CONTEXT ADAPTIVE DEEP NEURAL NETWORKS FOR FAST ACOUSTIC MODEL ADAPTATION IN NOISY CONDITIONS

Marc Delcroix, Keisuke Kinoshita, Chengzhu Yu[*]*, Atsunori Ogawa, Takuya Yoshioka, Tomohiro Nakatani*

NTT Communication Science Laboratories, NTT corporation,
2-4, Hikaridai, Seika-cho (Keihanna Science City), Soraku-gun, Kyoto 619-0237 Japan
{marc.delcroix}@lab.ntt.co.jp

ABSTRACT

Deep neural network (DNN) based acoustic models have greatly improved the performance of automatic speech recognition (ASR) for various tasks. Further performance improvements have been reported when making DNNs aware of the *acoustic context* (e.g. speaker or environment) for example by adding auxiliary features to the input, such as noise estimates or speaker i-vectors. We have recently proposed a context adaptive DNN (CA-DNN), which is another approach to exploit the acoustic context information within a DNN. A CA-DNN is a DNN that has one or several factorized layers, i.e. layers that use a different set of parameters to process each acoustic context class. The output of a factorized layer is obtained by the weighted sum over the contribution of the different context classes, given weights over the context classes. In our previous work, the class weights were computed independently of the recognizer. In this paper, we extend our previous work by introducing the joint training of the CA-DNN parameters and the class weights computation. Consequently, the class weights and the associated class definitions can be optimized for ASR. We report experimental results on the AURORA4 noisy speech recognition task showing the potential of our approach for fast unsupervised adaptation.

Index Terms: Automatic speech recognition, Deep neural networks, Acoustic model adaptation, Context adaptive DNN, Factorized DNN

1. INTRODUCTION

The increased use of deep neural network (DNN) based acoustic models [1, 2] has created a crucial need for techniques for adapting DNNs to the *acoustic context* (e.g. speaker or environment) seen at test time. There has been much research on DNN adaptation, focusing mainly on three main directions, i.e. input feature transformation using transforms trained discriminatively [3, 4] or independently of the DNN [5, 6], direct adaptation or transformation of the DNN parameters [7–13], and using auxiliary *context features* representing the acoustic context (such as i-vectors or noise estimates) to the input of the DNN layers [14–19]. Among these approaches, exploiting auxiliary context features appears particularly promising for fast adaptation since only several seconds of speech may be sufficient to compute the auxiliary features. The success of these approaches reveals that performance gains can be achieved by making the DNN aware of the acoustic context. However, simply inputting the auxiliary features to a DNN may not necessarily be the best approach for exploiting context information. Other variations have been proposed such as inputting the auxiliary features to several layers or adding layers to transform the auxiliary features before inputting it to the DNN [17, 18].

[*]Chengzhu Yu is with the University of Texas at Dallas. He contributed to this work while he was an intern at NTT.

We have recently proposed a different approach for incorporating acoustic context into a DNN by making the DNN parameters directly dependent on the acoustic context. We called this approach context adaptive DNN (CA-DNN) [20]. A CA-DNN is realized by factorizing one or several layers into sub-layers associated with context classes or clusters. The output of a *factorized layer* is obtained as the weighted sum of the output associated with each sub-layer, weighted by context class weights. The proposed CA-DNN is thus related to cluster adaptive training (CAT) that was initially proposed for legacy GMM-HMM acoustic models [21] and recently revisited for DNN-based acoustic models [22, 23]. CAT also defines class/cluster dependent parameters and performs adaptation using a linear interpolation of the different class parameters. The main difference originates from the way the class weights are computed. CAT computes the class weights by optimizing likelihood [21] or cross entropy [22, 23] using adaptation data. The proposed CA-DNN derives the class weights from an external context representation such as i-vectors, which may arguably be less computationally demanding at test time. During training, the parameters of the network and the factorized layers are trained in a soft manner given the training data and the corresponding class weights. During testing, the CA-DNN is adapted by the weighted sum of its parameters given the class weights associated with the acoustic context of the test conditions.

