Interspeech 2018
2-6 September 2018, Hyderabad

Domain Adaptation Using Factorized Hidden Layer for Robust Automatic Speech Recognition

Khe Chai Sim, Arun Narayanan, Ananya Misra, Anshuman Tripathi, Golan Pundak, Tara N. Sainath, Parisa Haghani, Bo Li, Michiel Bacchiani

Google, USA

{khechai,arunnt,amisra,anshumant,golan,tsainath,parisah,boboli,michiel}@google.com

Abstract

Domain robustness is a challenging problem for automatic speech recognition (ASR). In this paper, we consider speech data collected for different applications as separate domains and investigate the robustness of acoustic models trained on multi-domain data on unseen domains. Specifically, we use Factorized Hidden Layer (FHL) as a compact low-rank representation to adapt a multi-domain ASR system to unseen domains. Experimental results on two unseen domains show that FHL is a more effective adaptation method compared to selectively fine-tuning part of the network, without dramatically increasing the model parameters. Furthermore, we found that using singular value decomposition to initialize the low-rank bases of an FHL model leads to a faster convergence and improved performance.

Index Terms: speech recognition, domain adaptation, factorized hidden layer

1. Introduction

Automatic speech recognition (ASR) systems are typically trained in a data-driven manner and therefore, are susceptible to performance degradation due to the mismatch between training and testing conditions, such as speaker, recording channel and acoustic environment. State-of-the-art ASR systems use deep neural networks (DNNs) for acoustic modeling [1, 2] and/or language modeling [3]. A wide range of DNN adaptation techniques for robust ASR are described in [4, 5].

In practice, it is often not possible to correctly identify the specific attributes (*e.g.* speaker and recording channel) of a given speech data. However, these attributes are somewhat correlated with the applications. For example, voice search [6] queries are typically short, near-field and recorded using a mobile device, while Google Home queries [7] are far-field and multi-channel. YouTube video captioning [8], on the other hand, deals with much longer speech data with different recording quality depending on the genre. Therefore, it is reasonable and convenient to consider each application as a separate domain.

Although it is possible to train domain-specific acoustic models given sufficient training data, this approach does not scale well to a large number of domains. Therefore, it is important to consider a compact adaptation method with a smaller number of domain-specific parameters. In this paper, we investigate the use of Factorized Hidden Layer (FHL) [9] to impose a low-rank constraint to reduce the number of domain-specific parameters. Previously, FHL has been extensively studied for speaker adaptation [10, 11, 12, 13]. FHL represents the parameters as a weighted linear interpolation of bases. For a matrix, such as the DNN weights, the bases can be represented by low-rank matrices to achieve a more compact model. For domain adaptation, we use a one-hot encoding for the interpolation weights such that each basis corresponds to one domain. The rank of the bases is adjusted to control the number of domain-specific parameters.

The remainder of this paper is organized as follows. Section 2 presents an overview and prior work on the domain-robust automatic speech recognition. Section 3 describes the factorized hidden layer (FHL) adaptation method and its application to domain adaptation. Section 4 presents the experimental setup and results.

2. Domain Robustness

Automatic speech recognition systems are sensitive to the data mismatch problem where the characteristics of the data at deployment time deviate from those of the training data. Acoustic mismatch can be caused by variability due to speaker (accent, speaking rate), channel (microphone, bandwidth, encoding) and environment (background noise, room condition).

Since there is no clear definition for what constitutes a domain, we will consider data for each application as a domain. The goal is to build an ASR system that is robust to domain mismatch. A simple approach is to train a *domain-invariant* model that generalizes well to unseen domains by using training data from multiple domains. This is referred to as multi-style training [14, 15, 16]. This approach works well as a general-purpose ASR system when the target domain cannot be explicitly determined during deployment.

