Speaker Adaptive Training of Deep Neural Network Acoustic Models using I-vectors

Yajie Miao, Hao Zhang, and Florian Metze

***Abstract*—In acoustic modeling, speaker adaptive training (SAT) has been a long-standing technique for the traditional Gaussian mixture models (GMMs). Acoustic models trained with SAT become independent of training speakers and generalize better to unseen testing speakers. This paper ports the idea of SAT to deep neural networks (DNNs), and proposes a framework to perform feature-space SAT for DNNs. Using i-vectors as speaker representations, our framework learns an adaptation neural network to derive speaker-normalized features. Speaker adaptive models are obtained by fine-tuning DNNs in such a feature space. This framework can be applied to various feature types and network structures, posing a very general SAT solution. In this work, we fully investigate how to build SAT-DNN models effectively and efficiently. First, we study the optimal configurations of SAT-DNNs for large-scale acoustic modeling tasks. Then, after presenting detailed comparisons between SAT-DNNs and the existing DNN adaptation methods, we propose to combine SAT-DNNs and model-space DNN adaptation during decoding. Finally, to accelerate learning of SAT-DNNs, a simple yet effective strategy, frame skipping, is employed to reduce the size of training data. Our experiments show that compared with a strong DNN baseline, the SAT-DNN model achieves 13.5% and 17.5% relative improvement on word error rates (WERs), without and with model-space adaptation applied respectively. Data reduction based on frame skipping results in 2× speed-up for SAT-DNN training, while causing negligible WER loss on the testing data.**

***Index Terms*—Deep neural networks, speaker adaptive training, acoustic modeling.**

I. Introduction

IN recent years, deep neural networks (DNNs) have achieved tremendous success in acoustic modeling. On a wide range of large vocabulary continuous speech recognition (LVCSR) tasks, DNNs have shown better performance than the traditional Gaussian mixture models (GMMs) [1], [2], [3], [4]. Although displaying superior generalization ability than GMMs [5], DNN models still suffer from the mismatch between acoustic models and testing speakers. As is the case with GMMs, DNN models experience a degradation of recognition accuracy when ported from training speakers to unseen testing speakers. For GMM models, speaker adaptation has proven to be effective in mitigating the effects of this mismatch [6], [7]. In general, speaker adaptation modifies speaker-independent (SI) models towards particular testing speakers or transforms the features of testing speakers towards the SI models. In the context of DNNs, researchers have also proposed a considerable number of methods for speaker adaptation. The most straightforward method is to re-update the SI model, or certain layers of the model, on the adaptation data of each testing speaker [8], [9]. In [10], [2], [11], the SI-DNN models are augmented with additional speaker-dependent (SD) layers that are trained on the adaptation data. Also, adaptation can be naturally achieved by training (and decoding) DNN models using speaker-adaptive (SA) features or features enriched with speaker-specific information [2], [12], [13].

Another technique closely related with speaker adaptation is speaker adaptive training (SAT) [14], [15]. When carried out in the feature space, SAT performs adaptation on the training set and projects training data into a speaker-normalized space. Parameters of the acoustic models are estimated in this new feature space. Acoustic models trained this way become independent of specific training speakers and thus generalize better to unseen testing speakers. In practice, SAT models generally give better recognition results than SI models, when speaker adaptation is applied to both of them. For GMM models, SAT has been a well-studied, long-standing technique. In comparison, SAT for DNN models has attracted less attention. Training DNNs with SA features [16], [2] or additional speaker-specific information [13], [17] can be naturally taken as a form of SAT. With the speaker-code adaptation scheme, [18] accomplishes SAT of DNNs by jointly learning the SD speaker codes and the SI DNN models. In [19], certain layers of the DNNs are assigned as the SD layers and trained on the speaker-specific basis, in order to model speaker variability. The other layers are trained by selecting the corresponding SD layers for different speakers.

In this paper, we propose a novel framework to perform feature-space SAT for DNNs. In our framework, building of *SAT-DNN* models starts from an initial SI-DNN model which has been trained over the entire training set. Then, we treat i-vectors [20], [21] extracted at the speaker level as a compact representation of each speaker's acoustic characteristics. An *adaptation neural network* is learned to convert i-vectors to speaker-specific linear feature shifts. Adding the shifts to the original DNN input vectors (e.g., MFCCs) produces a speaker-normalized feature space, in which the parameters of the SI-DNN are further updated. This finally gives us the SAT-DNN model. The adaptation network, together with i-vectors, models speaker variability, which enables SAT-DNNs to focus on modeling speech variability. Compared with the previous proposals [16], [2], [18], [19], our framework poses a general solution in that its applicability does not depend on the specific choices of the original input features and the network structures. For example, although called SI-DNN, the initial DNN model for SAT-DNN can be trained with non-

Manuscript submitted April 23, 2015. Y. Miao, H. Zhang, and F. Metze are with Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pttsburgh, PA 15213, USA. E-mail: {ymiao, haoz1, fmetze}@cs.cmu.edu

adaptive features (e.g., filterbanks) or adaptive features (e.g., fMLLRs). Also, in addition to DNNs, our framework can be applied readily to other types of acoustic models such as convolutional neural networks (CNNs) [22], [23], [24], [25] and recurrent neural networks (RNNs) [26], [27], [28], [29], [30]. The generality of our framework will be empirically demonstrated in the experiments.

During decoding, the SAT-DNN model is adapted simply by extracting the i-vector of each testing speaker, feedforwarding the i-vector through the adaptation network, and adding the generated feature shifts to the original feature vectors. The SAT-DNN model is finally decoded in the speaker-normalized feature space. Speaker adaptation in this manner has two advantages. First, extraction of i-vectors requires no transcriptions or word hypotheses to be available. This eliminates the need of the first-pass decoding and therefore achieves efficient unsupervised adaptation. Second, unlike most of the existing adaptation methods [10], [2], [11], [8], adaptation of SAT-DNNs does not perform network fine-tuning on the adaptation data. As a result, speaker adaptation becomes robust to hypotheses errors, especially when the first-pass hypotheses have high word error rates (WERs).

This paper reorganizes, expands and completes our previous development of SAT-DNN in [31], [32].[1] We fully investigate how to apply the SAT-DNN framework to large-scale acoustic modeling tasks effectively and efficiently.

- First, we determine the optimal configurations for SAT-DNNs. In particular, we make an attempt to bridge i-vector extraction with DNN training: during i-vector extraction, MFCCs are replaced with features from a DNN architecture that has been trained to classify context-dependent (CD) states. This adds phonetic discrimination to i-vectors and is empirically shown to benefit building of SAT-DNNs.
- Second, we present detailed performance comparisons between SAT-DNNs and the existing DNN adaptation methods. For better decoding results, model-space DNN adaptation is additionally applied atop of SAT-DNNs. Specifically, after i-vector based (feature-space) adaptation, the SAT-DNN model is further adapted in the speaker-normalized feature space using the recently-proposed learning hidden unit contributions (LHUC) method [33].
- Third, we study a data reduction strategy to speed up training of SAT-DNNs. This strategy, referred to as *frame skipping*, reduces the amount of training data for SAT-DNNs while keeping the number of i-vectors (speakers) unchanged.

Our experiments with a benchmark TED Talk dataset show that compared with the DNN baseline, the SAT-DNN model achieves 13.5% relative WER improvement. Additionally applying the LHUC-based model adaptation enlarges the improvement to 17.5% relative. Using frame skipping results in $2\times$ speed up for SAT-DNN training, while at the same time only 1.7% relative WER degradation is observed on the testing set.

[1]Our code is released at http://www.cs.cmu.edu/~ymiao/satdnn.html. Readers can reproduce our baseline DNN and SAT-DNN results with the code.