In [20], we tested CA-DNN for the TIMIT continuous phoneme recognition task. We used class weights obtained by GMM clustering of utterance-based i-vectors. The i-vectors can be computed blindly during testing allowing fast unsupervised adaptation. However, since the class weights are computed independently of the recognizer, they do not achieve an optimal representation of the acoustic context for ASR. In this paper, we extend our previous work by introducing the joint learning of the CA-DNN parameters and the class weights. The class weights are derived from i-vectors that are transformed using an *ancillary neural network*, which is trained jointly with the CA-DNN. Consequently, we can obtain class weights and thus context class representations that are optimized for ASR. We tested our approach on the AURORA4 [24] noisy speech recognition task showing the potential for fast unsupervised adaptation.

The remainder of this paper is as follows. In Section 2, we review the principles of CA-DNN. In Section 3, we introduce the novel structure of CA-DNN, which includes an ancillary network for context class weights computation and elaborate on the joint training procedure. We then present experimental results in Section 5. Finally, Section 6 concludes the paper and discusses future research directions.

2. CONTEXT ADAPTIVE DNN

2.1. Notations

Before describing CA-DNN, let us introduce notations by reviewing a conventional DNN used for acoustic modeling as shown in Figure 1-(a). A DNN based acoustic model is trained to output the HMM state posterior probabilities given input speech features. The DNN consists of several hidden layers. The input of the i^{th} layer of a DNN is denoted by $\mathbf{x}^{(i-1)}$, where by definition $\mathbf{x}^{(0)}$ corresponds to the input features or input layer. The output of the i^{th} layer, $\mathbf{x}^{(i)}$, is obtained as,

$$\begin{aligned} \mathbf{x}^{(i)} &= \sigma(\mathbf{z}^{(i)}), \\ \mathbf{z}^{(i)} &= \mathbf{W}^{(i)}\mathbf{x}^{(i-1)} + \mathbf{b}^{(i)}, \end{aligned} \tag{1}$$

where $\mathbf{W}^{(i)}$ and $\mathbf{b}^{(i)}$ are the weight matrix and bias vector of the linear transformation associated with the i^{th} layer, and $\sigma()$ is the activation function. In this paper we employ sigmoid activation functions [2]. The activation function of the last layer I, is a softmax function as,

$$x_n^{(I)} = softmax(z_n^{(I)}) = \frac{e^{z_n^{(I)}}}{\sum_{n'} e^{z_{n'}^{(I)}}}, \tag{2}$$

where $x_n^{(I)}$ is the n^{th} element of the output of the last layer and $z_n^{(I)}$ is to the n^{th} element of vector $\mathbf{z}^{(I)}$.

2.2. Principles of CA-DNN

We have recently proposed to modify the structure of a conventional DNN to make its parameters adaptive to the acoustic context. Following the ideas of committee machines [25, 26], we introduced a CA-DNN, which is a DNN with one or several layers that are factorized, meaning that they are decomposed in sub-layers that are associated with different acoustic context classes. The output of a factorized layer is obtained as the weighted sum of the contribution of each sub-layer as,

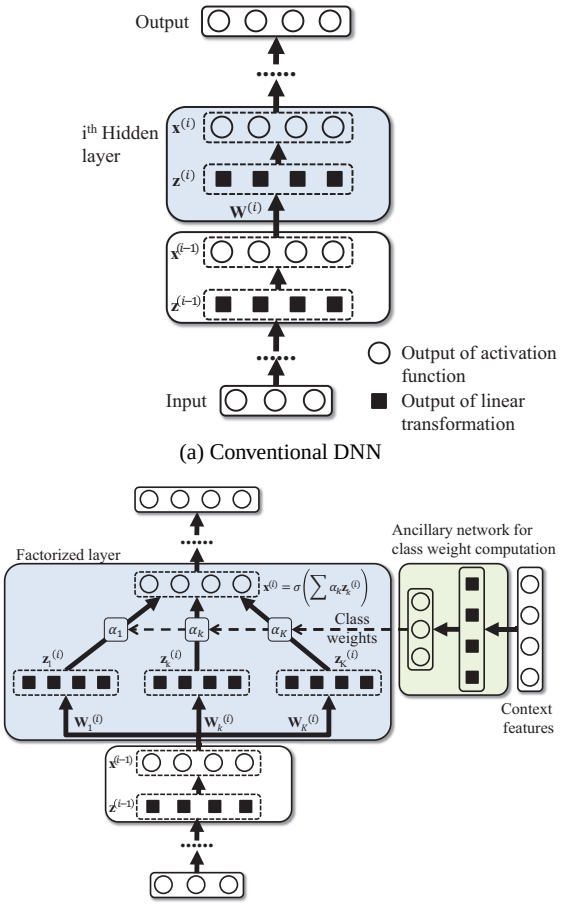
$$\mathbf{z}^{(i)} = \sum_{k=1}^{K} \alpha_k \underbrace{(\mathbf{W}_k^{(i)}\mathbf{x}^{(i-1)} + \mathbf{b}_k^{(i)})}_{\triangleq \mathbf{z}_k^{(i)}}, \tag{3}$$

