EXTENDED LOW-RANK PLUS DIAGONAL ADAPTATION FOR DEEP AND RECURRENT NEURAL NETWORKS

Yong Zhao, Jinyu Li, Kshitiz Kumar, and Yifan Gong

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA

{yonzhao; jinyli; kskumar; ygong}@microsoft.com

ABSTRACT

Recently, the low-rank plus diagonal (LRPD) adaptation was proposed for speaker adaptation of deep neural network (DNN) models. The LRPD restructures the adaptation matrix as a superposition of a diagonal matrix and a product of two low-rank matrices. In this paper, we extend the LRPD adaptation into the subspace-based approach to further reduce the speaker-dependent (SD) footprint. We apply the extended LRPD (eLRPD) adaptation for the DNN and LSTM models with emphasis placed on the applicability of the adaptation to large-scale speech recognition systems. To speed up the adaptation in test time, we propose the bottleneck (BN) caching approach to eliminate the redundant computations during multiple sweeps of development data. Experimental results on the short message dictation (SMD) task show that the eLRPD adaptation can reduce the SD footprints by 82% for the SVD DNN and 96% for the LSTM-RNN over the linear adaptation, while maintaining the comparable accuracy. The BN caching achieves up to 3.5 times speedup in adaptation at no loss of recognition accuracy.

Index Terms: deep neural network, recurrent neural network, long short-term memory, speaker adaptation

1. INTRODUCTION

The application of deep neural networks (DNNs) has achieved tremendous success for large vocabulary continuous speech recognition (LVCSR) [1, 2, 3, 4]. As an alternative to DNNs, recurrent neural networks (RNNs) have also been studied extensively for the task of acoustic modeling [5]. RNNs are characterized by recurrent connections on the hidden layers which allow modelling of long-range temporal context for improved sequence labelling. More recently, long short-term memory (LSTM) RNNs have been shown to outperform DNNs in speech recognition [6, 7, 8].

Despite their outstanding performance, both DNNs and LSTMs may still suffer from the accuracy degradation due to the potential acoustic mismatch between the training and test conditions. To mitigate this mismatch, various methods for speaker adaptation of neural networks (NNs) have been proposed. These methods can be classified into three categories [9]. First, the speaker-independent (SI) model, or certain layers of the model, is directly updated [10, 11, 12]. To avoid overfitting, conservative training such as Kullback-Leibler divergence (KLD) regularization [11] is proposed.

The second category of approaches inserts speaker-dependent (SD) linear transformation layer on top of certain layers in the generic model. The layer being adapted can be either input features, hidden layers, or the input to the softmax layer [13, 14, 15, 16, 17, 18]. The transformation-based methods may still suffer from overfitting, if the dimension of the adapted layer is high and the transformation matrix is in a full-rank form. In [19, 20], the original large full-size DNN model is converted to a much smaller low-rank DNN model by using singular value decomposition (SVD). Then, SVD bottleneck adaptation is done by applying the linear transform to the bottleneck layer. Thus, only matrices of much lower dimension are updated for each speaker. Many low-footprint adaptation techniques have been proposed to constrain the transforms to be structured, such as block-diagonal [15], diagonal [17, 21, 22], and bias only [23]. In [24], we proposed the low-rank plus diagonal (LRPD) adaptation, which restructures the transformation matrix as a superposition of a diagonal matrix and a product of two low-rank matrices. By varying the rank of the transformation, the LRPD contains the full and the diagonal transformation matrices as its special cases.