To cope better with a new domain, especially one that is substantially different from the training domains, it helps to include some training data from the new domain. Instead of retraining the model to include a new domain, it is quicker to fine-tune a multi-domain model to this new domain. Regularization techniques, such as dropout [17], can be used to prevent over-fitting. Additional loss term, such as the KL-divergence between the outputs of the original and fine-tuned models [18], can be used to ensure that the fine-tuned model does not deviate significantly from the original model. However, this results in one full model for each domain, which does not scale well to a large number of domains.

Compact model adaptation techniques offer a better solution for minimizing the mismatch between the model and the target domain. Many DNN adaptation methods have been previously studied, including i-vector feature augmentation [19, 20], speaker-code adaptation [21], cluster adaptive training (CAT) [22, 23], factorized hidden layer [9], learning hidden unit contribution (LHUC) [24], singular value decomposition based subspace adaptation [25] and low rank plus diagonal (LRPD) [26, 27]. A more comprehensive review of the recent DNN adaptation techniques can be found in [4].

3. Factorized Hidden Layer

Factorized Hidden Layer (FHL) [9] is a deep neural network (DNN) adaptation technique that provides a structured and compact representation. FHL represents the adapted parameter as a weighted linear interpolation of bases. An FHL representation of a vector is given by:

$$\tilde{\boldsymbol{b}}^{(d)} = \boldsymbol{b}_0 + \sum_{i=1}^{N} \lambda_i^{(d)} \boldsymbol{b}_i, \tag{1}$$

where $\boldsymbol{b}_0$ and $\tilde{\boldsymbol{b}}^{(d)}$ represent the vector before and after adaptation. $\boldsymbol{b}_i$ denotes the ith basis whose corresponding interpolation weight is given by $\lambda_i^{(d)}$. N is the total number of bases. Superscript (d) denotes domain-dependent parameters. Similarly, a matrix is expressed as:

$$\begin{aligned} \tilde{\boldsymbol{W}}^{(d)} &= \boldsymbol{W}_0 + \sum_{i=1}^{N} \lambda_i^{(d)} \boldsymbol{W}_i & (2) \\ &\approx \boldsymbol{W}_0 + \sum_{i=1}^{N} \lambda_i \boldsymbol{U}_i \boldsymbol{V}_i^{\top}, & (3) \end{aligned}$$

where $\boldsymbol{W}_0$ and $\tilde{\boldsymbol{W}}^{(d)}$ represent the matrix before and after adaptation. $\boldsymbol{W}_i$ denotes the ith basis, which are often approximated using a low-rank matrix given by $\boldsymbol{U}_i\boldsymbol{V}_i^{\top}$. If $\tilde{\boldsymbol{W}}^{(d)}$, $\boldsymbol{W}_0$ and $\boldsymbol{W}_i$ are matrices of size $N \times M$, then $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ are matrices of size $N \times R$ and $M \times R$, respectively, where R corresponds to the rank of $\boldsymbol{U}_i\boldsymbol{V}_i^{\top}$.

In this paper, we use a one-hot encoding to represent the interpolation weights such that each basis corresponds to one domain. This is different from the previous FHL work [9, 10, 11, 12, 13] where the interpolation weights are adapted. The FHL-adapted vector and matrix can then be written as:

$$\begin{aligned} \tilde{\boldsymbol{b}}^{(d)} &= \boldsymbol{b}_0 + \boldsymbol{\Delta}_b^{(d)} & (4) \\ \tilde{\boldsymbol{W}}^{(d)} &= \boldsymbol{W}_0 + \boldsymbol{\Delta}_w^{(d)} \approx \boldsymbol{W}_0 + \boldsymbol{\Delta}_u^{(d)} \boldsymbol{\Delta}_v^{(d)\top}, & (5) \end{aligned}$$

where $\boldsymbol{\Delta}_b^{(d)}$ and $\boldsymbol{\Delta}_w^{(d)}$ are the domain-dependent offset vector and matrix, respectively. $\boldsymbol{\Delta}_u^{(d)}\boldsymbol{\Delta}_v^{(d)\top}$ is a low-rank approximation of the offset matrix. The number of columns for $\boldsymbol{\Delta}_u^{(d)}$ and $\boldsymbol{\Delta}_v^{(d)}$ can be adjusted to control the rank of the effective offset matrix and the complexity of the adaptation model. Unlike other work on low-rank model compression [28, 25, 29], we apply low-rank approximation only to the domain-dependent parameters without compromising the performance of the base model.