where $\mathbf{W}_k^{(i)}$ and $\mathbf{b}_k^{(i)}$ and α_k are the weight matrix, bias vector and class weight associated with the k^{th} context class, respectively, and K is the number of context classes considered. The class weight vector $\boldsymbol{\alpha} = [\alpha_1 \dots \alpha_K]$ characterizes the acoustic context of a given utterance, which may depend on the task, e.g. the gender, speaker or acoustic environment (noise or reverberation). For example, $\boldsymbol{\alpha}$ can be obtained as the posteriors derived from speaker or environment clustering.

Note that it is also equivalent to express a CA-DNN as a conventional DNN whose parameters are obtained as the weighted sum of the parameters associated with each context class as,

$$\begin{cases} \mathbf{W}^{(i)} &= \sum_{k=1}^{K} \alpha_k \mathbf{W}_k^{(i)}, \\ \mathbf{b}^{(i)} &= \sum_{k=1}^{K} \alpha_k \mathbf{b}_k^{(i)}. \end{cases} \tag{4}$$

Equation (4) emphasizes that CA-DNN realizes the direct adaptation of the DNN parameters to the acoustic context given the context class weights. It can thus realize fast adaptation if the weights α_k can be computed from a small amount of speech data.



(a) Conventional DNN

(b) Context adaptive DNN with ancillary network for context class weights computation

Fig. 1. Schematic diagram of (a) a conventional DNN and (b) the proposed context adaptive DNN with the i^{th} layer replaced by a factorized layer. Note that the dotted boxes are included to emphasize intermediate steps in the computation of the output of a hidden layer (i.e. linear transformation and activation function) and are not actual hidden layers.

In [20] we proposed to use context weights computed independently of the recognizer. For example, we used posteriors obtained from GMM clustering of i-vectors as class weights. However, when the context weights are computed independently of the CA-DNN, they may not represent the acoustic context in an optimal way with respect to ASR performance. In the next section we propose an approach to obtain context weights and thus context class definitions optimized for ASR.

3. CA-DNN WITH JOINT TRAINING

3.1. Ancillary network for class weight computation

Instead of inputting directly class weights to the factorized layer of the CA-DNN [20], we introduce an ancillary network that computes class weights given context features. This ancillary network is equiv-

alent to a gating network used for example in mixture of experts models [25], except that its input consists of context features that are different from the input of the factorized layer. By interconnecting the ancillary network with the factorized layer, it becomes possible to train both networks jointly using error backpropagation. Consequently we can obtain classes and class weights optimized for ASR. Figure 1-(b) is a schematic diagram of a CA-DNN with the proposed ancillary network for class weight computation. We call such network a CA-DNN with joint training (CA-DNN-JT).

3.2. Joint training of CA-DNN and ancillary network

Let us denote by $\Theta \triangleq \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \ldots, \mathbf{W}_k^{(i)}, \mathbf{b}_k^{(i)} \ldots \mathbf{W}^{(I)}, \mathbf{b}^{(I)}\}$ the parameters of the CA-DNN and by $\Omega \triangleq \{\mathbf{V}^{(1)}, \mathbf{c}^{(1)} \ldots, \mathbf{V}^{(J)}, \mathbf{c}^{(J)}\}$ the parameters of the ancillary network, where $\mathbf{V}^{(j)}$ and $\mathbf{c}^{(j)}$ are the weight matrix and bias vector, and J is the number of layers of the ancillary network. We can jointly train Θ and Ω by optimizing the same objective function, $J(\Theta, \Omega)$. We use here the cross entropy criterion. We employ the stochastic gradient descent (SGD) with the error back-propagation algorithm to compute the gradient of the network parameters, by back-propagating the error signals to both the main and the ancillary networks.