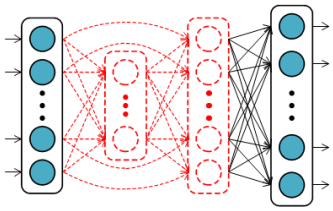
In the third category, the subspace methods aim to find a low dimensional subspace of the transformations, so that each transformation can be specified by a small number of parameters. One popular method in this category is the use of auxiliary features, such as i-vector [25, 26] and speaker code [23], which are concatenated with the standard acoustic features. It can be shown that the augmentation of auxiliary features is equivalent to confining the adapted bias vectors into a speaker subspace [9]. Other subspace methods include tensor-based adaptation [27], cluster adaptive training (CAT) [28, 29], factorized hidden layer (FHL) [30, 31], where the transformations are confined into the speaker subspace. The subspace methods introduce the connection layers to link the speaker representation parameters with the generic SI model. The connection layers can be linear, nonlinear, or a stack of multiple layers [32]. All the parameters (SI model, connection layers, and speaker representations) can be adaptively learned from the training data along with speaker labels.

Despite extensive research made, there remain outstanding challenges to real-world deployment of the adaptation algorithms. This paper presents out recent progress in speaker adaptation towards the cloud-based speech recognition systems. We first extend the LRPD adaptation into the subspace-based approach to further reduce the SD footprint, inspired by the FHL approach [30]. The extended LRPD (eLRPD) is learned in a lightly adaptive training manner by keeping the original SI model fixed. We then apply the eLRPD adaptation for the DNN and LSTM models and examine how various setups affect the recognition accuracy given a very powerful SI model. To speed up the adaptation in test time, we propose the bottleneck (BN) caching approach to eliminate the redundant computations on the model portion below the layer being adapted during multiple sweeps of development data. The proposed methods are evaluated on a short message dictation (SMD) task.

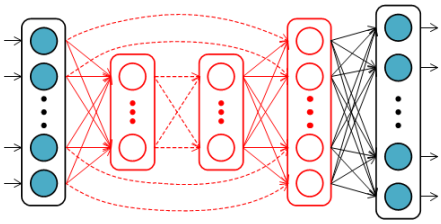
2. LOW-RANK PLUS DIAGONAL (LRPD) ADAPTATION

One popular approach for adapting DNNs is applying a linear transformation to a certain layer to account for the mismatch between the training and test conditions. One main issue with these adaptation techniques is that they have a large number of SD parameters per speaker due to the high dimensionality of the DNN layers.

We have recently presented the LRPD adaptation for DNNs in [24] to flexibly control the number of adaptation parameters according to the available adaptation data while maintaining the recognition accuracy. The algorithm is motivated by observing that the speaker-specific transformation matrix $\boldsymbol{W}_s$ is very close to an identity ma-



(a) LRPD adaptation



(b) Extended LRPD adaptation

Fig. 1: Illustration of the network structures of the LRPD and extended LRPD adaptation methods. Shaded nodes denote nonlinear units, unshaded nodes for linear units. Red dashed links correspond to the SD parameters.

trix, because the adapted model should not deviate too far from the SI model given the limited number of adaptation data.

Given a $k \times k$ adaptation matrix $\boldsymbol{W}_s$, we approximate it as a superposition of a diagonal matrix $\boldsymbol{D}_s$ and a product of two smaller low-rank matrices $\boldsymbol{P}_s$ and $\boldsymbol{Q}_s$, respectively

$$\boldsymbol{W}_{s,k\times k} \approx \boldsymbol{D}_{s,k\times k} + \boldsymbol{P}_{s,k\times c}\boldsymbol{Q}_{s,c\times k}. \quad (1)$$

The number of elements in the LRPD decomposition is $k(2c+1)$, whereas the original $\boldsymbol{W}_s$ has k^2 elements. If $c \ll k$, this can significantly reduce the adaptation model footprint. We can see that the LRPD adaptation amounts to inserting two linear layers above the layer being adapted, as shown in Fig. 1a.

The LRPD contains the full and the diagonal adaptation matrices as its special cases. When $c = 0$, the LRPD is reduced to adaptation with diagonal matrix. Specifically, if we apply the diagonal transforms before or after all non-linear layers, we may achieve the sigmoid [22] or learning hidden unit contribution (LHUC) [21] adaptation. Moreover, the LRPD adaptation can not only be applied to adapt a full-size DNN, but also be applied to the bottleneck layer of a SVD DNN, leading to a combination of the SVD bottleneck adaptation [20] and LRPD adaptation.

