DOMAIN AND SPEAKER ADAPTATION FOR CORTANA SPEECH RECOGNITION

Yong Zhao, Jinyu Li, Shixiong Zhang, Liping Chen, and Yifan Gong

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

{yonzhao; jinyli; zhashi; lipch; ygong}@microsoft.com

ABSTRACT

Voice assistant represents one of the most popular and important scenarios for speech recognition. In this paper, we propose two adaptation approaches to customize a multi-style well-trained acoustic model towards its subsidiary domain of Cortana assistant. First, we present anchor-based speaker adaptation by extracting the speaker information, i-vector or d-vector embeddings, from the anchor segments of 'Hey Cortana'. The anchor embeddings are mapped to layer-wise parameters to control the transformations of both weight matrices and biases of multiple layers. Second, we directly update the existing model parameters for domain adaptation. We demonstrate that prior distribution should be updated along with the network adaptation to compensate the label bias from the development data. Updating the priors may have a significant impact when the target domain features high occurrence of anchor words. Experiments on Hey Cortana desktop test set show that both approaches improve the recognition accuracy significantly. The anchor-based adaptation using the anchor d-vector and the prior interpolation achieves 32% relative reduction in WER over the generic model.

Index Terms: deep neural network, domain adaptation, speaker adaptation, anchor embedding

1. INTRODUCTION

The application of deep neural networks (DNNs) [1, 2, 3, 4] and recurrent neural networks (RNNs) [5, 6, 7, 8] has achieved tremendous success for large vocabulary continuous speech recognition (LVCSR). As speech recognition technologies continue to improve, voice assistant becomes ubiquitous on computers, mobile devices and smart speakers, such as Amazon's Alexa, Apple's Siri, Google Now and Microsoft's Cortana. Due to the popularity and importance of the voice assistant scenario, there are growing interests to leverage a large amount of data harvested from various speech application scenarios to boost the performance of voice assistant.

Domain adaptation approaches have been proposed to adapt an existing well-trained model to the target domain [9, 10, 11]. The entire model, or certain layers of the model, is directly updated [9, 12, 13]. To avoid overfitting, conservative training such as Kullback-Leibler divergence (KLD) regularization [10] is proposed. In the context of speaker adaptation, many techniques have been proposed to insert speaker-dependent (SD) transformation layers into the generic model. The layers being adapted can be either input features, hidden layers, or output layers [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]. One issue with these techniques is that they need to learn the adaptation parameters using the backpropagation algorithm [25] in the test time, which is considerably more expensive than the adaptation of the Gaussian mixture models (GMMs).

Recently, i-vector based adaptation [26, 27, 28] has gained popularity since i-vectors can be estimated independent of DNN training [29]. The acoustc features are augmented with i-vectors to form the input to the DNNs. Since i-vectors capture useful speaker acoustic properties, the DNNs are trained to be robust to speaker variations. The augmentation of the i-vectors can be considered as adapting the bias in the first hidden layer [30].

Speaker embeddings such as i-vectors can also be appended in any layers of deep networks to transform both the weight matrixes and biases [31, 32]. The mapping from speaker embedding to the SD transformations can be linear or through a stack of multiple layers [33]. The main network and the adaptation components can be jointly trained in a speaker adaptive training (SAT) scheme. In [34], factorized hidden layers (FHL) constructs the SD transformation matrix as linear combination of rank-1 matrixes. The combination weights are initialized with i-vectors and optimized during training. The SD transformation matrix in FHL can be regarded as the low-rank plus diagonal (LRPD) decomposition [35]. In [36, 37], i-vectors are mapped through a network to element-wise scaling and bias parameters.