3.2.1. Gradients w.r.t the CA-DNN parameters

The gradient of the parameters of a factorized layer can be obtained as,

$$\begin{cases} \dfrac{\partial J(\Theta,\Omega)}{\partial \mathbf{W}_k^{(i)}} &= \alpha_k \boldsymbol{\delta}^{(i)} (\mathbf{x}^{(i-1)})^T, \\ \dfrac{\partial J(\Theta,\Omega)}{\partial \mathbf{b}_k^{(i)}} &= \alpha_k \boldsymbol{\delta}^{(i)}, \end{cases} \tag{5}$$

where $\boldsymbol{\delta}$ is the back-propagated error that is expressed as,

$$\boldsymbol{\delta}^{(i)} = ((\mathbf{W}^{(i+1)})^T \boldsymbol{\delta}^{(i+1)}) \odot \sigma'(\mathbf{z}^{(i)}). \tag{6}$$

$\odot$ is the Hadamard product and $\sigma'(\mathbf{z}^{(i)})$ is the derivative of the activation function w.r.t. $\mathbf{z}^{(i)}$. Equation (5) is similar to the expression of the gradient for a conventional neural network [27] except for the introduction of the weighting term α_k. Moreover, Eq. (6) is identical to the expression for a conventional DNN but $\mathbf{z}^{(i)}$ should be calculated with Eq. (3) and $\mathbf{W}^{(i+1)}$ should be calculated with Eq. (4) if layer $i+1$ is factorized.

3.2.2. Gradient w.r.t. the ancillary network parameters

The gradient with respect to the ancillary network parameters, Ω, can be obtained with the chain rule as,

$$\frac{\partial J(\Theta,\Omega)}{\partial \Omega} = \frac{\partial J(\Theta,\Omega)}{\partial \boldsymbol{\alpha}} \frac{\partial \boldsymbol{\alpha}}{\partial \Omega}. \tag{7}$$

The gradient can thus be obtained with conventional error backpropagation but with the error signal at the output of the ancillary network expressed as,

$$\frac{\partial J(\Theta,\Omega)}{\partial \boldsymbol{\alpha}} = \sum_n \underbrace{\frac{\partial J(\Theta,\Omega)}{\partial x_n^{(i)}} \frac{\partial x_n^{(i)}}{\partial z_n^{(i)}}}_{=\delta_n^{(i)}} \frac{\partial z_n^{(i)}}{\partial \boldsymbol{\alpha}} \tag{8}$$

$$= \sum_n \delta_n^{(i)} \begin{bmatrix} z_{1,n}^{(i)} \\ \vdots \\ z_{K,n}^{(i)} \end{bmatrix}, \tag{9}$$

where $x_n^{(i)}$, $z_n^{(i)}$, $z_{k,n}^{(i)}$ and $\delta_n^{(i)}$ are the n^{th} component of the vectors $\mathbf{x}^{(i)}$, $\mathbf{z}^{(i)}$, $\mathbf{z}_k^{(i)}$ and $\boldsymbol{\delta}^{(i)}$. The implementation of the proposed method requires thus simple modifications of an existing DNN training implementation.

We investigate two types of activation functions for the output layer of the ancillary network, i.e. softmax and linear. The softmax activation function ensures that the class weights sum up to 1. Imposing this constraint is in accordance to our previous experiments where we used Gaussian posteriors as class weights but it is not necessary. Moreover, when using softmax activation, the gradient in Eq. (7) takes very small values because the derivative of the softmax activation is often close to zero. In contrast, using a linear activation does not impose any constraint on the weights but may mitigate the vanishing gradient problem of the softmax activation.

4. RELATION WITH PREVIOUS WORKS

We have already mentioned in the introduction that CA-DNN is directly related to cluster adaptive training [21–23] with the main difference originating from the class weight computation.