3.1. FHL Adaptation for LSTM

Long Short-Term Memory (LSTM) [30, 31] is commonly used in acoustic modeling for state-of-the-art automatic speech recognition systems [31]. Previously, FHL has been applied to LSTM for speaker adaptation [12]. An LSTM model is described by the following equations:

$$\begin{aligned} \boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{ri}\boldsymbol{r}_{t-1} + \boldsymbol{W}_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i\right) \\ \boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{rf}\boldsymbol{r}_{t-1} + \boldsymbol{W}_{cf}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f\right) \\ \boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh\left(\boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{W}_{rc}\boldsymbol{r}_{t-1} + \boldsymbol{b}_c\right) \\ \boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ro}\boldsymbol{r}_{t-1} + \boldsymbol{W}_{co}\boldsymbol{c}_{t-1} + \boldsymbol{b}_o\right) \\ \boldsymbol{m}_t &= \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right) \\ \boldsymbol{r}_t &= \boldsymbol{W}_{mr}\boldsymbol{m}_t \end{aligned}$$

Table 1: *Amount of train/adaptation data for each dataset.*

Dataset	Applications	Hours
Train	YouTube	117k
	Dictation	18k
	Voice Search	17k
	Google Home	8k
	Call-center	1k
Adaptation	Telephony	689
	Voice Mail	3k
Evaluation	Telephony	16
	Voice Mail	8

where t is the time step and σ is the sigmoid function. $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$, $\boldsymbol{c}_t$, $\boldsymbol{m}_t$ and $\boldsymbol{r}_t$ are the vectors for input gate, forget gate, output gate, cell state, cell output, and projection values respectively. $\boldsymbol{W}_{**}$ are weight matrices and $\boldsymbol{b}_*$ are bias vectors. All the peephole weight matrices, $\boldsymbol{W}_{c*}$, are diagonal. When coupled input-forget gates are used [32], $\boldsymbol{W}_{*f} = \boldsymbol{I} - \boldsymbol{W}_{*i}$. The bias vectors and the diagonal vector of the peephole weight matrices can be adapted using Eq. 4. Likewise, the full weight matrices, $\boldsymbol{W}_{x*}$ and $\boldsymbol{W}_{r*}$, can be adapted using Eq. 5. In TensorFlow [33], the $\boldsymbol{W}_{x*}$ and $\boldsymbol{W}_{r*}$ matrices are stacked together to form a single weight matrix for more efficient matrix multiplication:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_{xi} & \boldsymbol{W}_{ri} \\ \boldsymbol{W}_{xf} & \boldsymbol{W}_{rf} \\ \boldsymbol{W}_{xc} & \boldsymbol{W}_{rc} \\ \boldsymbol{W}_{xo} & \boldsymbol{W}_{ro} \end{bmatrix} \tag{6}$$

We apply FHL to $\boldsymbol{W}$ instead of the individual weight matrices.

3.2. Parameter Initialization

The parameters $\boldsymbol{b}_0$ and $\boldsymbol{W}_0$ can be initialized from an unadapted baseline system and fixed during subsequent domain adaptation. The domain-specific parameters ($\boldsymbol{\Delta}_b^{(d)}$, $\boldsymbol{\Delta}_u^{(d)}$ and $\boldsymbol{\Delta}_v^{(d)}$) are fine-tuned using data from a target domain. Instead of a zero or random initialization, a better way to initialize these parameters is to minimize the approximation error due to the low-rank approximation in Eq. 5 using Singular Value Decomposition (SVD). This is achieved by first fine-tuning the weight matrix without a low-rank constraint to obtain $\boldsymbol{W}^{(d)}$, from which we compute $\boldsymbol{\Delta}_w^{(d)}$). Then we compute the SVD decomposition of $\boldsymbol{\Delta}_w^{(d)}$ to obtain

$$\boldsymbol{\Delta}_w^{(d)} \approx \boldsymbol{\Delta}_u^{(d)} \boldsymbol{\Delta}_v^{(d)\top} \tag{7}$$

We keep the first r columns of $\boldsymbol{\Delta}_u^{(d)}$ and $\boldsymbol{\Delta}_v^{(d)}$ to achieve the desired rank.