2.1. Extended LRPD (eLRPD) adaptation

One issue with the LRPD adaptation is that the number of free parameters is proportional to the dimension of the layer being adapted. When the layer dimension is high, we have to greatly reduce the rank c to control the size of the SD footprint. In [24], we observed that applying the LRPD above a small-sized bottleneck layer of the SVD DNN performs better than above a layer of the full-sized DNN.

To mitigate this problem, we extend the LRPD adaptation into the subspace-based approach, inspired by the factorized hidden layer (FHL) approach for adapting the DNN models [30]. Specifically, we insert another full matrix $\boldsymbol{T}_s$ with the size of $c \times c$ between the two low-rank matrices as follows:

$$\boldsymbol{W}_{s,k\times k} \approx \boldsymbol{D}_{s,k\times k} + \boldsymbol{P}_{k\times c}\boldsymbol{T}_{s,c\times c}\boldsymbol{Q}_{c\times k} \quad (2)$$

where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are redefined as matrices independent of speakers and dedicated to connect the SD parameters $\boldsymbol{T}_s$ with the generic SI model. It can be shown that by (2), the transformation matrices with $\boldsymbol{D}_s$ deducted are constrained to lie in a subspace spanned by a set of rank-one matrices.

The number of SD elements in the eLRPD is $c^2 + k$, compared with $k(2c+1)$ in the conventional LRPD. This makes the SD footprint under less influence from the dimensionality of the layer being adapted. The eLRPD is equivalent to inserting three linear layers above the layer being adapted, as shown in Fig. 1b.

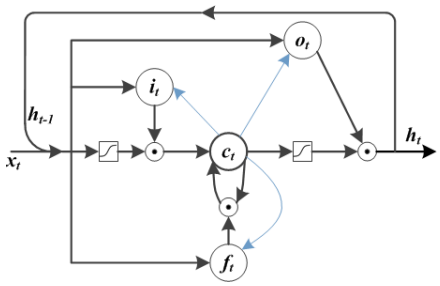


Fig. 2: A memory block of LSTM.

2.2. Lightly adaptive training

We need to adaptively learn the connection weight matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ from the training data, and $\boldsymbol{D}_s$ and $\boldsymbol{T}_s$ for both the training and test speakers. During adaptive training, the generic SI model combined with $\boldsymbol{P}$ and $\boldsymbol{Q}$ forms the canonical model. Typically, the update of the canonical model and the speaker representations are interleaved over all training data.

However, from the practice perspective, this training scheme has problems. First, the generic SI model after being updated, when using alone, is not guaranteed to produce the best recognition performance against the test utterances. Thus, it may require that the original SI model be also kept for the first pass decoding. Second, during adaptive training, the algorithm has to frequently switch between SD parameters according to the speaker identities of the training utterances. This poses a great challenge to the underlying deep learning algorithm, which is implemented based on the massive parallelization power of the Graphical processing units (GPUs).

To address this issue, we adopt a lightly adaptive training approach to only adaptively learn $\boldsymbol{P}$ and $\boldsymbol{Q}$, while keeping the original SI model fixed. Furthermore, since $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{D}_s$, and $\boldsymbol{T}_s$ are small in size, we select a small portion of the training data for adaptive training. In test time, we use the SI model to generate the hypothesis for unsupervised adaptation. The SD parameters are then estimated.

3. ADAPTATION OF LSTM-RNN MODELS

Fig. 2 depicts the conventional LSTM that has been most commonly used in the literature. An LSTM layer is composed of multiple memory blocks. Each memory block consists of self-connected memory cells $\boldsymbol{c}_t$ and three multiplicative gate units (input $\boldsymbol{i}_t$, output $\boldsymbol{o}_t$, and forget $\boldsymbol{f}_t$) to control the flow of information. Furthermore, the LSTM is enriched with peephole connections that link the memory cells to the gates. The LSTM outputs $\boldsymbol{h}_t$ are recurrently fed as the inputs. Given the input sequence $\boldsymbol{x}_t$, an LSTM layer computes the gates and memory cells activations to generate the outputs $\boldsymbol{h}_t$ sequentially. Moreover, [7] proposed that a projection matrix can be used to transform the outputs to a low dimension. Deep LSTM RNNs are formed by stacking multiple LSTM layers. More details of the LSTM-RNN formulation can be found in [33].