CA-DNN is also related to committee machines that distribute the learning among a number of expert networks and combine their outputs [25, 26]. In particular the mixture of experts model [28] also employs a gating network to compute the weights associated with each expert. A similar approach has been investigated for speech recognition [29]. These studies usually employ the same input features for the expert networks and the gating network. In our work, we employ context features that can represent the long-term acoustic context and that differ from the input speech features.

Finally, factorization of hidden layers of a DNN have also been used recently for speaker adaptive training [12, 30, 31], to better exploit training data over different tasks [32], or as an extension of multi-task learning for low resource ASR [33]. These approaches largely differ from the proposed CA-DNN as they do not perform a linear interpolation of the different class parameters.

5. EXPERIMENTS

We carried out experiments using the AURORA4 [24] speech corpus. AURORA4 is a noisy version of WSJ0 5k, which includes different types of noise. There are four evaluation sets, i.e., A (clean), B (six types of additive noises), C (clean with channel distortion), D (six types of additive noises with channel distortion). All experiments were performed using the multi-condition training data set of AURORA4. The training data set consists of 83 speakers and about 14 hours of data. Note that the training and testing conditions are relatively matched, i.e. the training data includes the same noise and channel conditions as the evaluation data but the SNRs of training data are 10-20 dB and those of evaluation data are 5-15 dB.

5.1. Settings

The baseline acoustic model consists of a DNN with 5 hidden layers each with 1024 units per layer and 3042 output units corresponding to the HMM states. We used sigmoid activation functions for all hidden layers. The speech features consists of 24 log mel filterbank coefficients appended with delta and acceleration coefficients. We employed 11 concatenated speech features as input to the DNN (792 dimensions in total). The speech features were processed with utterance level cepstral mean normalization, and further normalized using mean and variance normalization parameters calculated on the training data.

We trained the DNN using discriminative pre-training, and then fine-tuned it using the cross entropy criterion. For the fine tuning,

Table 1. WER for the evaluation set of AURORA4 using bigram and trigram language models. The best results are highlighted with bold font.

	bigram	trigram
DNN	15.2	11.9
DNN + i-vect	**14.3**	11.3
CA-DNN ($i = 1$)	16.0	12.4
CA-DNN-JT ($i = 1$, softmax)	14.7	11.3
CA-DNN-JT ($i = 1$, linear)	**14.3**	**11.2**

we used an initial learning rate of 0.3, a momentum of 0.9 and a batch size of 128. The learning rate was gradually decreased when the frame accuracy did not improve for a cross validation set.

In addition to the speech features, we used i-vectors as context features. The i-vectors were obtained by training a GMM universal background model (UBM) of 512 dimensions. We then extracted i-vectors of 80 dimensions. The i-vectors were finally processed with LDA to reduce their size to 25 components, using the speaker labels to train the LDA transformation. We used in this experiment *utterance level i-vectors* computed with Kaldi [34]. Note that since the speech data contains noise and channel mismatch, the i-vectors may also represent noise/channel characteristics in addition to speakers characteristics.

For CA-DNN, we tested two approaches to obtain the class weights. The first approach follows [20], where we used a GMM trained from the i-vectors to compute posteriors that are used as class weights. We used a GMM with 4 components. The second approach consists of the proposed CA-DNN-JT. For CA-DNN-JT, the inputs of the ancillary network are the utterance level i-vectors described above. The ancillary network consists of a single hidden layer with 25 hidden units and sigmoid activation functions. The output layer has 4 output units. We performed experiments with both softmax and linear output layers. The parameters of the CA-DNN and the ancillary network are jointly trained using the procedure described in Section 3. In this paper, we only investigated the factorization of a single layer.

For all experiments, we used both bi-gram and tri-gram language models for decoding. In both cases, we used a language model weight scale of 15 and a beam value of 400. The results are evaluated in terms of word error rate (WER) for the evaluation set.

5.2. Results

Table 1 shows the WER for a conventional DNN, DNN with i-vector auxiliary features, CA-DNN with class weights obtained from i-vector posteriors and the proposed CA-DNN-JT with softmax and linear output layers [1] . The results for CA-DNN were obtained when factorizing the first layer (i.e. $i = 1$). Note that our baseline is comparable with other results reported on the task with a similar configuration [22]. However, it is slightly worse than the state-of-the-art on this task [15] because we used discriminative pre-training and a network with 5 layers instead of 7 to speedup the experimental turnaround. Moreover, we do not use CMLLR feature adaptation because we focus here on fast adaptation and CMLLR requires more than one utterance to compute the feature transformation parameters.