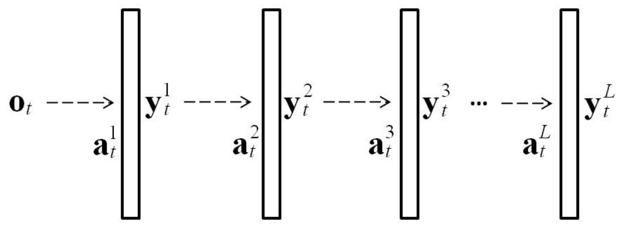


Fig. 1. Architecture of the DNN model used in this work. The vector $\mathbf{o}_t$ represents the inputs to the DNN. At the i-th layer, $\mathbf{a}_t^i$ is the pre-nonlinearity outputs and $\mathbf{y}_t^i$ is the outputs with the activation function applied.

The remainder of the paper is organized as follows. In Section II, we review previous work related to our paper. Section III introduces our framework for SAT of DNNs, and Section IV presents the experiments. Section V studies the comparisons and combinations of SAT-DNNs and DNN adaptation approaches. We examine acceleration of SAT-DNN training in Section VI and have our conclusions in Section VII.

II. Related Work

In this section, we give a brief review of DNN acoustic models, as well as the existing DNN SAT and speaker adaptation methods. Moreover, since building of SAT-DNNs relies on speaker i-vectors, we also show some basics regarding i-vector extraction.

A. DNN Acoustic Models

The architecture of the DNN we use is shown in Fig. 1. A DNN is a multilayer perceptron (MLP) which consists of many hidden layers before the softmax classification layer. Each hidden layer computes the outputs of hidden units given the input vectors. We denote the feature vector at the t-th frame as $\mathbf{o}_t$. Normally $\mathbf{o}_t$ is the concatenation of multiple neighbouring frames surrounding t. The quantities shown in Fig. 1 can be computed as:

$$\mathbf{a}_t^i = \mathbf{W}_i \mathbf{x}_t^i + \mathbf{b}_i \qquad \mathbf{y}_t^i = \sigma(\mathbf{a}_t^i) \qquad 1 \le i \le L \tag{1}$$

where L is the total number of layers, the weight matrix $\mathbf{W}_i$ connects the i-1-th and i-th layers, and $\mathbf{b}_i$ is the bias vector of the i-th layer. The inputs to the i-th layer $\mathbf{x}_t^i$ can be formulated as:

$$\mathbf{x}_t^i = \begin{cases} \mathbf{o}_t & i = 1 \\ \mathbf{y}_t^{i-1} & 1 < i \le L \end{cases} \tag{2}$$

For hidden layers $1 \le i < L$, the activation function $\sigma(x)$ takes the form of the standard logistic functions (sigmoid and tanh) or other newly-developed functions (e.g., rectifier [34] and maxout [35], [36]). The final classification layer $i = L$ uses the softmax function.

When applied as a *hybrid model*, the DNN is trained to classify each speech frame to CD HMM states. Suppose that we use the negative cross-entropy as the loss function and the training set contains T frames. DNN training involves minimizing the following objective:

$$\mathbb{L} = -\sum_{t=1}^{T}\sum_{j=1}^{J} \mathbf{g}_t(j) \log \mathbf{y}_t^L(j) \tag{3}$$

where J is the total number of CD states (i.e., classification targets), $\mathbf{g}_t$ is the one-hot label vector on frame t which is obtained via forced alignment with an existing GMM/DNN model, and $\mathbf{y}_t^L$ is the output vector of the softmax layer. Error back-propagation is commonly adopted to optimize this objective. The gradients of the model parameters can be derived from the derivatives of the objective function with respect to the pre-nonlinearity outputs $\mathbf{a}_t^i$. At the softmax layer, the error vector for frame t is:

$$\boldsymbol{\epsilon}_t^L = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_t^L} = \mathbf{y}_t^L - \mathbf{g}_t \tag{4}$$

Assume that each of the hidden layers uses the logistic sigmoid activation function. The errors back-propagated to the i-th hidden layer can be represented as

$$\boldsymbol{\epsilon}_t^i = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_t^i} = (\mathbf{W}_{i+1}^T \boldsymbol{\epsilon}_t^{i+1}) \odot \mathbf{y}_t^i \odot (1 - \mathbf{y}_t^i) \tag{5}$$

where $\odot$ represents element-wise multiplication. In practice, we use the mini-batch based stochastic gradient descent (SGD) algorithm as the optimizer. In this case, model parameters are updated with gradients accumulated over the entire mini-batches.

Outputs from the whole DNN architecture represent the posterior probabilities of CD HMM states given the input $\mathbf{o}_t$. During decoding, we in fact need the emission probability of the feature vector with respect to each state. According to the Bayes rule, the observation probability given each state j can be computed as:

$$p(\mathbf{o}_t|j) \propto \mathbf{y}_t^L(j)/p(j) \tag{6}$$

where $p(j)$ is the prior probability of state j which can be estimated from the alignment of the training data.

B. Speaker Adaptation and SAT of DNNs

Speaker adaptation acts as an effective procedure to reduce mismatch between training and testing conditions. Previous work has proposed a number of techniques for speaker adaptation of DNN models. These methods can be categorized into 3 classes. The first class augments the SI-DNN model with additional SD layers, and the parameters of these layers are learned on the target speakers. In [10], [2], a layer is inserted between the input features and the first hidden layer. This additional layer plays a similar role to the feature-space maximum likelihood linear regression (fMLLR) transforms estimated for GMMs. Alternatively, the SD layer can be inserted after the last hidden layer [10]. Yao *et al.* [11] demonstrate that transforming the outputs of the final hidden layer is equivalent to adapting the parameters of the softmax layer. After carrying out singular value decomposition (SVD) over DNN weight matrices, Xue *et al.* [37] perform adaptation by learning SD matrices that are inserted between the decomposed weight matrices. In the recently proposed learning hidden unit contributions (LHUC) approach [33], a SD vector is attached to every hidden layer of the SI-DNN and learned on a particular testing speaker. These SD vectors are applied to the SI-DNN hidden outputs with element-wise multiplication, regulating the contributions of hidden units.

The second class of adaptation methods trains (and decodes) DNNs over SA features. For example, features transformed with vocal tract length normalization (VTLN) and fMLLR have been applied successfully as DNN features, showing notable advantage over non-adaptive features [2], [12]. Another way of normalizing speaker variability is to augment the input features with additional speaker-specific information. Speaker i-vectors [20], [21] have been explored for this purpose. For example, Saon *et al.* [13] append i-vectors to the original features at each speech frame. The effectiveness of incorporating i-vectors is further verified in [38], [17]. I-vector based adaptation is further developed in [39], [40] where separate vectors are used to represent finer-grained acoustic factors such as speakers, environments and noise. In [25], Sainath *et al.* make an attempt to incorporate i-vectors into CNNs models. The incorporation is achieved either by appending i-vectors directly to every feature map from the convolution layers, or by adopting the joint CNN/DNN architecture introduced in [23]. In [41], [42], the speaker representation (referred to as speaker codes) is learned on each speaker through network fine-tuning, instead of being extracted prior to DNN training.

The third class adapts the parameters of the SI-DNN model directly, without changing the architecture of the SI-DNN. For instance, [8] examines how the performance of DNN adaptation is affected by updating different parts (the input layer, the output layer, or the entire network) of the SI-DNN. Updating the entire DNN may suffer from overfitting, especially when the amount of adaptation data is limited. To address this issue, Yu *et al.* [9] propose to regularize speaker adaptation with the Kullback-Leibler (KL) divergence between the output distributions from the SI and the adapted DNNs. This regularizer prevents the parameters of the adapted DNN from deviating too much from the SI-DNN. Price *et al.* [43] add a softmax layer with context-independent (CI) states on top of the softmax layer with CD states. This helps alleviate the problem of rare CD classes having no or few examples on the adaptation data. To enhance the robustness of adaptation, Huang *et al.* [44] estimate the adaptation parameters via maximum a posteriori (MAP) linear regression, which naturally incorporates prior knowledge into the adaptation process. In [45], instead of model parameters, the shape of the activation function used in the SI-DNN is adjusted to match the testing conditions.

