INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Context adaptive neural network for rapid adaptation of deep CNN based acoustic models

Marc Delcroix, Keisuke Kinoshita, Atsunori Ogawa, Takuya Yoshioka, Dung Tran, 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

Using auxiliary input features has been seen as one of the most effective ways to adapt deep neural network (DNN)-based acoustic models to speaker or environment. However, this approach has several limitations. It only performs compensation of the bias term of the hidden layer and therefore does not fully exploit the network capabilities. Moreover, it may not be well suited for certain types of architectures such as convolutional neural networks (CNNs) because the auxiliary features have different time-frequency structures from speech features. This paper resolves these problems by extending the recently proposed context adaptive DNN (CA-DNN) framework to CNN architectures. A CA-DNN is a DNN with one or several layers factorized in sub-layers associated with an acoustic context class representing speaker or environment. The output of the factorized layer is obtained as the weighted sum of the contributions of each sub-layer, weighted by acoustic context weights that are derived from auxiliary features such as i-vectors. Importantly, a CA-DNN can compensate both bias and weight matrices. In this paper, we investigate the use of CA-DNN for deep CNN-based architectures. We demonstrate consistent performance gains for utterance level rapid adaptation on the AURORA4 task over a strong network-in-network based deep CNN architecture.
Index Terms: Acoustic model adaptation, context adaptive network, auxiliary features, deep convolutional neural networks

1. Introduction

The emergence of deep neural network (DNN)-based acoustic models has generated an increased interest for neural network adaptation techniques to speakers or environments. Adaptation techniques include feature transformations [1–4], direct adaptation or transformation of all or part of the DNN parameters [5–12], and exploiting auxiliary features [13–20]. In particular, bias compensation can be easily implemented by augmenting the input of a network or a hidden layer with auxiliary features such as noise estimates or i-vectors [13, 14]. Exploiting auxiliary features is especially appealing because it does not require labels to achieve adaptation and it enables rapid adaptation since the auxiliary features can be computed using a small amount of speech data.

Most neural network adaptation techniques have been investigated for fully connected deep neural networks. However, recent ASR systems rely more and more on advanced architectures such as deep convolutional neural networks (CNNs) [21, 22], deep long short time memory (LSTM) networks [23] etc. In particular, deep CNN with network-in-network (NiN)-based acoustic models have been shown to be powerful for noise robust ASR tasks [24, 25]. In such architectures, several convolutional layers are stacked below fully connected layers. There have been some investigations for exploiting auxiliary features for CNN-based networks [17, 20, 21]. However, the auxiliary features cannot simply be added to the input of the convolutional layers, because auxiliary features such as i-vectors usually exhibit a different time-frequency structure than speech features. Therefore, auxiliary features have been used to compute a bias term directly added to the input features [17] or as auxiliary features to the input of the fully connected layers that are above the convolutional layers in a CNN architecture [20, 21].

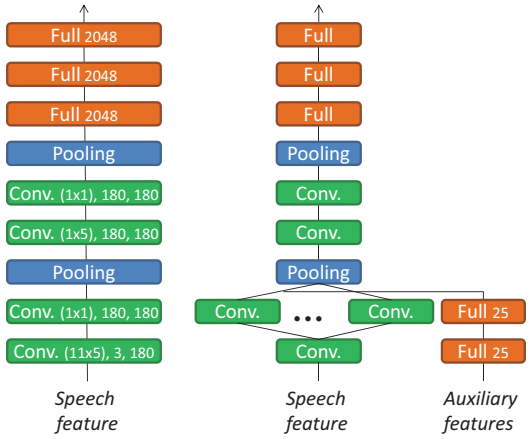
In this paper, we investigate the use of context adaptive deep neural networks (CA-DNN) for deep CNN-based acoustic models such as NiN. A CA-DNN is a neural network architecture we recently proposed [26, 27], which has one or several layers factorized in sub-layers. Each sub-layer is associated with an acoustic context class or cluster. The output of the factorized layer is obtained as the weighted sum of the contribution of each sub-layer, weighted by acoustic context class weights that are derived from auxiliary features. CA-DNN provides an alternative to inputting i-vectors to the input or hidden layers of the network. It has two potential advantages, (1) it can adapt both the bias terms and the weight matrices of the network, (2) it can be naturally applied to convolutional layers. Therefore it can be used to adapt the lower layers of the network that may carry more speaker related information [10, 28]. We demonstrate experimentally the potential of context adaptive CNN (CA-CNN) on the AURORA4 noisy speech recognition task for rapid utterance based speaker adaptation. The proposed CA-CNN achieves up to 6 % relative word error rate (WER) reduction over a strong NiN-based deep CNN baseline. To the best of our knowledge, this constitutes the first attempt to use context/cluster adaptive DNN to CNN architectures. In addition, we provide visualization of the context class weights that shows that the proposed CA-CNN automatically learns to differentiate between male and female speakers.