4. Experimental Results

Table 1 summarizes the amount of data in the training, adaptation and evaluation data sets. The multi-domain training set consists of 161k hours of human-transcribed speech data from three main application sources. The majority of the data come from YouTube (117k hours or 73%). The remaining training data come from dictation, voice search, Google Home and Call-center. To evaluate the domain adaptation performance, we chose the Telephony and Voice Mail data sets as unseen domains. We extract 512-dimensional low-frame-rate log mel

Table 2: *Number of parameters for the baseline models.*

Layers	Per layer Count	Per layer %	Cumulative Count	Cumulative %
LSTM 0	4,723,712	(13.9)	4,723,712	(13.9)
LSTM 1	6,296,576	(18.5)	11,020,288	(32.4)
LSTM 2	6,296,576	(18.5)	17,316,864	(50.9)
LSTM 3	6,296,576	(18.5)	23,613,440	(69.4)
LSTM 4	4,132,608	(12.1)	27,746,048	(81.5)
Dense	6,299,648	(18.5)	34,045,696	(100.0)

Table 3: *WER performance of baseline models*

Model	WER Telephony	WER Voice Mail
Multi-domain	18.8	16.3
+ fine-tune	10.7	12.1

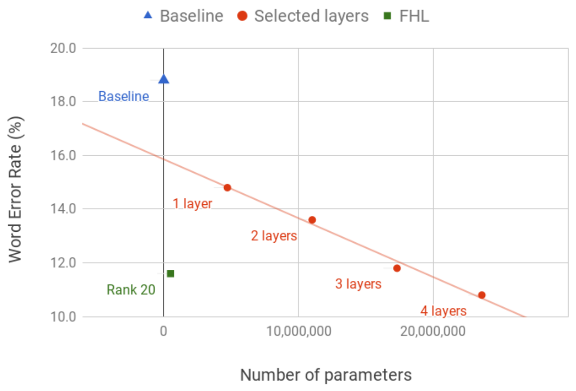

Figure 1: *WER vs. number of adaptation parameters for models adapted to the Telephony domain using the sMBR criterion.*

features [34] from speech waveform to train multi-layer LSTM acoustic models. The baseline model is trained using the multi-domain training data described in the previous section. The models are initially trained using the cross-entropy loss function for about 16 epochs, followed by 0.8 epochs of sMBR training [35].

The baseline model comprises 5 LSTM layers followed by a fully connected softmax layer. The first 4 LSTM layers have 1024 units while the fifth LSTM layer has 768 units. The output layer has 8192 units that correspond to the clustered triphones. Table 2 shows the number of model parameters associated with each layer. For domain adaptation, we applied FHL to all five LSTM layers (as described in Section 3.1). There are only about 500k adaptation parameters for FHL with rank 20 (which account for about .5% of the total number of parameters in the baseline model). With rank 100, the total number of adaptation parameters increases to about 2.4M (7.1%). Nevertheless, they are still much smaller compared to the number of parameters for one layer of the network (13.9% of the baseline model parameters).

Table 3 shows the word error rate (WER) performance of the baseline multi-domain model and the models fine-tuned to the respective unseen domains. It is evident from the results that due to the mismatch between the training and evaluation data sets, there is a substantial performance improvement when the entire model is further fine-tuned to the target domain. However, this results in having one full model for each domain, which may not be practical when dealing with a large number of domains. Next, we investigate adaptation using FHL to achieve a more compact per-domain model representation using low-rank approximation.