In comparison to speaker adaptation, past work has made fewer attempts on SAT of DNNs. Training DNNs with SA features [16], [2], [12] or additional speaker-specific information [13], [38], [17] can be treated as a form of SAT. In [18], Xue *et al.* append speaker codes [41], [42] to the hidden and output layers of the DNN model. SAT is achieved by jointly learning the speaker-specific speaker codes and the SI-DNN.

In [19], speaker variability is normalized by allocating certain layers of the DNN as the SD layers that are learned on a speaker-specific basis. Over different speakers, the other layers are adaptively trained by picking the SD layer corresponding to the current speaker. Although showing promising results, the application of these proposals is constrained to specific model structures. For example, the approach in [18] is not applicable to CNNs because the outputs from the hidden convolution layers are organized in the form of an array of feature maps. It is infeasible to fuse the speaker codes and the hidden convolutional activation via simple concatenation. Also, in these methods, adaptation of the resulting SAT models generally needs multiple decoding passes, which undermines the decoding efficiency. This motivates us to propose a more general solution for SAT of DNNs. The framework we present in this paper can be applied to different feature types and model structures. Furthermore, due to the incorporation of speaker i-vectors, speaker adaptation of the SAT models becomes both efficient and robust.

C. I-vector Extraction

The introduction of the i-vector paradigm has resulted in significant advancement in the speaker recognition and verification community [20], [21]. The i-vector approach differs from the earlier joint factor analysis (JFA) [46] in that it has a single variability subspace to model different types of variability in the speech signal. We assume to have a GMM model, referred to as universal background model (UBM), which consists of K Gaussian components. The t-th frame $\mathbf{o}_t$ from the s-th speech segment is formulated to be generated from the UBM model as follows:

$$\mathbf{o}_t \sim \sum_{k=1}^{K} r_t(k) N(\boldsymbol{\mu}_k + \mathbf{T}_k \mathbf{w}_s, \boldsymbol{\Sigma}_k) \tag{7}$$

where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the mean vector and covariance matrix of the k-th Gaussian component, the total variability matrices $\mathbf{T}_k$ span a subspace of the shifts by which the UBM means are adapted to particular segments, $r_t(k)$ represents the posterior probability of the k-th Gaussian given the speech frame. The latent vector $\mathbf{w}_s$ follows a standard normal distribution and describes the coordinates of the mean shift in the total variability subspace. The maximum a posterior (MAP) point estimate of the vector $\mathbf{w}_s$, i.e., the i-vector, represents salient information about all types of variability in the speech segment. The readers can refer to [20], [21] for more details.

Given a collection of speech segments, the UBM model can be trained using the standard maximum likelihood estimation (MLE). To build the i-vector model, sufficient statistics are accumulated on each speech frame based on the posteriors with respect to the UBM. These statistics are used to learn the total variability matrices and extract the i-vectors. Then, a scoring model, e.g., probabilistic linear discriminant analysis (PLDA), can be applied to the i-vectors for speaker recognition/verification. When extracted at the speaker (rather than segment) level, i-vectors provide a low-dimensional representation of the acoustic characteristics of individual speakers. Previous work [47], [13], [38], [17] has applied i-vectors successfully to speaker adaptation of both GMM and DNN acoustic models. In this paper, we leverage i-vectors to perform adaptive training of DNNs. It is worth noting that although called SAT, adaptive training performed in this work also pertains to other types of variability (noise, channel, etc.) that is naturally encapsulated by the i-vector representations.

III. Speaker Adaptive Training of DNNs

This section introduces our SAT-DNN framework. We first present the overall architecture and then give details about the two training steps: training the adaptation network and updating the DNN parameters. Finally, we elaborate on how to decode the SAT-DNN models.

A. Architecture of SAT-DNNs

For GMM-based acoustic modeling, speaker-specific fMLLR transforms are estimated on the training speakers if SAT is performed in the feature-space. The GMM parameters are then updated in the fMLLR-transformed feature space. We port this idea to DNN models, and the SAT-DNN architecture is shown in Fig. 2. Suppose that an i-vector has been extracted for each speaker as described in Section II-C, where the segment now contains all the frames from the speaker. As with SAT of GMMs, SAT of DNNs starts from a SI-DNN model (on the right of Fig. 2) that has been well trained over the entire training set. Two steps are then taken to build the SAT-DNN model. First, with the SI-DNN fixed, a smaller adaptation neural network (on the left of Fig. 2) is learned to take i-vectors as inputs and generate speaker-specific linear shifts to the original DNN feature vectors. In Fig. 2, the t-th input vector $\mathbf{o}_t$ comes from speaker s whose i-vector is denoted as $\mathbf{i}_s$. The layers of the adaptation network are indexed by $-I$, $-I+1$, ..., -2, -1, where I is the total number of layers in the adaptation network. The values attached to the adaptation network can be computed as:

$$\mathbf{a}_s^i = \mathbf{W}_i \mathbf{x}_s^i + \mathbf{b}_i \qquad \mathbf{y}_s^i = \phi(\mathbf{a}_s^i) \qquad -I \leq i \leq -1 \tag{8}$$

where ϕ is the activation function of each layer in the adaptation network, $\mathbf{x}_s^i$ is the inputs to the i-th layer and can be represented as:

$$\mathbf{x}_s^i = \begin{cases} \mathbf{i}_s & i = -I \\ \mathbf{y}_s^{i-1} & -I < i \leq -1 \end{cases} \tag{9}$$

After the adaptation network is trained, the speaker-specific output vector $\mathbf{y}_s^{-1}$ is added to every original feature vector $\mathbf{o}_t$ from speaker s:

$$\mathbf{z}_t = \mathbf{o}_t \oplus \mathbf{y}_s^{-1} \tag{10}$$

where $\oplus$ represents element-wise addition. This gives us a new feature space which is expected to be more speaker normalized.

The second step involves updating the parameters of the DNN model in the new feature space. We keep the adaptation network unchanged. The SI-DNN is fine-tuned by taking $\mathbf{z}_t$ as the new feature vector at each frame t. This finally generates the SAT-DNN model that is more independent of

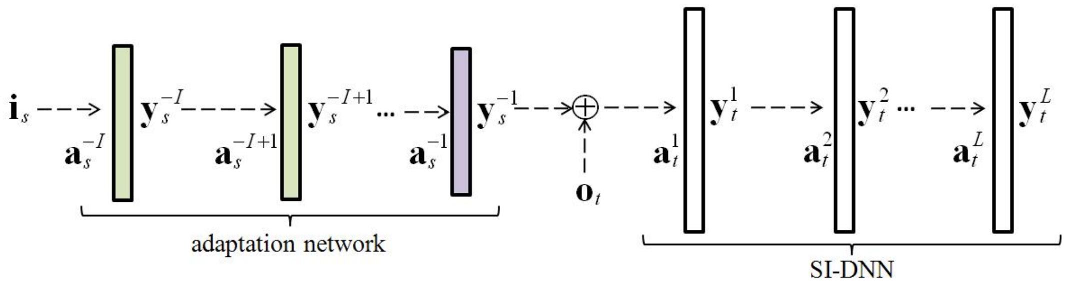


Fig. 2. Architecture of the SAT-DNN model. On the right is the SI-DNN which serves as the starting point for SAT-DNN. On the left is the adaptation neural network whose inputs are speaker i-vectors $\mathbf{i}_s$. In the adaptation network, the green layers are the hidden layers, whereas the purple layer is the output layer. At the i-th layer, $\mathbf{a}_s^i$ is the pre-nonlinearity outputs and $\mathbf{y}_s^i$ is the activations.

specific speakers. Note that during this updating step, the DNN parameters use the original SI-DNN as the initial values, instead of being learned from scratch.

The formulation of the adaptation network outputs as linear feature shifts imposes two constraints. First, the output layer of the adaptation network has to use the linear identity activation function $\phi(x) = x$ by which we are not restricting the direction and value range of the feature shifts. Second, the number of units at the output layer (i.e., the dimension of $\mathbf{y}_s^{-1}$) has to equal the dimension of the original feature vector $\mathbf{o}_t$. Both training of the adaptation network and updating of the DNN model can be realized with the standard error back-propagation. We will give more details in the following two subsections.