In the remainder of the paper, we first review the NiN-based deep CNN architecture in Section 2. We then elaborate on its extension to CA-CNN in Section 3. In Section 4, we discuss the relations of our method with previous works. We then present experimental results in Section 5. Finally, Section 6 concludes the paper and discusses potential future research directions.

2. Convolutional neural network and Network-in-Network

As an example of deep CNN acoustic model, we base our discussion on the NiN architecture [29] shown in Figure 1-(a). The NiN architecture discussed in this paper is similar to the one proposed in [24, 25], which has been shown to achieve high performance in noisy conditions. In the following subsections we briefly review the principle of a convolutional layer and the NiN architecture.

http://dx.doi.org/10.21437/Interspeech.2016-203



(a) NiN (b) CA-CNN (i=2)

Figure 1: Schematic diagram of (a) a NiN-based deep CNN network, (b) the corresponding CA-CNN with the second layer ($i = 2$) replaced with a context adaptive layer.

2.1. Convolutional layer

Figure 2-(a) is a schematic diagram of the i^{th} convolutional layer of a network. The input of the convolutional layer consists of time-frequency feature maps, i.e. for the first layer, it usually consists of three maps associated with static, Δ and $\Delta\Delta$ coefficients. The input feature maps are processed by a set of convolutional filters then passed through an activation function to generate the output feature maps as,

$$z_{t,f,m}^{(i)} = \sum_n \mathbf{w}_{n,m}^{(i)} * \mathbf{x}_{t,f,n}^{(i-1)} + b_m^{(i)}, \quad (1)$$

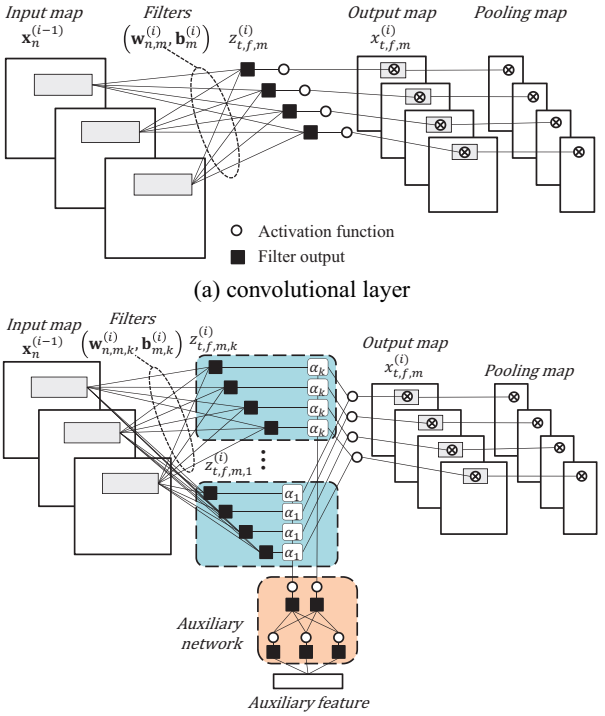
$$x_{t,f,m}^{(i)} = \sigma(z_{t,f,m}^{(i)}), \quad (2)$$

where $*$ is the convolution operator, $z_{t,f,m}^{(i)}$ is the output of the filter associated with the m^{th} output feature map for time index t and frequency index f, $\mathbf{x}_{t,f,n}^{(i-1)}$ is an input local patch centered around (t, f) of the n^{th} feature map, $\mathbf{w}_{n,m}^{(i)}$ and $b_m^{(i)}$ are the filter and bias associated with the n^{th} input and m^{th} output feature maps, and σ is an activation function. After the convolution operation, the resolution of the output feature map can be reduced using pooling. Here we use max-pooling, which is realized by taking the maximum over non-overlapping rectangular sub-regions of the feature map. Note that each convolutional layer is not always followed by a pooling layer.