However, there remain challenges for deploying the i-vector based system in real-time speech application, where an i-vector is estimated for each utterance [28]. First, the i-vectors estimated at the utterance level are far more noisy compared with the speaker i-vectors. Second, the decoding does not start until the entire utterance is available and the i-vector is extracted. One solution that is particularly applicable in the voice assistant scenario is to extract the speaker information from the anchor words. Users employ the anchor words, or wake-up words, to activate the voice assistant and then make requests or ask questions. With the speaker embedding from the anchor words, the decoding can start immediately after the anchor segment is received. Moreover, in the presence of the determined speech content, the anchor embeddings are supposed to capture speaker and environmental characteristics more accurately, compared with the utterance level embeddings. In [38], both feature-based and model-based methods have been proposed to exploit the anchor embeddings to improve the model to combat the interfering speech.

In this paper, we investigate two approaches to customize a multi-style well-trained model towards its subsidiary domain of Cortana assistant. First, we propose an anchor-based speaker adaptation scheme by extracting the speaker embeddings, i-vectors or d-vectors, from the anchor segments of 'Hey Cortana'. The anchor embeddings are mapped to layer-wise SD parameters to control the transformations of multiple layers in both weight matrix and bias. The transformation matrix is defined in a low-rank plus diagonal (LRPD) decomposition [35]. It allows us to flexibly controls the number of adaptation parameters according to the available adaptation data. Second, we directly update the existing model parameters for the domain adaptation. We identify a subtle and often overlooked difference between the DNN adaptation and the GMM adaptation: the prior distribution should be updated along with the adaptation of the DNN model to compensate the label bias from the development data. Experiments conducted on Hey Cortana desktop test set show that both approaches produce significant reductions in WER over the SI model.

2. SPEAKER ADAPTATION USING ANCHOR EMBEDDING

In this section, we present the proposed framework for speaker adaptation using the speaker embeddings extracted from the anchor seg-

ments. In the traditional i-vector based system, i-vectors are extracted as speaker embeddings and augmented to acoustic feature vectors to form the input to a DNN. It amounts to adapting the bias in the first hidden layer. We extend the i-vector based system by fully leveraging the deep structure of the DNN model. The anchor embeddings are applied to control the adaptation of multiple layers in both weight matrices and biases.

Given a DNN with L hidden layers, the activation output, $\boldsymbol{h}^l$, at the l-th hidden layer is recursively defined as the nonlinear transformation of the $(l-1)$-th layer:

$$\boldsymbol{h}^l = \sigma(\boldsymbol{z}^l) = \sigma(\boldsymbol{W}^l \boldsymbol{h}^{l-1} + \boldsymbol{b}^l) \tag{1}$$

where $\boldsymbol{z}^l$ is the excitation vector, $\boldsymbol{W}^l$ is the weight matrix, $\boldsymbol{b}^l$ is the bias vector, and $\sigma(\cdot)$ is an element-wise sigmoid activation function. $\boldsymbol{h}^0 = \boldsymbol{x}$ is the input observation vector. The output layer is normalized by the softmax function to produce the posterior probability, $p(q|\boldsymbol{x})$, of senone q.

Fig. 1 illustrates a general structure of the proposed adaptation algorithm. The main and adaptation networks are shown in the black and red parts, respectively. The l-th layer is adapted based on the anchor embeddings as follows:

$$\boldsymbol{z}_s^l = \left(I + \boldsymbol{P}^l \boldsymbol{U}_s^l \boldsymbol{Q}^l\right) \boldsymbol{W}^l \boldsymbol{h}^{l-1} + \boldsymbol{v}_s^l + \boldsymbol{b}^l \tag{2}$$

where the weight matrix and bias are transformed through the SD parameters $\boldsymbol{U}_s^l$ and $\boldsymbol{v}_s^l$, respectively.

The term $\left(I + \boldsymbol{P}^l \boldsymbol{U}_s^l \boldsymbol{Q}^l\right)$ in Eq. (2), corresponding to the transformation of the weight matrix, is defined in a low-rank plus diagonal (LRPD) decomposition [35, 39]. We restructure the transformation matrix as a superposition of an identify matrix $\boldsymbol{I}$ and a product of three low-rank matrices $\boldsymbol{P}^l \boldsymbol{U}_s^l \boldsymbol{Q}^l$, where $\boldsymbol{P}^l$ and $\boldsymbol{Q}^l$ are the speaker-independent (SI) matrices connecting the SD matrix $\boldsymbol{U}_s^l$ with the size of $c \times c$ to the main network. When c is much smaller than the dimension of the layer being adapted, the LRPD can signicantly reduce the SD transformation parameters. Moreover, the LRPD contains the full and the diagonal transformation matrices as its special cases. It allows us to flexibly controls the number of adaptation parameters according to the available adaptation data.