Both DNN and LSTM are special cases of the NNs and so many generic adaptation techniques developed for DNNs can be applied to LSTMs directly. However, the LSTM is more complicated and contains many components. Furthermore, it is uncertain if the recurrent loop has sufficiently exploited the long-range speaker characteristics and left less room for further improvement due to speaker adaptation.

In [12], we have conducted an extensive study of speaker adaptation for LSTM models. Two adaptation approaches were studied: updating existing LSTM components and inserting linear transformation above the LSTM output layers. In this paper, we further present applying the LRPD adaptation for the LSTM models and examine how various setups affect the recognition accuracy given a more powerful LSTM model.

Intuitively, we should insert the transformation matrix at the points where the flows of information afflux. We choose two points to implement this idea. First, we apply a linear transform above the output of the LSTM layers. This is equivalent to transforming the input to the next LSTM layer. Alternatively, we apply a linear transform above the concatenation of $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$, noting that both $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$ are fed together to the gates and memory cells of the LSTM layer.

4. ACCELERATING ADAPTATION IN TEST TIME

Learning efficiency is a critical challenge to real-world deployment of the adaptation algorithms. First, a cloud-based speech recognition system may serve millions of users. Updating the adaptation models of all users may last a few months using a single GPU. Second, if the speech recognizers are embedded in client sides, the adaptation will be constrained by the computational capacity of the client devices, where GPUs may not be available. Furthermore, it is also desirable to design an adaptation algorithm that is twofold fast, i.e., learning in a short time given a small amount of development data. It would open up the opportunity for online and incremental adaptation, where the speakers and environmental conditions change dynamically.

The gradient-based backpropagation (BP) algorithm is time consuming and performed serially over many iterations of the training data. The standard way of multiplying an $m \times n$ matrix by an $n \times p$ matrix has complexity of $O(mnp)$. Thus, it can be shown that the computational complexity of either forward or backward propagation per minibatch is $O(Nd)$, where N is the total number of model parameters and d is the minibatch size. The time for the model update step is negligible as it is performed once per minibatch.

For adaptation, we may need an extra forward pass for the KLD regularization to compute the output from the reference SI model. We usually run BP in multiple (10-20) sweeps through the development data. Thus, the computational complexity of the learning is $O(3NFh)$, where F is the size of the development data and h is the number of sweeps.

To reduce the computational cost, we notice that only weights in the adaptation layer are updated and the outputs from the layer being adapted, say l, do not change along the evolution of the adaptation model. Thus, we can eliminate the redundant computations on those layers below layer l during multiple sweeps by caching the outputs from layer l, once they are computed. The storage used for caching is affordable, since usually the number of adaptation utterances is less than hundreds and the dimension of the layer being adapted is less than thousands. This method is referred to as the bottleneck (BN) caching, though we can apply it on layers other than BN. The algorithm is summarized as follows:

1. Split the SI model into two sub-models along the lowest adaptation layer.
2. Dump the BN output of the lower sub-model over all the development data.
3. Perform adaptation on the upper sub-model using the BN features as input.

The computational complexity is reduced to $O((N_{lo}+3N_{up}h)F)$, where N_{lo} and N_{up} are the sizes of the lower and upper sub-models, respectively. If the adaptation layer is inserted just above the last nonlinear hidden layer, the optimization problem is casted as logistic regression, whose objective function is convex. This helps to make the optimization more efficient and reduce the number of training iterations, particularly with 2-nd order optimization techniques.