Although the linear shifts are applied to the input features, the SAT-DNN framework can be extended easily to learn adaptation networks on the hidden layers. For example, we can apply an adaptation network to the first hidden layer of the SI-DNN. SAT-DNN in this case resembles the approach described in [13] which appends speaker i-vectors to input features for adaptation. In our experiments (Section V-A), SAT-DNN is shown to outperform this feature-concatenation method by a large margin. This indicates that learning linear shifts to the hidden layers is less effective than to the input features. A further extension is to have a separate adaptation network for every layer (input and hidden) of the SI-DNN. However, in this case, the number of model parameters increases dramatically, which makes the model training vulnerable to overfitting. Due to these considerations, this paper constrains the application of the adaptation network to the input features.

B. Training of the Adaptation Network

The adaptation network is learned in a supervised manner, with errors back-propagated from the SI-DNN. The objective function remains to be Equation (3), i.e., the negative cross-entropy computed with the outputs from the SI-DNN softmax layer and the ground-truth labels. However, the inputs of the DNN are now the new input features $\mathbf{z}_t$, instead of the original features $\mathbf{o}_t$. Our goal is to optimize the objective with respect to the parameters of the adaptation network. The derivatives of the objective function with respect to the new feature vector $\mathbf{z}_t$ can be written as:

$$\frac{\partial \mathbb{L}}{\partial \mathbf{z}_t} = \mathbf{W}_1^T \boldsymbol{\epsilon}_t^1 \tag{11}$$

where $\boldsymbol{\epsilon}_t^1$ represents the error vector back-propagated to the first hidden layer of the SI-DNN, $\mathbf{W}_1$ is the weight matrix connecting the input features and the first hidden layer. At the output layer of the adaptation network, we have $\mathbf{y}_s^{-1} = \mathbf{a}_s^{-1}$ because of the identity activation function. This output layer has the error vector as:

$$\boldsymbol{\epsilon}_s^{-1} = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_s^{-1}} = \frac{\partial \mathbb{L}}{\partial \mathbf{y}_s^{-1}} = \frac{\partial \mathbb{L}}{\partial \mathbf{z}_t} \tag{12}$$

which is further back-propagated into the adaptation network. The parameters of the adaptation network are updated with gradients derived from the error vectors. We can see that these gradients depend not only on the speaker i-vectors but also on the DNN input features $\mathbf{o}_t$. Therefore, the training data of the adaptation network consist of the combinations of i-vectors and their corresponding speech frames. The number of training examples equals the number of speech frames, rather than the number of speakers. Although we also get the gradients of the SI-DNN parameters, these parameters are not updated.

Our assumption about the adaptation network is that it takes i-vectors as extra information and normalizes the DNN input features into a speaker-independent space. We qualitatively examine the validity of this assumption by visualizing the original and normalized features. Specifically, from the training set used in our experiments, we select two arbitrary speakers. For each speaker, his/her speech frames $\mathbf{o}_t$ are projected to 2-dimensional vectors using principal component analysis (PCA). These 2 dimensions represent the coordinates along which the original feature vectors display the largest variation. In Fig. 3, we plot the data points with the horizontal axis representing the first dimension and the vertical axis representing the second dimension. The red and blue dots belong to the two speakers respectively. Following the same procedures, we visualize the normalized feature vectors $\mathbf{z}_t$ from the same two speakers in Fig. 4.

We can see in Fig. 3 that the feature spaces of the two speakers show discrepancy, indicating that variability exists between the two speakers. In Fig. 4, the non-overlapping area shrinks, especially along the first dimension (the horizontal axis) where the data points exhibit larger variability. This reveals that compared with the original features $\mathbf{o}_t$, the new feature vectors $\mathbf{z}_t$ from different speakers tend to fall into the same space. The adaptation network truly acts to normalize inter-speaker variability emerging from speech signals.

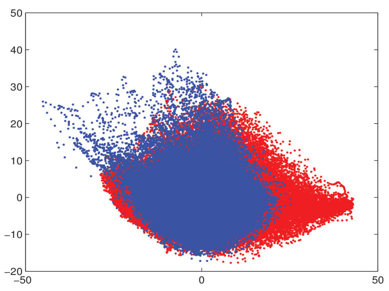


Fig. 3. Visualization of the original features $\mathbf{o}_t$ on a 2D plan. The two training speakers are represented with blue and red dots respectively.

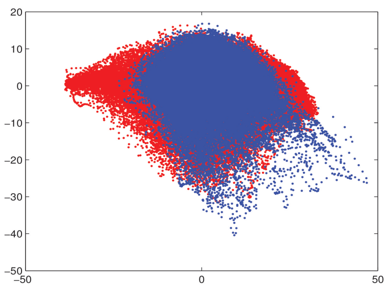


Fig. 4. Visualization of the speaker-adapted features $\mathbf{z}_t$ on a 2D plan. The data points are from the same two speakers as in Fig. 3.

C. Updating of the DNN Model

After the adaptation network is trained, updating of the DNN model is straightforward to accomplish. Parameters of the DNN are initialized with parameters of the SI-DNN model and fine-tuned with the negative cross-entropy objective. The only difference is that the inputs of the DNN are now the speaker-normalized features $\mathbf{z}_t$. Parameters of the adaptation network are kept fixed during this step. When fine-tuning terminates, we get the final SAT-DNN model. Unlike in SAT of GMMs, the two steps in building of SAT-DNN models are iterated only once. Performing more iterations is found to hurt the recognition accuracy in our experiments.

From its training process, we can see that our SAT-DNN approach poses a general framework. It does not depend on specific choices of the original DNN input features. Although called SI-DNN, the initial DNN model can be trained on either SI features (e.g., filterbanks, MFCCs, etc.) or speaker-adapted features (e.g., fMLLRs, filterbanks with VTLN, etc.). Moreover, in addition to DNNs, the approach can be applied naturally to other types of deep learning models such as CNNs [22], [23], [24], [25] and RNNs [26], [27], [28], [29]. In our experiments, we will demonstrate the extension of SAT-DNN to various model and feature types.

D. Decoding of SAT-DNNs

Decoding of SAT models generally requires speaker adaptation on the testing data. For SAT-DNN models, speaker adaptation simply involves extracting the i-vector for each testing speaker and feeding the i-vector into the adaptation network. This produces the linear feature shifts specific to this speaker. Adding the feature shifts to the original feature vectors generates the speaker-adapted feature space, in which the SAT-DNN model is decoded. We summarize the major steps for training and decoding of SAT-DNN models as follows:

Training

1) Train the SI-DNN acoustic model over all the training data.
2) Re-align the training data with the SI-DNN model. The alignment *align-dnn* serves as new targets.
3) Extract the i-vector $\mathbf{i}_s$ for each training speaker.
4) Fix the parameters of the SI-DNN. Learn the adaptation network with the input features $\mathbf{o}_t$, i-vectors $\mathbf{i}_s$ and the supervision *align-dnn*.
5) Fix the parameters of the adaptation network. Update the parameters of the DNN model with the new features $\mathbf{z}_t$ and the supervision *align-dnn*.

Decoding

1) Extract the i-vector of each testing speaker.
2) Feed the i-vector to the adaptation network. Produce the speaker-specific feature shifts.
3) Decode the SAT-DNN model in the speaker-adapted feature space $\mathbf{z}_t$.

As reviewed in Section II-B, most of the existing speaker adaptation methods require multiple passes of decoding for unsupervised adaptation. The first pass of decoding generates initial hypotheses from which we derive the frame-level supervision. In comparison, because i-vector extraction uses only the speech data, adaptation of SAT-DNN models requires one single pass of decoding. Furthermore, with no fine-tuning on the adaptation data, adaptation of SAT-DNNs is insensitive to hypotheses errors. This is especially an advantage when the first-pass hypotheses have high WERs. Therefore, using SAT-DNN models, we can achieve both efficient and robust unsupervised adaptation. More empirical evidence will be presented in Section V.