The SD parameters $\boldsymbol{U}_s^l$ and $\boldsymbol{v}_s^l$ are mapped from the anchor embeddings $\boldsymbol{e}_s$ as follows:

$$\boldsymbol{U}_s^l = vec^{-1}(\boldsymbol{u}_s^l) = vec^{-1}(\boldsymbol{f}^l(\boldsymbol{e}_s)) \tag{3}$$

$$\boldsymbol{v}_s^l = \boldsymbol{g}^l(\boldsymbol{e}_s) \tag{4}$$

where $\boldsymbol{f}^l$ and $\boldsymbol{g}^l$ are auxiliary networks with $\boldsymbol{e}_s$ as input, and $vec^{-1}(\cdot)$ converts a vector to a matrix in terms of the columns. These auxiliary networks each consist of multiple fully-connected layers followed by the last linear layer and may share the bottom layers. The use of the auxiliary networks eliminates the need to learn the SD parameters in the test time. Reshaping the SD vector $\boldsymbol{u}_s^l$ into a matrix $\boldsymbol{U}_s^l$ allows us to make full control of the transformation matrices. This is in contrast to other methods [34, 40, 36], which map the speaker embeddings to the diagonal elements of the transformation matrices. Note that reshaping the SD vector is feasible when we significantly reduce the size of the SD matrix $\boldsymbol{U}_s^l$.

Moreover, the LRPD can be extended to subsume the transformations of both the weight matrix and bias. If we associate $\boldsymbol{Q}^l$ with a bias term $\boldsymbol{b}_q^l$, the low-rank part of the transformation becomes:

$$\boldsymbol{P}^l \boldsymbol{U}_s^l (\boldsymbol{Q}^l \boldsymbol{W}^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_q^l) = \boldsymbol{P}^l \boldsymbol{U}_s^l \boldsymbol{Q}^l \boldsymbol{W}^l \boldsymbol{h}^{l-1} + (\boldsymbol{b}_q^{l\mathrm{T}} \otimes \boldsymbol{P}^l) \boldsymbol{u}_s^l \tag{5}$$

where $\otimes$ is the Kronecker product. Thus, the transformation of both the weight matrix and bias depends on the single SD vector $\boldsymbol{u}_s^l$.

Although having the similar architecture as the FHL adaptation proposed in [34], the proposed framework differs from FHL in two aspects. First, in FHL, the i-vectors are used to initialize the diagonal elements of the SD transformation matrix and the full SD transformation still needs to be learned for improved performance. In contrast, we directly convert the SD parameter vectors into a full transformation matrix, forgoing the DNN learning in the test time. Second, FHL directly uses the i-vectors as the SD transformation vectors among multiple layers. However, we observe optimal recognition performance when SD vectors are mapped from the embeddings through separate networks. The proposed approach also subsumes the embedding-based SAT [36, 37], where the transformation is conducted through element-wise scaling and bias.

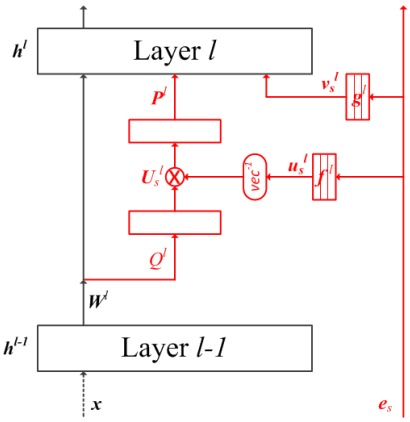

Fig. 1: Illustration of the model structure of the anchor-based speaker adaptation. The red part corresponds to the adaptation network appended to the main network.