5. EXPERIMENTS AND RESULTS

The proposed methods are evaluated on a Microsoft internal Windows Phone short message dictation (SMD) task using DNN and LSTM models, respectively. The SMD data set consists of 7 speakers, giving a total number of 20,203 words. A separate development set of 200 sentences per speaker is used for model adaptation.

5.1. Adaptation of DNNs

We first report experiments using the SVD-based DNN acoustic models. The baseline DNN model is trained with 300hr voice search (VS) and SMD data. The SI GMM-HMM acoustic model has approximately 288K Gaussian components and 5976 senones trained with the MLE procedure, followed by fMPE and BMMI. The baseline SI CD-DNN-HMM system takes as input a 22-dimension mean-normalized log-filter bank feature with up to second-order derivatives and a context window of 11 frames, forming a vector of 726-dimension (66×11) input. On top of the input layer there are 5 hidden layers with 2048 units for each. The output layer has a dimension of 5976. We convert the full-size DNN model to the SVD DNN model by doing SVD on all the matrices except the one between the input and the first hidden layer, and keep 40% of total singular values. The numbers of units on the linear layers after SVD are 256, 272, 224, 256, and 368, from bottom to top. We then retrained the SVD model and obtained comparable accuracy to the full-size model. The baseline SI SVD DNN systems achieve 21.43% WER on the 7-speaker test set. More details of SVD-based DNN model training can be found in [19]. This DNN system is the same as the one used in [24], where the LRPD is conducted for supervised adaptation. In this work, we move forward and evaluate the performance of the eLRPD adaptation for both supervised and unsupervised adaptation. The Computational Network Toolkit (CNTK) [34] is used for neural network training.

Table 1: Footprints and WERs (in %) for the linear, LRPD, and eLRPD adaptations applied above the third BN layer of the SVD DNN.

Model	# SD Params.	Super.	Unsup.
Linear	50.4K	17.07	18.75
LRPD, c=5	2.6K	17.92	19.36
LRPD, c=10	4.8K	17.73	19.16
LRPD, c=20	9.2K	17.37	18.92
eLRPD, c=30	1.1K	18.25	19.43
eLRPD, c=50	2.7K	17.74	19.12
eLRPD, c=100	10.0K	17.19	18.82

The adaptation is applied above the third BN layer with 224 units of the SVD DNN, as it is shown in [35] that adapting intermediate layer provides more benefits than adapting boundary layers. Naturally, applying a full transformation upon the BN layer of SVD DNN model leads to the SVD BN adaptation [20]. We will see whether the eLRPD adaptation can further reduce and improve over an already very compact SVD BN adaptation. The KLD regularization [11] is applied, where the regularization weight is empirically set to 0.1 and 0.2 in supervised and unsupervised setups, respectively.

Table 1 compares the speaker-specific footprints and WERs for the linear, LRPD, and eLRPD adaptation of different configurations. It is observed that the linear adaptation produces the best WER of 17.07% for supervised and 18.75% for unsupervised setups, respectively. This translates to 20.3% and 12.5% relative improvement over the baseline SI model. Compared with the linear adaptation, the eLRPD ($c = 100$) reduces the SD parameters by 80% with only a small degradation in accuracy (less than 1% relative). It also performs slightly better than the LRPD ($c = 20$) in accuracy, both having the comparable footprints. When we reduce the model sizes further, both the methods begin to degrade in performance. Meanwhile, the eLRPD exhibits better compression effectiveness. Specifically, the eLRPD ($c = 50$) achieves the similar performance to the LRPD ($c = 10$), while saving the footprint by 44%.

Table 2: Learning time used during adaptation using the BN caching. *# Params.* denotes the footprint of the upper sub-model in the BN caching.