The convolutional layer is characterized by four numbers $(P \times Q, N, M)$, the filter size $(P \times Q)$ and the number of input and output feature maps N, M, respectively. Note that the $P \times Q$ convolution operation over a $T \times F$ feature map results in a $(T - P + 1) \times (F - Q + 1)$ feature map, limiting therefore the number of convolution layers that can be used.

2.2. Network-in-Network

A deep CNN is obtained by stacking several convolutional layers. A NiN is a deep CNN architecture, where a 1×1 convolution layer is added between the standard convolution layer and the pooling operation. An example of a NiN architecture is shown in Figure 1-(a). The 1×1 convolution layer realizes a cross-feature map multi-layer perceptron (MLP) that enables to capture more complex non-linear structures with a very marginal increase in the number of parameters. The pooling operation is performed after the 1×1 convolutional layer.



(a) convolutional layer

(b) context adaptive convolutional layer

Figure 2: Schematic diagram of a (a) convolutional layer and (b) context adaptive convolutional layer.

The NiN architecture we used here consists of 4 convolutional layers, 2 pooling layers and 3 fully connected layers. We used 1×2 pooling layers, meaning that we reduce the dimension of the feature map in the frequency axis by a factor of 2 after pooling.

3. Context adaptive CNN

The proposed CA-CNN is derived from the NiN architecture by replacing one layer with a context adaptive layer. Figure 2-(b) is a detailed representation of a context adaptive convolutional layer. Following our prior work on CA-DNN [27], the context adaptive convolutional layer is realized by creating K sets of filters, where K is the number of acoustic context classes we consider. The convolution operations generate thus $M \times K$ features maps $z_{t,f,m,k}^{(i)}$ as,

$$z_{t,f,m,k}^{(i)} = \sum_n \mathbf{w}_{n,m,k}^{(i)} * \mathbf{x}_{t,f,n}^{(i-1)} + b_{m,k}^{(i)}, \quad (3)$$

where $\mathbf{w}_{n,m,k}^{(i)}$, $b_{m,k}^{(i)}$ are the filter and bias associated with the k^{th} acoustic context class. The output of the context adaptive layer is then obtained by the weighted sum of the feature maps, which are then processed by the activation function as,

$$x_{t,f,m}^{(i)} = \sigma(\sum_{k=1}^{K} \alpha_k z_{t,f,m,k}^{(i)}), \quad (4)$$

where α_k represents the context class weights. This operation reduces the number of output feature maps back to M maps.

Note that Eq. (4) can also be interpreted as a convolutional layer, which parameters are adapted to the acoustic context by

giving context class weights α_k i.e.,

$$\hat{\mathbf{w}}_{n,m}^{(i)} = \sum_{k=1}^{K} \alpha_k \mathbf{w}_{n,m,k}^{(i)}, \quad (5)$$

$$\hat{b}_{m}^{(i)} = \sum_{k=1}^{K} \alpha_k b_{m,k}^{(i)}, \quad (6)$$

where $\hat{\mathbf{w}}_{n,m}^{(i)}$ and $\hat{b}_{m}^{(i)}$ are the adapted filter and bias, respectively. Consequently, we can achieve rapid adaptation if the acoustic context weights α_k can be obtained with a small amount of adaptation data. Note that in contrast to conventional approaches using i-vectors as inputs, the proposed approach can also adapt the filter coefficients as shown in Eq. (5).

Building on top of our previous work on CA-DNN [27], we use an auxiliary network to compute the acoustic context weights α_k. This small auxiliary network transforms input auxiliary features such as i-vectors into the acoustic context class weights. All parameters of the main network and auxiliary network are jointly trained to minimize the cross entropy (CE). Consequently the acoustic context class weights and thus class definitions are optimized for recognition.

In this paper, we apply the context adaptive convolutional layer to a single layer of the baseline NiN architecture. Figure 1-(b) illustrates the architecture obtained when factorizing the second convolutional layer.

