free speech recognition," in *Proc. of ISCA International Workshop on Hands-Free Speech Communication*, 2001.

[7] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. of INTERSPEECH*, 2011, pp. 437–440.

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

[9] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A.-R. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[10] J. Pan, Cong Liu, Z. Wang, Y. Hu, and H. Jiang, "Investigation of deep neural networks (DNN) for large vocabulary continuous speech recognition: Why DNN surpasses GMMS in acoustic modelling," in *Proc. of International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2012.

[11] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[12] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system," in *Proc. of EUROSPEECH*, 1995.

[13] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. De Mori, "Linear hidden transformations for adaptation of hybrid ANN/HMM models," *Speech Communication*, vol. 49, no. 10, pp. 827–835, 2007.

[14] B. Li and K. C. Sim, "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems," in *Proc. of INTERSPEECH*, 2010.

[15] J. Stadermann and G. Rigoll, "Two-stage speaker adaptation of hybrid tied-posterior acoustic models," in *Proc. of IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*, 2005.

[16] S. M. Siniscalchi, J. Li, and C.-H. Lee, "Hermitian polynomial for speaker adaptation of connectionist speech recognition systems," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2152–2161, 2013.

[17] 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. of IEEE Spoken Language Technology Workshop (SLT)*, 2012.

[18] 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. of IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[19] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Proc. of IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[20] Z. Tuskea, R. Schlutera, and H. Ney, "Deep hierarchical bottleneck MRASTA features for LVCSR," in *Proc. of IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*, 2013.

[21] 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. of IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*, 2013.

[22] O. Abdel-Hamid and H. Jiang, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition," in *Proc. of INTERSPEECH*, 2013.

[23] S. Xue, O. Abdel-Hamid, H. Jiang, and L. Dai, "Direct adaptation of hybrid DNN/HMM model for fast speaker adaptation in LVCSR based on speaker code," in *Proc. of IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*, 2014.

[24] J. S. Bridle and S. J. Cox, "Recnorm: simultaneous normalization and classification applied to speech recognition," in *Advances in Neural Information Processing Systems 3 (NIPS)*, 1991, vol. 3, pp. 234–240.

[25] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *Proc. of IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*, 2013.

[26] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *Proc. of International Conference on Spoken Language Processing (ICSLP)*, 1996, vol. 2, pp. 1137–1140.

[27] T. Anastasakos, J. McDonough, and J. Makhoul, "Speaker adaptive training: a maximum likelihood approach to speaker normalization," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1997, vol. 2, pp. 1043–1046.

[28] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[29] P. Zhou, L. Dai, and H. Jiang, "Sequence training of multiple deep neural networks for better performance and faster training speed," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[30] P. Zhou, L. Dai, H. Jiang, Y. Hu, and Q. Liu, "A state-clustering based multiple deep neural networks modelling approach for speech recognition," *submitted to IEEE Trans. on Acoustics, Speech and Signal Processing*, 2013.

[31] H. Jiang and L. Deng, "A robust compensation strategy for extraneous acoustic variations in spontaneous speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 10, no. 1, pp. 9–17, 2002.

[32] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.

[33] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[34] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, pp. 153, 2007.

[35] A.-R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modelsling using deep belief networks," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[36] Y. Bao, H. Jiang, C. Liu, Y. Hu, and L. Dai, "Investigation on dimensionality reduction of concatenated features with deep neural network for LVCSR systems," in *Proc. of IEEE International Conference on Signal Processing (ICSP)*, 2012.

[37] Y. Bao, H. Jiang, L. Dai, and C. Liu, "Incoherent training of deep neural networks to de-correlate bottleneck features for speech recognition," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[38] S. Zhang, Y. Bao, P. Zhou, H. Jiang, and L. Dai, "Improving deep neural networks for LVCSR using dropout and shrinking structure," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[39] F. Seide, G. Li, Xie Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011, pp. 24–29.

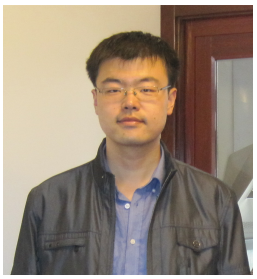
[40] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2013.

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

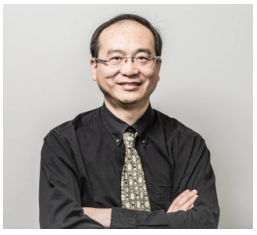
Biographies of the Authors

Shaofei Xue was born in China in 1988. He received the B.S. degree in Electrical Engineering from the University of Science and Technology of China (USTC), Hefei, China in 2006. He is currently a Ph.D. candidate in USTC working on speech recognition. His current research interests include deep learning for speech recognition and other pattern recognition applications.

Ossama Abdel-Hamid received his B.Sc. with honours in Information Technology from Cairo University, Egypt in 2002, from where he received his M.Sc. in 2007. He is currently a Ph.D. candidate at Computer Science



department, York University, Canada. He joined the speech research group at RDI, Egypt in the period from 2003 to 2007. Moreover, he had internships at IBM Watson research centre in 2006, Google in 2008, and Microsoft Research in 2012. His current research focuses on improving automatic speech recognition performance using deep learning methods.



Hui Jiang (M'00-SM'11) received B.Eng. and M.Eng. degrees from University of Science and Technology of China (USTC), and his Ph.D. degree from the University of Tokyo, Tokyo, Japan in September 1998, all in electrical engineering. From 2000 to 2002, he worked in Dialogue Systems Research, Multimedia Communication Research Lab, Bell Labs, Lucent Technologies Inc., Murray Hill, NJ. He joined Department of Computer Science and Engineering, York University, Toronto, Canada as an Assistant Professor on fall 2002 and was promoted to Associate Professor in 2007 and to Full Professor in 2013. He served as an associate editor for IEEE Trans. on Audio, Speech and Language Processing between 2009 and 2013. His current research interests lie in machine learning methods with applications to speech and language processing.

Li-Rong Dai was born in China in 1962. He received the B.S. degree in electrical engineering from Xidian University, Xian, China, in 1983 and the M.S. degree from Hefei University of Technology, Hefei, China, in 1986, and the Ph.D. degree in signal and information processing from the University of Science and Technology of

China (USTC), Hefei, in 1997. He joined University of Science and Technology of China in 1993. He is currently a Professor of the School of Information Science and Technology, USTC. His current research interests include speech synthesis, speaker and language recognition, speech recognition, digital signal processing, voice search technology, machine learning, and pattern recognition. He has published more than 50 papers in these areas.

Liu Qingfeng was born in China in 1973. He received B.S degree in Electronic Engineering in 1995 and M.S. degree in communication and Electronic system in 1998, and Ph.D. degree in signal and information processing in 2003, all from University of Science and Technology of China (USTC). Dr. Liu founded iFLYTEK in 1999 and has been the president since then and assumed the Chairman of the Board in April 2009. Dr. Liu is currently Director of National Engineering Laboratory of Speech and Language Information Processing, USTC and an adjunct professor of USTC. His current research interests lie in speech technology and its industrial applications, particularly for mobile Internet. He has published more than 20 papers in these areas.