IV. Experiments and Results

The proposed SAT-DNN framework is evaluated on a LVCSR task of transcribing TED talks. We present our experimental setup which is followed by results and analysis.

A. Experimental Setup

1) Dataset: Our experiments use the benchmark TEDLIUM dataset [48] which was released to advance ASR on TED talks. This publicly available dataset contains 774 TED talks that amount to 118 hours of speech data. We take this dataset as our training set and each TED talk is treated as a speaker. Decoding is done on the *dev2010* and *tst2010* test sets defined by the ASR track of the previous IWSLT evaluation campaigns. The IWSLT evaluation focuses on building end-to-end speech translation systems in which

ASR is a critical compoment. For comprehensive evaluation, we merge the two sets into a single test set which contains 19 TED talks, i.e., 4 hours of speech.

2) GMM Models: We use the open-source Kaldi toolkit [49]. The GMM systems are built with the *tedlium* recipe that has been released together with Kaldi. Adopting Kaldi's standard setup facilitates the replication of our results by other researchers. We first train the initial MLE model using 39-dimensional MFCC+$\triangle$+$\triangle\triangle$ features. Then 7 frames of MFCCs are spliced together and projected down to 40 dimensions with linear discriminant analysis (LDA). A maximum likelihood linear transform (MLLT) is estimated on the LDA features and generates the LDA+MLLT model. Discriminative training with the boosted maximum mutual information (BMMI) objective [50] is finally performed on top of the LDA+MLLT model. Our GMM model has 3980 CD triphone states and an average of 20 Gaussian components per state.

3) DNN Baseline: We build the DNN models using our PDNN framework [51]. The class label on each speech frame is generated by the GMM model through forced alignment. We use a DNN with 6 hidden layers, each of which has 1024 units and the logistic sigmoid activation function. The last softmax layer has 3980 units corresponding to the CD states. The inputs are 11 neighbouring frames (5 frames on each side of the current frame) of 40-dimensional log-scale filterbank coefficients with per-speaker mean and variance normalization. The DNN parameters are initialized with the stacked denoising autoencoders (SdAs) [52] which in our experiments have been found to give comparable performance with restricted Boltzmann machines (RBMs) [53].

For fine-tuning, we optimize the negative cross-entropy objective using the exponentially decaying "newbob" learning rate schedule. Specifically, the learning rate starts from a big value 0.08 and remains unchanged until the increase of the frame accuracy on a cross-validation set between two consecutive epochs falls below 0.2%. Then the learning rate is decayed by a factor of 0.5 at each of the subsequent epochs. The whole learning process terminates when the frame accuracy fails to improve by 0.2% between two successive epochs. We use the mini-batch size of 256 for SGD, and a momentum of 0.5 for gradient smoothing.

4) SAT-DNN: The DNN model which has been trained serves as the SI-DNN for constructing the SAT-DNN model. The adaptation network contains 3 hidden layers each of which has 512 units and uses the logistic sigmoid activation function. The output layer has 440 (the dimension of the filterbank features) units and uses the identity activation function. Parameters of the adaptation network are randomly initialized. Training of the adaptation network and updating of the DNN follow the same fine-tuning setting as training of the SI-DNN.

I-vector extraction is conducted with Kaldi's in-built i-vector functionality. The i-vector extractor takes 19-dimensional MFCCs and log-energy as the features. Computing deltas and accelerations finally gives a 60-dimensional feature vector on each frame. Both the UBM model and the total variability matrices are trained on the entire training set. For each training and testing speaker, we extract a 100-dimensional i-vector which is observed to give the optimal recognition performance in our previous studies [31], [32].

B. Basic Results

During decoding, the original Kaldi *tedlium* recipe relies on a generic CMU language model. In our setup, we train a lightweight trigram language model using 180k sentences of TED talk transcripts. This in-domain language model is pruned aggressively for decoding efficiency. Table I compares the results of the DNN and SAT-DNN models on the test set. The SI-DNN model gets a WER of 20.0% which is notably better than the discriminatively trained GMM model. Compared to the results obtained by the Kaldi *tedlium* recipe, our SI-DNN model achieves lower WERs, meaning that we are working with a reasonable baseline. When training the SAT-DNN model, we can employ the frame-level alignment generated from the GMM model, i.e., the same alignment as used by the SI-DNN training. In this case, Table I shows that the SAT-DNN has a WER of 18.1%, that is 9.5% relative improvement over the SI-DNN. Alternatively, we can re-align the training data with the SI-DNN and use the newly-generated alignment during SAT-DNN training (both learning of the adaptation network and updating of the DNN). This re-alignment step improves the WER of the SAT-DNN further to 17.7%. Also, we observe that SAT-DNN training with the new DNN-generated alignment converges faster than with the old GMM-generated alignment. Thus, unless otherwise stated, training of the SAT-DNN models always uses the new alignment in the rest of our experiments.

TABLE I
PERFORMANCE COMPARISONS BETWEEN THE SI-DNN BASELINE AND SAT-DNN MODELS IN TERMS OF WERS(%) ON THE TEST SET. THE LAST COLUMN SHOWS THE RELATIVE IMPROVEMENT OF SAT-DNNS OVER THE SI-DNN.

Model	Alignment	WER(%)	Rel. Imp. (%)
BMMI-GMM	—	24.1	—
SI-DNN	from GMM	20.0	—
SAT-DNN	from GMM	18.1	9.5
SAT-DNN	from DNN	17.7	11.5

C. Bridging I-vector Extraction with DNN Training

I-vector extraction in Section II-C aims to optimize the objective of speaker modeling. The UBM model and the total variability matrices are trained with MLE. In contrast, DNN and SAT-DNN models try to distinguish phonetic classes and are trained in a discriminative manner. The separate training of these two parts with respect to different objectives may hurt the performance of the SAT-DNN models. Past work [54] has studied approaches to leveraging DNN models in i-vector extraction. In [54], the UBM used in i-vector extraction is replaced with a DNN that has been trained for acoustic modeling. In this case, the probabilities $r_t(k)$ are posteriors of states outputted by the DNN, instead of posteriors of the Gaussians from the UBM. This manner of incorporating DNNs into i-vector extraction results in significant improvement on a benchmark speaker recognition task [54].

In this paper, we bridge i-vector extraction with DNN training from the front-end perspective. Prior to building the i-vector extractor, we learn a DNN model that is designed for bottleneck-feature (BNF) generation. Instead of MFCCs, outputs from the bottleneck layer of such a BNF-DNN are taken as the features for training the i-vector extractor (both the UBM and total variability matrices). Only changing the front-end enables us to take advantage of the existing i-vector extraction pipeline, without making major modifications. The incorporation of the DNN-based features adds phonetic discrimination ability to the extracted i-vectors, which potentially benefits the SAT-DNN training.

Following our Deep BNF (DBNF) architecture [55], [56], the BNF-DNN has 6 hidden layers in which the 5th layer is a bottleneck layer. To be consistent with MFCCs, the bottleneck layer has 60 nodes whereas each of the other hidden layers has 1024 nodes. Inputs to the BNF-DNN are 11 neighbouring frames of 40-dimensional filterbank features. In Table II, we show the results of SAT-DNNs using i-vectors extracted from MFCCs and BNFs respectively. Within the SAT-DNN framework, BNF-based i-vectors result in slight improvement (0.4% absolute) over MFCC-based i-vectors. Applying the BNF-based i-vectors brings the WER of the SAT-DNN model down to 17.3% that is 13.5% relative improvement compared to the SI-DNN baseline.

TABLE II
WER(%) OF SAT-DNN MODELS WITH I-VECTORS FROM MFCC AND BNF FEATURES RESPECTIVELY. THE LAST COLUMN SHOWS THE RELATIVE IMPROVEMENT OF SAT-DNNS OVER THE SI-DNN BASELINE FROM TABLE I.