The adaptation model is learned in a lightly adaptive training fashion. We first train the main network using the multi-style data and keep it fixed. Then we adaptively train the adaptation network using the target domain data with the anchor embeddings as a second input stream.

3. DOMAIN ADAPTATION WITH PRIOR INTERPOLATION

In a hybrid DNN-HMM system for speech recognition, the neural network is trained to predict the posterior probability, $p(q_t|\boldsymbol{x}_t)$, of each senone q_t. During decoding, the state posterior probability is divided by the prior probability, $p(q_t)$, of the state q_t to form the state likelihood $p(\boldsymbol{x}_t|q_t)$ used in the HMM:

$$p(\boldsymbol{x}_t|q_t) \propto p(q_t|\boldsymbol{x}_t)/p(q_t) \tag{6}$$

In contrast, a traditional GMM-HMM system directly employs the GMM to estimate the state likelihoods.

When it comes to model adaption, there is a subtle and often overlooked difference between the DNN adaptation and the GMM adaptation. Conceptually, the prior distribution should be updated along with the adaptation of the DNN model to compensate the label bias from the development data. When the target domain has similar senone coverage to the source domain or has very limited development data, this issue does not matter. It poses an issue when there is a remarkable difference in the senone coverage. For example, in the domain of Cortana assistant, a large amount of speech queries begin with the anchor phrase 'Hey Cortana'. Moreover, the speaking style and contents of Cortana queries are more conversational and functional than those from other scenarios such as voice search (VS) and short message dictation (SMD).

A natural choice to perform prior adaptation is to interpolate the prior distributions estimated from the source domain and the target domain.

$$\hat{p}(q_t) = (1-\rho)\tilde{p}(q_t) + \rho p^{SI}(q_t) \quad (7)$$

where ρ is the prior interpolation weight, and $p^{\text{SI}}(q_t)$ and $\tilde{p}(q_t)$ are the prior probabilities estimated from the SI domain and the development data. This is analogous to the Kullback-Leibler (KL) regularization proposed in [10], where the target posterior probability $\hat{p}(q_t|\boldsymbol{x}_t)$ is a linear interpolation of the distribution estimated from the SI model and the ground truth alignment of the adaptation data.

4. EXPERIMENTS AND RESULTS

The experiments were performed using Microsoft internal live US English data of 3,400hr. These data were collected through of a number of deployed speech services including VS, SMD, and Cortana assistant. Cortana assistant can be activated through the anchor phrase 'Hey Cortana'. Within the 3,400hr data, we selected 220hr Hey Cortana desktop utterances that begin with 'Hey Cortana' to adapt the generic SI model. Note that different from the conventional adaptation setup, the data set we used to train the SI model subsumes the target domain. Thus, the gains from adaptation should not be simply attributed to the significant acoustic mismatch between the source and target domains. The models were tested in a separate Hey Cortana desktop data set, which consists of 38,350 words from 6,139 utterances.

The first SI baseline is an 8-layer DNN model trained with the 3400hr data set. The input feature is the 80-dimension log-filterbank (LFB) feature with up to second-order derivatives, augmented with a context window of 11 frames. On top of the input layer there are 6 hidden layers each with 2,048 units followed by the last hidden layer with 4,096 units. The output layer consists of 9,801 senones. A second SI baseline is trained using the 220hr Hey Cortana desktop set. It consists of 5 hidden layers each with 2,048 units. The Computational Network Toolkit (CNTK) [41] is used for neural network training.