	# Params.	GPU	CPU
No caching	8.2M	18s	372s
Caching 3-rd BN layer	4.3M	9s	199s
Caching 5-th BN layer	2.2M	5s	106s

In the second experiment, we investigate how the BN caching affects the adaptation speed of the DNN models. Table 2 summarizes the learning time by inserting a linear layer above the 3-rd and 5-th BN layers. The training sweeps 10 passes of 50 utterances. The training is carried out on a Dell Precision T3600 workstation of 16 cores and a single NVIDIA Tesla K20X GPU, respectively. We have observed that the BN caching reduces the learning time by half on the 3-rd BN layer using the GPU and CPUs in out setups. The savings in time are almost proportional to the savings in the upper sub-model size during the BP training. In the extreme case, if we adapt the last BN layer (5-th), the BN caching may speed up the learning by about a factor of 3.5.

5.2. Adaptation of LSTM-RNNs

We further evaluate the speaker adaptation on the same SMD task using a more powerful LSTM-RNN model. The training data consist of 2600hr live US English data. The SI LSTM acoustic model is formed by stacking 4 LSTM layers which are followed by the softmax layer [7]. Each LSTM layer has 1024 memory cells and the output size of each LSTM layer is reduced to 512 by linear projection. The acoustic feature is the 80-dimensional static log filter-bank (LFB). The softmax layer has a dimension of 5980. The LSTM-RNN is trained to minimize the frame-level cross-entropy criterion using the truncated back propagation through time (BPTT) algorithm (back to 20 frames). There is no frame stacking, and the output HMM state label is delayed by 5 frames. When training LSTM, the backpropagation through time (BPTT) step is 20. A trigram LM is used for decoding with around 8M n-grams. The baseline LSTM model achieves 14.75% WER on the SMD task, which is significantly better than the DNN used in the previous experiments. Many factors contribute to this gain including the superiority in topology of the LSTM-RNN over the DNN, the involvement of much more training data, and the use of a more powerful LM.

Table 3: Footprints and WERs (in %) for the linear, LRPD, and eLRPD adaptations applied on the LSTM model.

Layer adapted	Model	# SD Params.	Super.	Unsup.
x_t^4	Linear	256K	13.65	14.29
$\{x_t^4, h_{t-1}^4\}$		1.0M	13.71	14.16
h_t^4		256K	13.59	14.27
h_t^4	LRPD, c=10	11.0K	13.87	14.18
	LRPD, c=20	21.0K	13.74	14.06
h_t^4	eLRPD, c=50	2.9K	13.89	14.23
	eLRPD, c=100	10.3K	13.62	14.12

Table 3 compares the results of various adaptation setups for adapting LSTM-RNN. First, we investigate the effects of inserting linear transformations at different positions of the last LSTM layer. Other LSTM layers are not reported, as it is shown in [12], adapting top hidden layers for the LSTM-RNN model is more effective than adapting lower layers. Three positions are probed, x_t^4, h_t^4, and a concatenation of $\{x_t^4, h_{t-1}^4\}$. All these configurations produce the similar performance, around 8% relative improvement for supervised setup and around 4% for unsupervised setup over the baseline LSTM-RNN. The gain we achieve here becomes smaller than the gain using the DNN model. We conjecture that the recurrent topology of the LSTM-RNN make it more effective to capture and normalize long-range speaker characteristics than the DNN. Moreover, the LSTM-RNN learns richer speaker variations through a large amount of training data and thus generalizes better to unseen speakers. The gain reported here is also small compared with that reported in [12]. This is because we established a much stronger SI LSTM baseline than the one in [12] by using more training data and more complicated training strategies.

We next apply the different adaptation methods at the output of the last hidden layer h_t^4, since we can greatly speed up the learning by caching h_t^4. It is observed that the LRPD and eLRPD adaptation for LSTM-RNNs has a similar trend to those for the DNN. Specifically, the eLRPD ($c = 100$) reduces the SD footprint by 96% over the linear adaptation without loss of accuracy. It also exhibits better compression effectiveness than the LRPD adaptation.