4. Relation to prior works

There have been a few related studies investigating the use of auxiliary features for adaptation of CNNs [17,20,21]. It is more challenging to incorporate auxiliary features to a CNN than to a DNN. Indeed, a CNN assumes that the input features obey time and frequency locality properties. Speech features obey this property but auxiliary features such as i-vectors may not. Consequently, the i-vectors should be incorporated into the convolutional layer by concatenating the feature with every localized frequency patch, making the system more complex [21]. An alternative consists of adding the auxiliary features to the upper fully connected layers [20,21], which may potentially limit the impact of adaptation as lower layers of a DNN are known to carry more speaker/environment related information than the upper layers [28].

In [17], i-vectors were also used for speaker adaptive training of a CNN by using them to compute a bias term that was directly added to the input features. This approach also transformed i-vectors using an auxiliary network that was jointly trained with the CNN. However, the auxiliary network outputs the bias term, where we use it to compute the acoustic class weights.

Finally, our proposed CA-CNN is also related to other works that proposed to factorize architectures for DNNs. For example, the network architecture is similar to that used for cluster adaptive training of DNNs [30,31], with the main difference originating from the class weight computation. However, these approaches have only been investigated for conventional fully connected DNNs so far. This paper constitutes to the best of our knowledge the first attempt to use factorized architectures for deep CNN adaptation.

5. Experiments

We carried out experiments using the AURORA4 [32] 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, which consists of 83 speakers and about 14 hours of data. The evaluation data consists of 8 speakers (5 males and 3 females) and 330 utterances. Note that the training and testing conditions are relatively matched for sets A and B, 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. The channel used in sets C and D is unseen during training.

5.1. Settings

Our baseline system consists of a NiN-based deep CNN architecture shown in Figure 1-(a). The output consists of 3042 output units corresponding to the HMM states. The speech features consists of 40 log mel coefficients appended with static, Δ and $\Delta\Delta$ coefficients. We employed 11 concatenated speech features as input to the DNN (1320 dimensions in total). The input features were organized into three 11×40 time-frequency feature maps, one for static, Δ and $\Delta\Delta$. The speech features were processed with utterance level mean normalization, and further normalized using mean and variance normalization parameters calculated on the training data. We used parametrized ReLU (PReLU) [33] activation functions for all hidden layers.

We used i-vectors as auxiliary features. They were obtained by training a GMM universal background model (UBM) of 512 dimensions. We then extracted i-vectors of 80 dimensions, which 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]. In addition to the baseline NiN, we also compared our proposed method with adding i-vectors to the input of the first fully connected layer.

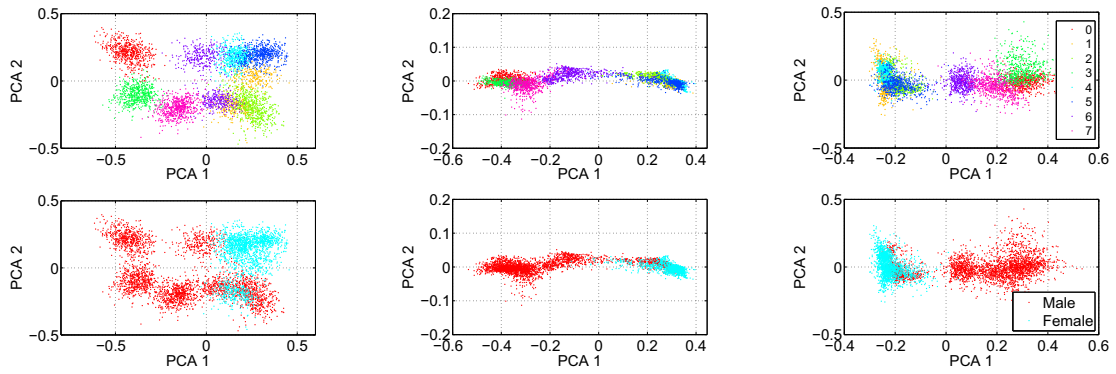
The proposed CA-CNN architecture is as discussed in Section 2 and shown in Figure 1-(b). The auxiliary network consists of 2 hidden layers each with 25 hidden units with PReLU activations and an output layer with linear activation as in our previous work [27]. We used 2 acoustic context classes as it performed slightly better than using 4 or more classes.