4.1. FHL versus Selective Fine-tuning

We compare FHL with selective fine-tuning to see how performance varies with the number of learnable parameters. For FHL, the basis rank is adjusted to vary the number of adaptation parameters. For selective fine-tuning, we adjust the adaptation complexity by updating only the parameters of the first n LSTM layers. Fig. 1 shows the trade off between word error rate performance (vertical axis) and the number of adaptation parameters (horizontal axis) for models trained with the sMBR criteria. The performance of the fine-tuned models improve roughly linearly with increasing number of fine-tuned layers. Fine-tuning all 4 LSTM layers (23.6M parameters) achieves 10.8% WER. On the other hand, FHL adaptation with rank 20 achieved 11.6% WER using only 0.5M domain-specific parameters.

4.2. Varying adaptation data sizes

Next we examine the performance of domain adaptation using smaller amount of adaptation data. Table 4 shows the WER of adapting to the Telephone and Voice Mail domains using 10 hours, 30 hours and 100 hours of data. To prevent the models from over fitting to the training subsets, we terminate training when the performance diverges on a held-out development set. When fine-tuning the full model, performance begins to diverge after around 40 epochs for the three training subsets. For the Telephony domain, increasing the fine-tuned layers improves performance up to 4 layers, beyond which there is sign of over-fitting. For Voice Mail, most of the gains come from fine-tuning just the first two layers.

The performance of FHL adaptation is worse than fine-tuning the entire model, even when there are only 10 hours of data. This is counter-intuitive as we expect a more compact model to be more robust when the amount of training data is small. This suggests that by starting from a well-trained model with appropriate learning rate and early stopping criterion, it is possible to fine-tune the entire model reliably using only a small amount of data. Nevertheless, FHL offers a much more compact model representation. The number of parameters associated with the FHL rank-100 bases is only 7.1% of the baseline model. Increasing the basis rank from 20 to 100 does not improve performance. This is surprising given that fine-tuning the model gave better performance. This suggests that the low-rank bases are much harder to learn. We will show in the next section that SVD initialization can mitigate this issue.

4.3. Random vs. SVD Initialization

Finally, we compared random and SVD initialization for the FHL bases. We obtained $\boldsymbol{\Delta}_w^{(d)}$ from a fine-tuned model and measure the normalized SVD approximation error of Eq. 7 as

Table 4: *Comparison of selective fine-tuning and FHL adaptation using different amount of adaptation data.*

Model	Amount of Data	WER Telephony	WER Voice Mail
Baseline	—	18.8	16.3
Fine-tune (1 layer)	10hrs	16.6	14.5
	30hrs	16.0	14.0
	100hrs	15.5	13.7
Fine-tune (2 layer)	10hrs	15.9	13.6
	30hrs	15.5	13.5
	100hrs	14.6	12.5
Fine-tune (3 layer)	10hrs	14.3	13.5
	30hrs	13.5	13.3
	100hrs	12.5	12.6
Fine-tune (4 layer)	10hrs	13.4	13.5
	30hrs	12.6	13.2
	100hrs	11.7	12.4
Fine-tune (5 layer)	10hrs	13.8	13.4
	30hrs	13.1	13.0
	100hrs	12.3	12.4
Fine-tune (all)	10hrs	14.0	13.4
	30hrs	13.2	12.9
	100hrs	12.3	12.4
FHL (rank 20)	10hrs	15.2	13.7
	30hrs	14.1	13.3
	100hrs	13.0	12.8
FHL (rank 100)	10hrs	15.2	13.7
	30hrs	14.0	13.3
	100hrs	13.0	12.7

follows:

$$E_{\texttt{svd}} = \frac{\left\| \mathbf{\Delta}_w^{(d)} - \mathbf{\Delta}_u^{(d)} \mathbf{\Delta}_v^{(d)\top} \right\|_F}{\left\| \mathbf{\Delta}_w^{(d)} \right\|_F} \tag{8}$$

where $||\cdot||_F$ denotes the Frobenius norm of a matrix. The error has a minimum value of zero when full rank is used (*i.e.* no low-rank approximation) and a maximum value of one when the rank is zero (*i.e.* no adaptation). Fig. 2 shows how the approximation error varies with rank when adapting to the telephony domain with different amount of data. The approximation error is smaller for models fine-tuned with smaller amount of data.