6. CONCLUSION

In this paper, we have extended the previously proposed LRPD adaptation into the subspace-based approach to further reduce the speaker-specific footprint. The LRPD restructures the adaptation matrix as a superposition of a diagonal matrix and a product of two low-rank matrices. The eLRPD adaptation inserts another SD matrix between the two low-rank matrices. To speed up the adaptation in test time, we proposed the BN caching approach to eliminate the redundant computations during multiple sweeps of development data. Experimental results on the SMD task showed that the eLRPD adaptation can reduce the SD footprint by 82% for the SVD DNN and 96% for the LSTM-RNN compared with the linear adaptation, while maintaining the comparable accuracy. Meanwhile, the BN caching can achieve up to 3.5 times speedup for adaptation at no loss of recognition accuracy.

7. REFERENCES

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

[2] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. Interspeech*, 2012, pp. 2578–2581.

[3] L. Deng, J. Li, J.-T. Huang, et al., "Recent advances in deep learning for speech research at Microsoft," in *Proc. ICASSP*, 2013, pp. 8604–8608.

[4] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, 2011, pp. 30–35.

[5] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[6] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 273–278.

[7] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, 2014.

[8] Y. Miao, J. Li, Y. Wang, S.-X. Zhang, and Y. Gong, "Simplifying long short-term memory acoustic models for fast training and decoding," in *Proc. ICASSP*, 2016, pp. 2284–2288.

[9] D. Yu and L. Deng, "Adaptation of deep neural networks," in *Automatic speech recognition: a deep learning approach*, pp. 193–215. Springer, 2015.

[10] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Proc. ICASSP*, May 2013, pp. 7947–7951.

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

[12] C. Liu, Y. Wang, K. Kumar, and Y. Gong, "Investigations on speaker adaptation of LSTM RNN models for speech recognition," in *Proc. ICASSP*, 2016.

[13] 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. Eurospeech*, 1995, pp. 2171–2174.

[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. Interspeech*, 2010, pp. 526–529.

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

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

[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. Workshop on Spoken Language Technology*, 2012, pp. 366–369.

[18] Y. Miao and F. Metze, "On speaker adaptation of long short-term memory recurrent neural networks," in *Proc. Interspeech*, 2015.

[19] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. Interspeech*, 2013, pp. 2365–2369.

[20] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *Proc. ICASSP*, 2014, pp. 6409–6413.

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

[22] Y. Zhao, J. Li, J. Xue, and Y. Gong, "Investigating online low-footprint speaker adaptation using generalized linear regression and click-through data," in *Proc. ICASSP*, 2015, pp. 4310–4314.

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

[24] Y. Zhao, J. Li, and Y. Gong, "Low-rank plus diagonal adaptation for deep neural networks," in *Proc. ICASSP*, 2016.

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

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

[27] D. Yu, X. Chen, and L. Deng, "Factorized deep neural networks for adaptive speech recognition," in *International workshop on statistical machine learning for speech processing*, 2012.

[28] T. Tan, Y. Qian, M. Yin, Y. Zhuang, and K. Yu, "Cluster adaptive training for deep neural network," in *Proc. ICASSP*. IEEE, 2015, pp. 4325–4329.

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

[30] L. Samarakoon and K. C. Sim, "Factorized hidden layer adaptation for deep neural network based acoustic modeling," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 12, pp. 2241–2250, Dec. 2016.

[31] J. Li, J.-T. Huang, and Y. Gong, "Factorized adaptation for deep neural network," in *Proc. ICASSP*, 2014, pp. 5574–5578.

[32] 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 Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 12, pp. 1713–1725, 2014.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, pp. 1735–1780, 1997.

[34] A. Agarwal, E. Akchurin, C. Basoglu, et al., "An introduction to computational networks and the computational network toolkit," Tech. Rep. MSR-TR-2014-112, Microsoft, 2014.

[35] K. Kumar, C. Liu, K. Yao, and Y. Gong, "Intermediate-layer DNN adaptation for offline and session-based iterative speaker adaptation," in *Proc. Interspeech*, 2015, pp. 1091–1095.