From Table 1, we observe that using i-vector auxiliary features achieves about 5 % relative improvement. CA-DNN without joint training degraded performance in this case. However, the proposed CA-DNN-JT with linear output layer achieved a performance improvement comparable with using i-vectors as auxiliary features. The performance degraded when we used a softmax output layer, probably because of the vanishing gradient as discussed in subsection 3.2.2.

[1]All networks have the same number of hidden layers and hidden units. CA-DNNs have more parameters since they include a factorized hidden layer. Note that we confirmed in [20] that the performance improvement brought about by CA-DNN was not due to the increased number of parameters.

Table 2. WER for different position of the factorized layer using the trigram language model. The best results are highlighted with bold font.

Factorized layer, i	1	2	3	4	5
CA-DNN-JT (linear)	11.2	**10.7**	10.9	11.1	11.3

Table 3. WER for the evaluation set of AURORA4 using bigram and trigram language models for baseline DNN system, a system with i-vectors auxiliary input features and the proposed CA-DNN-JT. The best results are highlighted with bold font.

	A	B	C	D	Avg.
bigram					
DNN	6.1	10.2	11.0	22.4	15.2
DNN + i-vect	5.6	9.5	**9.6**	21.3	14.3
CA-DNN-JT ($i = 2$, linear)	**5.3**	**9.4**	9.9	**20.2**	**13.7**
trigram					
DNN	**3.7**	7.4	8.4	18.3	11.9
DNN + i-vect	**3.7**	7.1	**7.4**	17.4	11.3
CA-DNN-JT ($i = 2$, linear)	**3.7**	**6.6**	7.6	**16.4**	**10.7**

Table 2 shows the results for different factorized layers. We observe that better performance could be achieved when factorizing the second, third or fourth layer. When factorizing the second layer, we could achieve a 10% relative improvement over our DNN baseline.

Finally, Table 3 shows detailed results for the different test sets with bigram and trigram language models. The proposed CA-DNN-JT improves performance over the DNN baseline in all conditions and outperforms i-vector auxiliary features except for set C (clean with channel mismatch). CA-DNN-JT appears especially efficient in noisy conditions. Note that using i-vectors as auxiliary input features performs bias compensation of the first layer. This may explain why it performs better for compensating the channel mismatch of the set C. Since CA-DNN realizes adaptation of the weight matrices, both approaches could be complementary, and we will explore their combination in future work.

6. CONCLUSIONS

In this paper we proposed an extension of the recently proposed CA-DNN. We introduced an ancillary network to compute context class weights, and interconnected it with the CA-DNN to enable joint training. This allows obtaining context classes and class weights optimized for ASR. CA-DNN provides an alternative to conventional approaches for making DNN aware of the acoustic context by exploiting auxiliary information. For utterance-based unsupervised adaptation on the AURORA4 task, the proposed CA-DNN-JT achieved performance competitive with conventional approach that uses i-vectors as auxiliary input features.

In future work, we will explore other configurations and topologies for the CA-DNN and ancillary network to achieve better context class representation [35] and to reduce the number of model parameters [11, 31].

7. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[2] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 14–22, 2012.

[3] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system," in *Proc. of EUROSPEECH'95*, 1995, pp. 2171–2174.

[4] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *Proc. of SLT'12*, 2012, pp. 366–369.

[5] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. of ASRU'11*, 2011, pp. 24–29.

[6] T. Yoshioka, A. Ragni, and M. J. F. Gales, "Investigation of unsupervised adaptation of DNN acoustic models with filter bank input," in *Proc. of ICASSP'14*, 2014, pp. 6344–6348.

[7] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. de Mori, "Adaptation of hybrid ANN/HMM models using linear hidden transformations and conservative training," in *Proc. of ICASSP'06*, vol. 1, 2006, pp. 1189–1192.

[8] B. Li and K. C. Sim, "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems," in *Proc of INTERSPEECH'10*, 2010, pp. 526–529.