The embedding vectors are extracted from the anchor phrase 'Hey Cortana'. The average duration of the anchor segments is about 0.6 seconds. Two types of anchor embeddings are extracted to capture the speaker and environmental acoustic properties: i-vector and d-vector [42]. The i-vectors are trained using 13-dim MFCC static coefcients appended with the first and second derivatives. The universal background model (UBM) consists of 512 diagonal covariance Gaussians and a 100-dim i-vector is generated for each anchor segment. In addition, it has been shown that d-vectors can achieve comparable performance as i-vectors in the task of text-dependent speaker verification. A bottleneck DNN model is trained with speaker labels as targets in the output layer. The model consists of 4 hidden layers, each with 1024 units, followed a bottleneck layer of 100 units. The input is 41 spliced frames within the anchor segments. The output layer has 8,398 units representing training speakers. D-vectors are obtained by averaging the bottleneck outputs over the anchor segments.

Table 1 presents the results of several reference systems evaluated on the Hey Cortana desktop test set. A trigram language model with around 8 million n-grams is used for decoding. The two SI baselines trained with 3400hr and 220hr data achieve 16.36% and 20.20% WER, respectively. This confirms that the multi-style training usually outperforms training individual models from separate data sets. The SAT models in the last two rows are the standard i-vector based system, except that the two embeddings are extracted from the anchor segment other than the entire utterance. 'L1 ivec bias' in the table means that the bias of the first hidden layer L1 is adapted via the anchor i-vector. Both i-vector and d-vector based systems outperform the corresponding 220hr SI model by 13% and 18% relative WER reduction (WERR), respectively. Compared with the results obtained by Saon *et al.* [26] showing 10% WERR using the speaker i-vectors and by Senior and Lopez-Moreno [28] showing 4% WERR using the utterance i-vectors, it indicates that the anchor-level embedding can effectively capture and normalize the speaker and environmental variations for speech recognition. Moreover, the anchor d-vector performs better than the anchor i-vector.

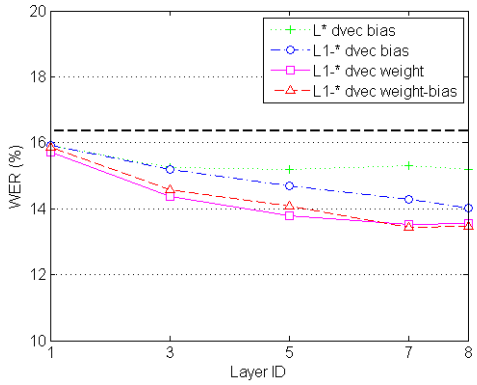

Fig. 2: WERs (%) for adapting biases and weight matrices of single layer and multiple layers from bottom to top using the anchor d-vectors. The dashed level line is the 3400hr SI baseline.

Table 1: WER (%) of the SI and SAT models on the Hey Cortana desktop test set.

Model	WER (%)
3400hr SI	16.36
220hr SI	20.20
220hr SAT, L1 ivec bias	17.49
220hr SAT, L1 dvec bias	16.64

4.1. Speaker adaptation using anchor embeddings

We first evaluate the proposed speaker adaptation approach using the anchor d-vectors. We keep fixed the main network trained using the 3400hr data set, and train the adaptation network in a lightly SAT fashion. The anchor embedding is mapped to the layer-wise adaptation vectors $\boldsymbol{u}_s^l$ and $\boldsymbol{v}_s^l$ through individual auxiliary networks. Each auxiliary network consists of 2 fully-connected hidden layers with 100 units per layer and the last linear layer with the right units determined by the adaptation network. The vector $\boldsymbol{u}_s^l$ is reshaped into $\boldsymbol{U}_s^l$ of size 10×10 for transforming the weight matrix. These settings are found to produce good adaptation results.