Table 5: *Comparing FHL adaptation (rank 100) using random versus SVD initialization.*

Dataset	Amount of Data	WER Random	WER SVD
Telephony	10hrs	15.2	14.7
	30hrs	14.0	13.7
	100hrs	13.0	12.5
Voice Mail	10hrs	13.7	13.5
	30hrs	13.3	13.0
	100hrs	12.7	12.3

In Table 5, we compared the WER performance of the FHL adapted models using random versus SVD initialization (as described in Section 3.2). The results show that SVD initialization consistently achieved better performance compared to random initialization by around 0.2–0.5% absolute. Fig. 3 shows the WER improvements of FHL rank-100 adaptation to the Telephony domain with 100 hours of data. It is clear from the figure that SVD initialization improves convergence substantially. Random initialization achieved 13.0% WER after 350 epochs. With SVD initialization, FHL adaptation achieved 13.1% WER after only 63 epochs and 12.5% WER after 130 epochs.

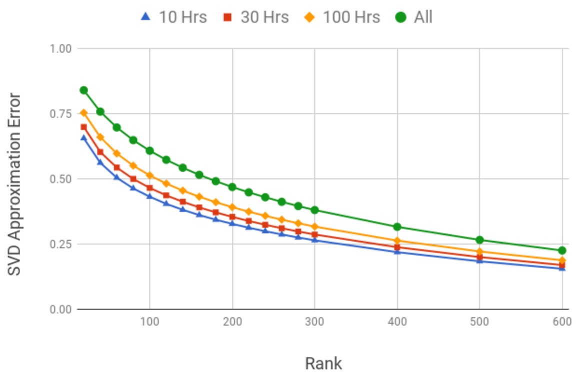

Figure 2: *SVD approximation error vs. FHL rank.*

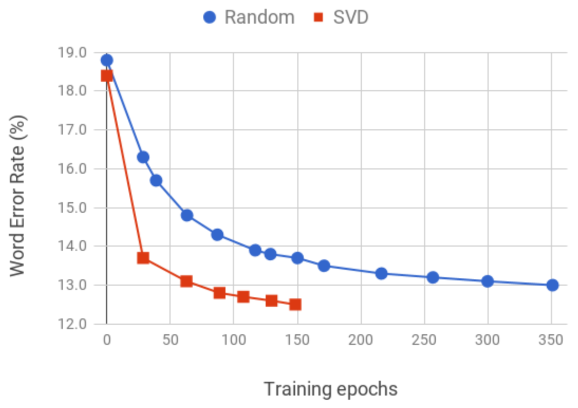

Figure 3: *Comparison of WER improvement with training epochs for random and SVD initialization of FHL bases.*

5. Conclusions

This paper presents an investigation into the use of Factorized Hidden Layer (FHL) to achieve compact model adaptation to unseen domains. Although increasing the diversity of the multi-domain training data improves performance on unseen domains, further adaptation using FHL on small amount of target domain data can significantly improve recognition performance without dramatically increasing the model parameters. We found that adaptation using a low-rank approximation for the LSTM weight matrices for all the layers is more effective compared to selectively fine-tune only part of the network. Furthermore, singular value decomposition can be used to initialize the low-rank approximation for faster convergence and better performance.

6. References

[1] 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,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2012.

[3] S. Kumar, M. Nirschl, D. Holtmann-Rice, H. Liao, A. T. Suresh, and F. Yu, “Lattice rescoring strategies for long short term memory language models in speech recognition,” *arXiv preprint arXiv:1711.05448*, 2017.