Model	Features for I-vectors	WER(%)	Rel. Imp. (%)
SAT-DNN	MFCCs	17.7	11.5
SAT-DNN	BNFs	17.3	13.5

D. Variable Data Amount for I-vector Extraction

In the testing set, each speaker on average has 780 seconds of speech data. During decoding, the previous experiments have used all the data to estimate i-vectors for adaptation. In certain scenarios (e.g., online decoding), however, the adaptation data may become highly limited. This subsection examines how the amount of adaptation data affects the recognition performance of SAT-DNNs. For convenience of formulation, we use the variable β to denote the average amount of adaptation data per speaker in terms of seconds. By tuning the maximum number of frames one speaker can have, we systematically reduce β from 780 to 12, by a factor of 2 each time. Note that we only vary the data amount for adaptation (i.e., i-vector estimation), keeping the data to be eventually decoded unchanged. Fig. 5 depicts the variation of WERs with different values of β. Reducing β up to 25 leads to either zero or minor WER degradation. We start to observe notable degradation when β equals 12. This suggests that SAT-DNN performs robustly to reduced amount of adaptation data, and thus is applicable to online decoding.

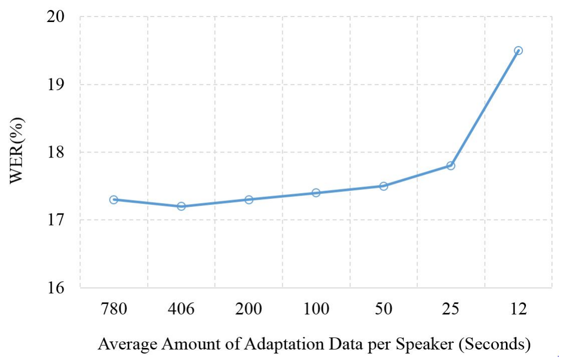


Fig. 5. The WERs of SAT-DNN models when β is reduced from 780 to 12.

E. Application to fMLLR Features

So far, we have looked at the non-adaptive filterbank features as the DNN inputs. This subsection examines how the SAT-DNN model works when the DNN inputs are speaker-adapted fMLLR features. We perform SAT over the LDA+MLLT GMM model, which produces the SAT-GMM model and fMLLR transforms of the training speakers. DNN inputs include 11 neighbouring frames of 40-dimensional fMLLR features. Training of the DNN with fMLLRs follows the same protocol as training of the DNN with filterbanks, using alignment generated by the SAT-GMM model. During SAT-DNN training, the frame-level class labels come from re-alignment with the new fMLLR-based DNN model. Table III shows the performance of the resulting SAT-DNN models. Compared to the DNN model, the SAT-DNN obtains 7.9% relative improvement (17.4% vs 18.9%) in terms of WER. Because speaker variability has been partly modeled by fMLLR transforms, the gains achieved by SAT-DNN here are less significant than the gains on filterbank features.

TABLE III
PERFORMANCE COMPARISONS BETWEEN DNN AND SAT-DNN MODELS WHEN THE INPUTS ARE FMLLR FEATURES. THE LAST COLUMN SHOWS THE RELATIVE IMPROVEMENT OF THE SAT-DNN OVER THE DNN BASELINE.

Model	WER(%)	Rel. Imp. (%)
DNN (baseline)	18.9	—–
SAT-DNN	17.4	7.9

F. Extension to BNFs

In Section III, we argue that our SAT approach is a general framework. To demonstrate this, we empirically study two extensions of SAT-DNNs. The first extension is to apply the SAT technique to BNF generation. In addition to CD-state classifiers, DNNs can also be used as feature extractors. A widely employed manner is to place a bottleneck layer in the DNN architecture. Outputs from the bottleneck layer are taken as new features on top of which GMM-based *tandem systems* are further built. As with Section IV-C, we turn to the DBNF method [55] for bottleneck feature extraction. DBNF is characterized by the asymmetric arrangement of the layers and multiple hidden layers before the bottleneck layer. Following our previous work [55], [56], the DBNF network used in this paper has 6 hidden layers in which the 5th hidden layer is a bottleneck layer. This bottleneck layer has 42 units, whereas each of the other hidden layers has 1024 nodes. The parameters of the 4 prior-to-bottleneck layers are initialized with SdA pre-training [52]. Inputs to the DBNF network are 11 frames of filterbank features. After this network is trained, we build a LDA+MLLT GMM model over the extracted BNF features. Specifically, at each frame, the 42-dimensional BNF features are further normalized with mean subtraction. Then, 7 consecutive BNF frames are spliced and then projected to 40 dimensions using LDA. On top of the LDA+MLLT model, 4 iterations of discriminative training is performed with the BMMI objective [50].

Applying SAT to BNF extraction is straightforward. We simply replace the DNN model in Fig. 2 with the DBNF network. When training is finished, we obtain the features $\mathbf{z}_t$ with the adaptation network and i-vectors. Speaker-adapted BNF features are generated by feeding $\mathbf{z}_t$ to the SAT-DBNF network. Table IV compares the performance of the final BMMI-GMM models when the bottleneck features are extracted from the DBNF and SAT-DBNF networks respectively. We observe that the GMM model with BNF features largely outperforms (17.7% vs 24.1%) the GMM model with MFCCs. Extracting features from the SAT-DBNF further reduces the WER to 16.2%, i.e., 32.8% and 8.5% relative improvement over the MFCC and the standard DBNF front-end respectively. This demonstrates the effectiveness of the SAT technique in improving the quality of bottleneck features.

TABLE IV
PERFORMANCE OF BMMI-GMM MODELS WHEN THE FEATURES ARE MFCCS, DBNF AND SAT-DBNF. THE LAST COLUMN SHOWS THE RELATIVE IMPROVEMENT OF THE SAT-DBNF OVER THE STANDARD DBNF.

Front-end	WER(%)	Rel. Imp. (%)
MFCCs	24.1	—
DBNF	17.7	—
SAT-DBNF	16.2	8.5

G. Extension to CNNs

The second extension is to port the SAT technique to CNN acoustic models. CNNs have been exploited as an alternative to DNNs for acoustic modeling [22], [23], [24], [25]. Instead of fully-connected weight matrices, CNNs use local filters in order to capture locality along the frequency bands. On top of the convolution layers, max-pooling layers are usually added to improve the shift invariance in the frequency domain. Our CNN architecture follows [23], consisting of 2 convolution and 4 fully-connected layers. We apply 2-dimensional convolution over both time and frequency. The CNN inputs are filterbanks+$\triangle$+$\triangle\triangle$, with a temporal context of 11 frames. The first convolution layer filters the inputs using 256×3 kernels each of which has the size of 9×9. This is followed by a max-pooling layer only along the frequency axis, with the pooling size of 3. The second convolution layer takes as inputs the outputs from the pooling layer and filters them with 256×256 kernels with the size of 4×3. All the convolution operations are overlapping with the stride of 1 whereas the pooling is always non-overlapping. Outputs from the convolution layers are 256 feature maps, each of which has the size of 8×1. We then place 4 fully-connected hidden layers and finally the softmax layer atop of the convolution layers. Both the convolution and the fully-connected layers use the logistic sigmoid activation function. Detailed configurations of the convolution layers are listed in Table V.

TABLE V
CONFIGURATIONS (FILTER AND POOLING SIZE) OF THE TWO CONVOLUTION LAYERS IN OUR CNN ARCHITECTURE.

	#1 conv layer		#2 conv layer	
	frequency	time	frequency	time
Filter	9	9	4	3
Pooling	3	1	1	1

As with SAT-DNNs, training of the SAT-CNN model starts from a well-trained SI-CNN model. The i-vectors are extracted with BNF features as described in Section IV-C. We learn the adaptation network that has the output dimension of $3\times11\times40$. The feature shifts are added to the original features and the CNN model is updated in the new feature space. Table VI presents the performance of the SI-CNN and SAT-CNN models. We can see that our SI-CNN model outperforms both the SI-DNN model with filterbanks and the DNN model with fMLLRs, revealing that the SI-CNN poses a strong baseline. In comparison with the SI-CNN, the SAT-CNN model gives a better WER. However, the 7.2% relative improvement (16.8% vs 18.1%) achieved by SAT-CNN over SI-CNN is less than the improvement achieved by SAT-DNN over SI-DNN. This is partly because CNNs normalize spectral variation more effectively than DNNs, which decreases the efficacy of the application of SAT.