Fig. 2 shows the results of speaker adaptation by transforming single layer and multiple layers from bottom to top using anchor d-vectors. It is observed that adapting the biases of a single layer (L*) improves the performance by 3%-8% relative. These gains are less than what we have observed on the d-vector based SAT system using 220hr data. Second, adapting multiple layers (L1-*) outperforms adapting a single layer. Basically, the recognition accuracy improves continuously as the number of the adapted layers increases until the last hidden layer L7. Moreover, adapting the weight matrix only, Eq. (5), produces comparable performance as adapting both the weight matrix and bias, Eq. (2), with a smaller footprint. Both are consistently better than adapting the bias only. Specifically, adapting

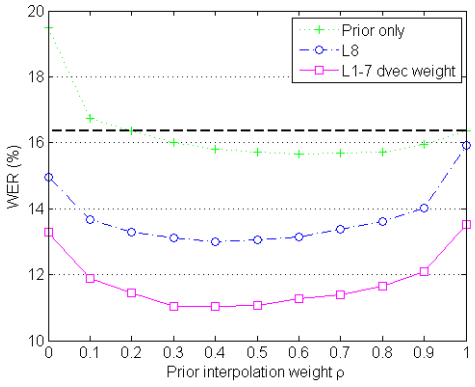

Fig. 3: WERs (%) against prior interpolation weights ρ for three adaptation models. The dashed level line is the SI baseline.

the weight matrices of the multiple layers L1-7 gives the WER of 13.51%, 18% relative reduction over the SI baseline.

4.2. Domain adaptation with prior interpolation

The second experiment is conducted by directly updating the existing layers without inserting SD layers. The goal is to examine how the prior interpolation impacts the adaptation performance. Fig. 3 compares the recognition results with different prior interpolation weights ρ for the SI model and the model with layer L8 updated (the model 'L1-7 dvec weight' will be discussed later). We can see directly adapting the softmax layer L8 with prior interpolation ($\rho = 0.5$) yields the WER of 13.06%, 20% relative reduction over the SI model. In contrast, the adaptation yields only 3% relative gain if the prior distribution is unchanged ($\rho = 1$). Moreover, adapting alone the priors of the SI model improves the performance by 4% relative, confirming that the estimation of the prior distribution should be consistent with that of the posterior probability in terms of data coverage.

Table 2 shows the list of top 10 words ranked by word count changes from the SI model to the model with layer L8 updated ($\rho = 1$) on the Hey Cortana desktop test set. It encloses the word counts of the reference transcripts, the SI model, and adapting layer L8 without and with the priors being updated, respectively. We observe that the adaptation can systematically rectify the errors committed by the SI model. Although the model, L8 ($\rho = 1$), improves little over the SI model, it reduces the recognition errors in 'have' and 'hello' at the expense of increased errors in words from the 'what' to 'anna' (3rd to 10th). The increased errors in 'what', 'hey', 'the', and 'what's' can be attributed to the higher occurrences of these words in the target domain than the generic domain. The increased errors in 'caught', 'nah', 'ne', and 'anna' are attributed to these words sharing the senones with 'Cortana'. After the prior interpolation, many such errors are effectively suppressed. This demonstrates that updating the priors has a significant impact when the target domain features high occurrence of anchor words.

Table 3 compares the results of the domain adaptation by updating multiple existing layers from top to bottom with $\rho = 0.5$. We can see adapting the top layer contributes most of the gain. Adapting more layers only slightly improves the performance.

4.3. Speaker adaptation with prior interpolation

In this section, we evaluate the anchor-based speaker adaptation more extensively by updating the priors, varying embedding vectors, and additionally updating the softmax layer. Fig. 3 shows the effect of the prior interpolation for the anchor-based model, L1-7 dvec weight. When $\rho = 0.5$, the improvement from adaptation (11.06% WER) can be further enlarged to 32% relative over the SI model, or 14% relative over the corresponding adaptation model without updaint the priors.

Table 4 gives more results. In these experiments, the prior interpolation weight ρ is set to 0.5. From the first row, it is confirmed that adaptation using the anchor d-vector outperforms the adaptation using the anchor i-vector. In the second row, we extend the anchor-based adaptation by further updating layer L8. However, for the i-vector system, this only yields slight gain over the one without updating L8, and the d-vector system shows no gain. We conjecture that deeply inserting adaptation layers from bottom to top may not only normalize the speaker and environmental variations, but also systematically shift the senone classification boundaries.

Table 2: Top 10 word count changes from the SI model to the model with layer L8 updated ($\rho = 1$) on Hey Cortana desktop test set.