[4] K. C. Sim, Y. Qian, G. Mantena, L. Samarakoon, S. Kundu, and T. Tan, “Adaptation of deep neural network acoustic models for robust automatic speech recognition,” in *New Era for Robust Speech Recognition*, S. Watanabe, M. Delcroix, F. Metze, and J. Hershey, Eds. Sringer, 2017, ch. 9, pp. 219–243.

[5] D. Yu and J. Li, “Recent progresses in deep learning based acoustic models,” *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 396–409, 2017.

[6] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, “Your word is my command: Google search by voice: a case study,” in *Advances in speech recognition*. Springer, 2010, pp. 61–90.

[7] L. Li, T. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. Chin, K. C. Sim, R. J. Weiss, K. Wilson, E. Variani, C. Kim, O. Siohan, M. Weintraub, E. McDermott, R. Rose, and M. Shannon, “Acoustic modeling for Google Home,” *INTERSPEECH*, pp. 399–403, 2017.

[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] L. Samarakoon and K. C. Sim, “Factorized hidden layer adaptation for deep neural network based acoustic modeling,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 12, pp. 2241–2250, 2016.

[10] ——, “On combining i-vectors and discriminative adaptation methods for unsupervised speaker normalization in DNN acoustic models,” in *Proc. ICASSP*. IEEE, 2016, pp. 5275–5279.

[11] ——, “Low-rank bases for factorized hidden layer adaptation of DNN acoustic models,” in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 652–658.

[12] L. Samarakoon, B. Mak, and K. C. Sim, “Learning factorized transforms for unsupervised adaptation of LSTM-RNN acoustic models,” *Proc. INTERSPEECH*, pp. 744–748, 2017.

[13] L. Samarakoon, K. C. Sim, and B. Mak, “An investigation into learning effective speaker subspaces for robust unsupervised DNN adaptation,” in *Proc. ICASSP*. IEEE, 2017, pp. 5035–5039.

[14] R. Lippmann, E. Martin, and D. Paul, “Multi-style training for robust isolated-word speech recognition,” in *Proc. ICASSP*, vol. 12. IEEE, 1987, pp. 705–708.

[15] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani, “Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home,” in *Proc. INTERSPEECH*, 2017.

[16] A. Narayanan, A. Misra, K. C. Sim, G. Pundak, A. Tripathi, M. Elfeky, P. Haghani, T. Strohman, and M. Bacchiani, “Toward domain invariant speech recognition: An empirical study,” *submitted to INTERSPEECH*, 2018.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

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

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

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

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

[22] 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.

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

[24] P. Swietojanski and S. Renals, “Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models,” in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 171–176.

[25] 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*. IEEE, 2014, pp. 6359–6363.

[26] Y. Zhao, J. Li, and Y. Gong, “Low-rank plus diagonal adaptation for deep neural networks,” in *Proc. ICASSP*. IEEE, 2016, pp. 5005–5009.

[27] Y. Zhao, J. Li, K. Kumar, and Y. Gong, “Extended low-rank plus diagonal adaptation for deep and recurrent neural networks,” in *Proc. ICASSP*. IEEE, 2017, pp. 5040–5044.

[28] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, “Low-rank matrix factorization for deep neural network training with high-dimensional output targets,” in *Proc. ICASSP*. IEEE, 2013, pp. 6655–6659.

[29] G. Mantena and K. C. Sim, “Entropy-based pruning of hidden units to reduce DNN parameters,” in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 672–679.

[30] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in *INTERSPEECH*, 2014.

[32] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[33] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “TensorFlow: A system for large-scale machine learning.” in *OSDI*, vol. 16, 2016, pp. 265–283.

[34] G. Pundak and T. N. Sainath, “Lower frame rate neural network acoustic models.” in *Proc. INTERSPEECH*, 2016, pp. 22–26.

[35] B. Kingsbury, T. N. Sainath, and H. Soltau, “Scalable minimum bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization,” in *Proc. INTERSPEECH*, 2012.