TABLE VI
PERFORMANCE COMPARISONS BETWEEN SI-CNN AND SAT-CNN MODELS. THE LAST COLUMN SHOWS THE RELATIVE IMPROVEMENT OF THE SAT-CNN MODEL OVER THE SI-CNN BASELINE.

Model	WER(%)	Rel. Imp. (%)
SI-CNN (baseline)	18.1	—
SAT-CNN	16.8	7.2

V. SAT AND SPEAKER ADAPTATION

In this section, we focus on the comparison and combination of SAT and speaker adaptation for DNN models. Our SAT approach is firstly compared against two competitive speaker adaptation methods. Then, we show that model-space adaptation can further improve the recognition accuracy of SAT-DNNs.

A. Comparing SAT and Speaker Adaptation

Past work [13] closely related with this paper performs speaker adaptation of DNNs by incorporating i-vectors. Specifically, at each frame t, the original DNN input vector $\mathbf{o}_t$ is concatenated with the corresponding speaker i-vector $\mathbf{i}_s$. The combined feature vector $[\mathbf{o}_t, \mathbf{i}_s]$ is treated as the new DNN inputs, in both training and testing. For convenience of formulation, this method is referred to as *DNN+I-vector*.

A more recent method for model-space adaptation is LHUC proposed in [33]. In this method, the SI-DNN model is adapted to a specific speaker s with a parameter vector $\mathbf{c}_s^i$ at each hidden layer. This SD vector acts to regulate the hidden activations of the SI-DNN model. The outputs of the i-th hidden layer now become:

$$\mathbf{y}_t^i = \psi(\mathbf{c}_s^i) \odot \sigma(\mathbf{W}_i \mathbf{x}_t^i + \mathbf{b}_i) \tag{13}$$

where $\odot$ represents element-wise multiplication, ψ is a function to constrain the range of the parameter vector and is set to $\psi(x) = 2/(1 + exp(-x))$ in [33]. During adaptation, only the SD parameters $[\mathbf{c}_s^i, 1 \leq i < L]$ are estimated on the adaptation data with error back-propagation, whereas the SI-DNN parameters remain unchanged.

Our implementation of these two methods follows [13] and [33]. For DNN+I-vector, a 100-dimensional i-vector is extracted for each training or testing speaker. All the other settings are consistent with the settings of the SI-DNN model in Section IV-A. For LHUC, the elements of $\mathbf{c}_s^i$ are initialized uniformly to 0. A first pass of decoding with the SI-DNN is done to get the frame-level supervision. Training of the SD parameters goes through 3 epochs with the constant learning rate of 0.8. Table VII shows the WERs of the DNN models adapted with these two methods. We observe that compared to the (unadapted) SI-DNN, DNN+I-vector reduces the WER by 2.5% relative, a much smaller improvement than reported in [13]. These diminished gains are partly because DNN+I-vector in our setup uses the raw i-vectors without any normalization [38] or dimension truncation [17]. Compared to the same SI-DNN baseline, DNN+LHUC brings 8.5% relative improvement. However, both methods are outperformed by the SAT-DNN model, which justifies the necessity of conducting complete SAT for DNN models. Also, the performance gap between DNN+I-vector and SAT-DNN suggests that our SAT-DNN framework removes the need for hand-crafted post-processing steps over i-vectors. The adaptation network in SAT-DNN naturally transforms the raw i-vectors with respect to the objective function related to acoustic modeling.

TABLE VII
PERFORMANCE COMPARISONS BETWEEN SAT-DNN AND SPEAKER ADAPTATION METHODS. THE LAST COLUMN SHOWS THE RELATIVE IMPROVEMENT OVER THE SI-DNN BASELINE.

Model/Adaptation Methods	WER(%)	Rel. Imp. (%)
SI-DNN (baseline)	20.0	—
SAT-DNN	17.3	13.5
DNN+I-vector[13]	19.5	2.5
DNN+LHUC[33]	18.3	8.5

For more complete comparisons, we experiment with LHUC when different sets of supervision hypotheses are used. Specifically, we perform decoding with the SI-GMM (LDA+MLLT), SAT-GMM and SI-DNN models respectively. As shown in Table VIII, these models have the WERs of 28.1%, 23.3% and 20.0%, simulating first-pass hypotheses of different quality levels. The resulting hypotheses are employed as supervision for LHUC-based adaptation. We observe that the performance of the adapted DNNs gets worse as the WER of the supervision hypotheses deteriorates. This degradation results from fine-tuning on the adaptation data where the erroneous supervision is taken as true class labels. On the contrary, the adaptation of the SAT-DNN model requires no fine-tuning on the adaptation data and therefore its performance is robust to supervision errors. SAT-DNN is more suitable for unsupervised adaptation, especially when the first-pass decoding has high WERs.

TABLE VIII
PERFORMANCE OF ADAPTED DNN MODELS USING THE LHUC ADAPTATION METHOD. THE SECOND COLUMN LISTS THE WERS OF DIFFERENT SETS OF SUPERVISION HYPOTHESES. THE THIRD COLUMN SHOWS THE WERS OF THE CORRESPONDING ADAPTED DNN MODELS.

1stPass System	WER(%) of Hypotheses	WER(%) of Adapted DNN
SI-GMM	28.1	19.3
SAT-GMM	23.3	18.4
SI-DNN	20.0	18.3

B. *Combining SAT and Model-space Adaptation*

The i-vector based adaptation of SAT-DNN is in fact feature-space adaptation because it normalizes the DNN inputs. After getting the speaker-normalized features ($\mathbf{z}_t$ in Equation (10)), the SAT-DNN model can be further adapted in this new feature space, using any of the existing model-space adaptation methods. We turn to LHUC for model-space adaptation. Table IX provides a summary of the complete results of the SI-DNN and SAT-DNN models when the input features are filterbanks. In this table, "+LHUC" means that the LHUC-based adaptation is applied over the SAT-DNN. We can see that applying LHUC on top of SAT-DNN produces nice gains in terms of WERs. Combining SAT-DNN and LHUC together finally gives us a WER of 16.5% on the test set, which is 17.5% relative improvement compared to the SI-DNN model.

TABLE IX
A SUMMARY OF THE PERFORMANCE OF SAT-DNNS USING I-VECTORS EXTRACTED FROM MFCCS AND BNFS RESPECTIVELY. "+LHUC" MEANS THAT THE LHUC-BASED ADAPTATION IS APPLIED OVER SAT-DNNS. THE LAST COLUMN SHOWS THE RELATIVE IMPROVEMENT OVER THE SI-DNN.

Model	WER(%)	Rel. Imp. (%)
SI-DNN	20.0	—
I-vectors from MFCCs		
SAT-DNN	17.7	11.5
+LHUC	16.7	16.5
I-vectors from BNFs		
SAT-DNN	17.3	13.5
+LHUC	16.5	17.5

VI. ACCELERATION OF SAT-DNN TRAINING

A major concern for building SAT-DNN models is the additional cost it imposes to the training pipeline. In this section, we investigate a simple but effective data reduction strategy to speed up the training of SAT-DNNs. The intuition is that the adaptation network models mapping from the i-vectors to the feature shifts. As a result, during training of the adaptation network, the number of distinct i-vectors plays a more important role than the number of speech frames. Moreover, because of taking the SI-DNN model for initialization, updating of the DNN supposedly needs less data compared with training from scratch. Based on these intuitions, we select one example out of every n speech frames for training of SAT-DNNs. That is, after picking a training example, we skip the following $n-1$ frames to pick the next one. By doing this, we keep the number of training speakers (i-vectors) unchanged. However, the training set is shrunk to $1/n$ of its original size, which speeds up the SAT-DNN training by n times.