	Ref	SI	L8 ($\rho = 1$)	L8 ($\rho = 0.5$)
WER (%)	–	16.36	15.92	13.06
have	61	575	111	117
hello	9	492	42	50
what	591	890	1067	764
hey	6212	6132	6306	6230
caught	0	1	102	23
nah	0	42	133	4
ne	0	15	98	3
the	1081	1203	1279	1184
what's	498	561	609	562
anna	0	13	55	5

Table 3: WERs (%) for the domain adaptation by updating multiple selected layers with prior interpolation $\rho = 0.5$.

	SI	L8	L7-8	L6-8	L1-8
WER (%)	16.36	13.06	12.99	12.98	12.90

Table 4: WERs (%) for the anchor-based speaker adaptation using i-vector and d-vector embeddings with prior interpolation $\rho = 0.5$.

Model	ivec	dvec
L1-7 weight	11.66	11.06
L8 + L1-7 weight	11.42	11.02

5. CONCLUSION

In this paper, we explored two approaches to adapt a multi-style well-trained model towards its subsidiary domain of Cortana assistant. We demonstrated that prior distribution should be updated along with the adaptation of the neural network to compensate the label bias from the development data. Updating the priors may have a significant impact when the target domain features high occurrence of anchor words. Moreover, we performed the anchor-based speaker adaptation by extracting the speaker embeddings from the anchor segments of 'Hey Cortana'. We proposed a deep adaptation framework where the anchor embedding controls the adaptation of multiple layers in both weight matrices and biases, along with the update of prior distribution. Experiments on Hey Cortana desktop test set showed that directly updating the softmax layer with prior interpolation yielded 20% relative reduction over the SI model. The anchor-based system using the anchor d-vector and the prior interpolation achieved 32% relative reduction over the SI model.

6. 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.

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

[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. IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

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

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

[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.

[9] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Proc. ICASSP*, 2013.

[10] 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.

[11] J. Li, M. L. Seltzer, X. Wang, R. Zhao, and Y. Gong, "Large-scale domain adaptation via teacher-student learning," in *Proc. Interspeech*, 2017.

[12] C. Liu, J. Li, and Y. Gong, "SVD-based universal DNN modeling for multiple scenarios," in *Proc. Interspeech*, 2015.

[13] T. Asami, R. Masumura, Y. Yamaguchi, H. Masataki, and Y. Aono, "Domain adaptation of DNN acoustic models using knowledge distillation," in *Proc. ICASSP*, 2017.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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. IEEE Workshop on Spoken Language Technology*, 2012.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] Z. Huang, S. M. Siniscalchi, I-F. Chen, J. Wu, and C.-H. Lee, "Maximum a posteriori adaptation of network parameters in deep models," in *Proc. Interspeech*, 2015.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *Cognitive modeling*. MIT Press, Cambridge, MA, 1988.

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

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

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

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

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

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

[32] 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.

[33] Y. Miao, H. Zhang, and F Metze, "Towards speaker adaptive training of deep neural network acoustic models," in *Proc. Interspeech*, 2014.

[34] 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.

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

[36] X. Cui, V. Goel, and G. Saon, "Embedding-based speaker adaptive training of deep neural networks," in *Proc. Interspeech*, 2017.

[37] L. Samarakoon and K. C. Sim, "Subspace LHUC for Fast Adaptation of Deep Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

[38] B. King, I-F. Chen, Y. Vaizman, Y. Liu, R. Maas, S. H. Parthasarathi, and B. Hoffmeister, "Robust speech recognition via anchor word representations," in *Proc. Interspeech*, 2017.

[39] Y. Zhao, J. Li, K. Kumar, and Y. Gong, "Extended low-rank plus diagonal adaptation for deep and recurrent neural networks," in *Proc. ICASSP*, 2017.

[40] X. Li and X. Wu, "I-vector dependent feature space transformations for adaptive speech recognition," in *Proc. Interspeech*, 2015.

[41] 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.

[42] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP*, 2014.