[9] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Proc. of ICASSP'13*, 2013, pp. 7893–7897.

[10] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Proc. of ICASSP'13*, 2013, pp. 7947–7951.

[11] 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 of ICASSP'14*, 2014, pp. 6359–6363.

[12] T. Ochiai, S. Matsuda, X. Lu, C. Hori, and S. Katagiri, "Speaker adaptive training using deep neural networks," in *Proc of ICASSP'14*, 2014, pp. 6349–6353.

[13] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proc of SLT'14*, 2014.

[14] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proc. ASRU'13*, 2013, pp. 55–59.

[15] M. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. of ICASSP'13*, 2013, pp. 7398–7402.

[16] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *Proc. of ICASSP'13*. IEEE, 2013, pp. 7942–7946.

[17] Y. Miao, H. Zhang, and F. Metze, "Towards speaker adaptive training of deep neural network acoustic models," in *Proc of INTERSPEECH'14*, 2014, pp. 2189–2193.

[18] J. Li, J.-T. Huang, and Y. Gong, "Factorized adaptation for deep neural network," in *Proc of ICASSP'14*, 2014, pp. 5537–5541.

[19] C. Yu, A. Ogawa, M. Delcroix, T. Yoshioka, T. Nakatani, and J. H. Hansen, "Robust i-vector extraction for neural network adaptation in noisy environment," in *Proc of INTERSPEECH'15*, 2015, pp. 2854–2857.

[20] M. Delcroix, K. Kinoshita, T. Hori, and T. Nakatani, "Context adaptive deep neural networks for fast acoustic model adaptation," in *Proc. of ICASSP'15*, 2015, pp. 4535–4539.

[21] M. Gales, "Cluster adaptive training of hidden Markov models," *IEEE Trans. Speech and Audio Process.*, vol. 8, no. 4, pp. 417–428, 2000.

[22] C. Wu and M. Gales, "Multi-basis adaptive neural network for rapid adaptation in speech recognition," in *Proc. of ICASSP'15*, 2015, pp. 4315–4319.

[23] T. Tan, Y. Qian, M. Yin, Y. Zhuang, and K. Yu, "Cluster adaptive training for deep neural network," in *Proc. of ICASSP'15*, 2015, pp. 4325–4329.

[24] N. Parihar and J. Picone, "DSR Front End LVCSR evaluation - AU/384/02," in *Aurora Working Group, European Telecommunications Standards Institute*, 2002.

[25] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[26] V. Tresp, "Committee machines," in *Handbook of Neural Network Signal Processing*, Y. Hu and J.-N. Hwang, Eds. Boca Raton, FL, USA: CRC Press, Inc., 2001, ch. 5.

[27] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2000.

[28] S. Masoudnia and R. Ebrahimpour, "Mixture of experts: a literature survey," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 275–293, 2014.

[29] D. Yu, X. Chen, and L. Deng, "Factorized deep neural networks for adaptive speech recognition," in *Proc. of IWSML'12*, 2012.

[30] T. Ochiai, S. Matsuda, H. Watanabe, X. Lu, C. Hori, and S. Katagiri, "Speaker adaptive training for deep neural networks embedding linear transformation networks," in *Proc. of ICASSP'15*, 2015, pp. 4605–4609.

[31] S. Li, X. Lu, Y. Akita, and T. Kawahara, "Ensemble speaker modeling using speaker adaptive training deep neural network for speaker adaptation," in *Proc of INTERSPEECH'15*, 2015, pp. 2892–2896.

[32] C. Liu, J. Li, and Y. Gong, "SVD-based universal dnn modeling for multiple scenarios," in *Proc of INTERSPEECH'15*, 2015, pp. 3244–3248.

[33] J. Cui, G. Saon, B. Ramabhadran, and B. Kingsbury, "A multi-region deep neural network model in speech recognition," in *Proc of INTERSPEECH'15*, 2015, pp. 3244–3248.

[34] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *Proc. of ASRU'11*, 2011.

[35] S. Kundu, G. Mantena, Y. Qian, T. Tan, M. Delcroix, K. C. Sim, and Y. K., "Joint acoustic factor learning for robust deep neural network based automatic speech recognition," in *ICASSP'16 (Submitted)*, 2016.