In Table X, we present the results of the resulting SAT-DNN models when n ranges from 1 to 4. The data size is measured by the ratio of the reduced size to the original size. No LHUC-based adaptation is performed in order to leave out the impact of other factors. With n = 2, we accelerate the training process by 2× while getting 1.7% relative WER degradation (17.6% vs 17.3%). A direct comparison is to shrink the training set by reducing the number of speakers, without skipping any speech frames. As shown in Table X, the results of SAT-DNNs using this speaker reduction strategy become notably worse, e.g., 7.0% relative degradation (18.5% vs 17.3%) when n is set to 2. This reveals that data reduction based on frame skipping is an effective strategy to speed up SAT-DNN training without causing significant loss on WERs.

TABLE X
PERFORMANCE OF SAT-DNN MODELS WHEN THE TRAINING SET IS SHRUNK BY FRAME SKIPPING ("FRAME") AND SPEAKER REDUCTION ("SPEAKER") RESPECTIVELY. THE DATA SIZE IS MEASURED BY THE RATIO OF THE REDUCED SIZE TO THE ORIGINAL SIZE.

Data size	Speed up	WER(%) Frame	WER(%) Speaker
1	1×	17.3	
1/2	2×	17.6	18.5
1/3	3×	18.1	18.9
1/4	4×	18.3	19.0

VII. CONCLUSIONS

In this paper, we have proposed a general framework to perform SAT for DNN acoustic models. The SAT approach relies on i-vectors and the adaptation neural network to realize feature normalization. This work explores the application of SAT-DNNs to LVCSR tasks. First, we determine the optimal configurations (e.g., features for i-vector extraction) for building of SAT-DNN models. The SAT approach is proved to be a general method in that it can be extended easily to different feature and model types. Second, we study the comparison and combination of SAT and speaker adaptation in the context of DNNs. Third, a data reduction strategy, frame skipping, is employed to accelerate the training process of SAT-DNNs. Our experiments have been conducted on a LVCSR task of transcribing TED talks. Compared to the DNN baseline, our SAT-DNN model achieves 13.5% relative improvement in terms of WERs. This improvement is enlarged to 17.5% when the LHUC-based model adaptation is further applied on top of SAT-DNNs. We also observe that the frame skipping strategy results in nice speed up for SAT-DNN training without resulting in significant WER degradation on the testing data.

ACKNOWLEDGMENT

The authors would like to thank Lu Jiang and Justin Chiu from Carnegie Mellon University for helpful discussions. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

REFERENCES

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

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

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

[4] N. Jaitly, P. Nguyen, A. W. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Thirteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2012.

[5] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide, "Feature learning in deep neural networks-studies on speech recognition tasks," *arXiv preprint arXiv:1301.3605*, 2013.

[6] C. J. 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.

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

[8] H. Liao, E. McDermott, and A. Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 368–373.

[9] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7893–7897.

[10] B. Li and K. C. Sim, "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems," in *Eleventh Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2010.

[11] 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 *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012.

[12] S. P. Rath, D. Povey, K. Veselỳ, and J. Cernockỳ, "Improved feature processing for deep neural networks," in *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2013, pp. 109–113.

[13] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 55–59.

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

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

[16] J. Trmal, J. Zelinka, and L. Müller, "On speaker adaptive training of artificial neural networks," in *Eleventh Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2010.

[17] A. Senior and I. Lopez-Moreno, "Improving DNN speaker independence with i-vector inputs," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 225–229.

[18] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1713–1725, 2014.

[19] T. Ochiai, S. Matsuda, X. Lu, C. Hori, and S. Katagiri, "Speaker adaptive training using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6349–6353.

[20] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Tenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2009, pp. 1559–1562.

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

[22] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8614–8618.

[23] H. Soltau, G. Saon, and T. N. Sainath, "Joint training of convolutional and non-convolutional neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5572–5576.

[24] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 10, pp. 1533–1545, 2014.

[25] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A.-r. Mohamed, G. Dahl, and B. Ramabhadran, "Deep convolutional neural networks for large-scale speech tasks," *Neural Networks*, 2014.

[26] A. L. Maas, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust ASR," in *Thirteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2012.

[27] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6645–6649.

[28] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[29] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[30] Y. Miao and F. Metze, "On speaker adaptation of long short-term memory recurrent neural networks," in *Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH) (To Appear)*. ISCA, 2015.

[31] Y. Miao, H. Zhang, and F. Metze, "Towards speaker adaptive training of deep neural network acoustic models," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[32] Y. Miao, L. Jiang, H. Zhang, and F. Metze, "Improvements to speaker adaptive training of deep neural networks," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014.

[33] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014.

[34] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, vol. 15, 2011, pp. 315–323.

[35] Y. Miao, F. Metze, and S. Rawat, "Deep maxout networks for low-resource speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 398–403.

[36] Y. Miao and F. Metze, "Improving language-universal feature extraction with deep maxout and convolutional neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[37] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6359–6363.

[38] V. Gupta, P. Kenny, P. Ouellet, and T. Stafylakis, "I-vector-based speaker adaptation of deep neural networks for French broadcast audio transcription," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6334–6338.

[39] P. Karanasou, Y. Wang, M. J. Gales, and P. C. Woodland, "Adaptation of deep neural network acoustic models using factorised i-vectors," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[40] J. Li, J.-T. Huang, and Y. Gong, "Factorized adaptation for deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5537–5541.

[41] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7942–7946.

[42] ——, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition," in *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2013, pp. 1248–1252.

[43] R. Price, I. Kenichi, and K. Shinoda, "Speaker adaptation of deep neural networks using a hierarchy of output layers," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014.

[44] Z. Huang, J. Li, S. M. Siniscalchi, I.-F. Chen, C. Weng, and C.-H. Lee, "Feature space maximum a posteriori linear regression for adaptation of deep neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

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

[46] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A study of interspeaker variability in speaker verification," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 5, pp. 980–988, 2008.

[47] M. Karafiát, L. Burget, P. Matejka, O. Glembek, and J. Cernocky, "iVector-based discriminative adaptation for automatic speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 152–157.

[48] A. Rousseau, P. Deléglise, and Y. Estève, "Ted-lium: an automatic speech recognition dedicated corpus." in *LREC*, 2012, pp. 125–129.

[49] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The Kaldi speech recognition toolkit," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 1–4.

[50] D. Povey, "Discriminative training for large vocabulary speech recognition," Ph.D. dissertation, University of Cambridge, 2005.

[51] Y. Miao, "Kaldi+PDNN: building DNN-based ASR systems with Kaldi and PDNN," *arXiv preprint arXiv:1401.6984*, 2014.

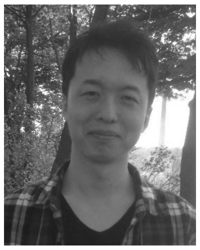
[52] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[53] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 599–619.

[54] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1695–1699.

[55] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3377–3381.

[56] J. Gehring, W. Lee, K. Kilgour, I. Lane, Y. Miao, and A. Waibel, "Modular combination of deep neural networks for acoustic modeling," in *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2013.



Yajie Miao received his B.E. degree from Nanjing University of Posts and Telecommunications, Nanjing, China, and M.E. degree from Tsinghua University, Beijing, China. He is now a Ph.D. candidate in School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

Hao Zhang received his B.E. degree from Tsinghua University, Beijing, China. He is now a master student in School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

Florian Metze (S99-M06) received the Diploma in Physics from Ludwig-Maximilians-Universitt Mnchen (LMU) in 1998, and the Ph.D. degree in Computer Science from Universitt Karlsruhe (TH) in 2005, for his dissertation on “Articulatory Features for Conversational Speech Recognition”. From 2006 to 2009, he was with Deutsche Telekom Laboratories (T-Labs) at Technische Universitt Berlin. He joined Carnegie Mellon Universitys Language Technologies Institute (LTI) in 2009, where he is currently an Associate Research Professor. He is also adjunct faculty in the Human Computer Interaction Institute (HCII). He has published over 150 peer-reviewed papers and holds several patents. His research interests are in the area of automatic speech recognition, multimedia and sound analysis, as well as human computer interaction.