All models were randomly initialized, and directly fine-tuned using the cross entropy criterion (i.e. we did not use any pre-training). We used an initial learning rate of 0.01, 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. The learning was stopped after 20 epochs. We used dropout regularization for all fully connected layers. We used our own decoder to perform the experiments and employed the Chainer library for training the networks [35]

For all experiments, we used a bigram language models for decoding unless otherwise stated. The results were evaluated in terms of word error rate (WER) for the evaluation set.

5.2. Results

Table 1 shows the WER for the three network configurations. The results for CA-CNN are shown for different factorized hidden layers. The baseline NiN is a strong baseline for the task. We observed that adding i-vectors provided small performance gains for clean and noisy conditions (sets A and B). The proposed CA-CNN could improve performance in most cases for test sets A, B and D. Better performance could be achieved when factorizing the convolutional layers ($i = 1, 2, 3, 4$) compared to factorizing the fully connected layers ($i = 5, 6$). Note that we also tested different configurations of the auxiliary network and increased number of acoustic context classes



(a) i-vectors (b) context class weights (2 classes) (c) context class weights (4 classes)

Figure 3: Plots of the PCA of i-vectors and acoustic class weights for 2 and 4 classes. Each dot corresponds to one utterance of the evaluation set. The colors represent the speaker labels, and gender labels for the upper and lower figures, respectively. (Best in color)

Table 1: WERs for the evaluation set of AURORA4 for NiN-based deep CNN, NiN + i-vectors and the proposed CA-CNN for different factorized layers (the asterisks indicate convolutional layers). The best results are highlighted with bold fonts.

	A	B	C	D	Avg.
NiN	5.3	8.3	**7.3**	17.0	11.74
NiN + i-vect	**4.9**	8.1	7.6	16.9	11.61
CA-CNN ($i = 1$)*	**4.9**	8.2	7.7	16.4	11.44
CA-CNN ($i = 2$)*	5.2	8.1	7.4	16.4	11.43
CA-CNN ($i = 3$)*	**4.9**	8.1	**7.3**	16.2	11.28
CA-CNN ($i = 4$)*	**4.9**	**7.9**	7.4	**16.1**	**11.18**
CA-CNN ($i = 5$)	5.2	8.0	8.2	17.0	11.66
CA-CNN ($i = 6$)	5.3	8.3	8.1	16.4	11.53

but could not obtain significant performance improvements.

Both NiN + i-vectors and CA-CNN could not improve performance on test set C that corresponds to unseen channel mismatch. This seems to demonstrate a potential limitation of auxiliary feature based approaches when dealing with acoustic context completely unseen during training. Performance improvement could probably be obtained if we would perform channel compensation on the input features. Note that in this experiment, since we are focusing on rapid adaptation, we did not use CMLLR to adapt the features as it would require more than one utterance to compute reliable CMLLR transforms.

5.3. Discussion

Figure 3 plots the i-vectors for the evaluation set and the derived acoustic class weights obtained with the CA-CNN when using 2 and 4 classes. We performed principal component analysis (PCA) to reduce the features to 2 dimensions to simplify the visualization.

As shown in Figure 3-(a) the i-vectors encompass speaker information. The acoustic context class weights shown in Figures 3-(b) and (c) were obtained by processing the i-vectors with the auxiliary network. It illustrates what acoustic context classes are learned by the network. From the top figures the speaker information seems less distinguishable in the acoustic class weights than for the i-vectors. In contrast, the auxiliary network emphasizes the distinction between male and female speakers as revealed by the two main clusters in the bottom figures. It is interesting that the auxiliary network learns this distinction although both the main NiN network and the auxiliary networks were randomly initialized. Note that one of the male speakers is included in the "female" cluster, which may naturally occur since the acoustic context class weights were computed based only on acoustic features.

When using 4 acoustic context classes, the auxiliary network seems to be able to distinguish even better between genders. However, we did not observe better performance in the 4 class cases (i.e. WER of 11.20 %). This may be due to the small amount of training data that prevents proper learning of a larger number of acoustic context class parameters. We may thus need to reduce the amount of parameters associated with each acoustic context class to further improve performance.

Table 2: WERs for the evaluation set of AURORA4 using bigram and trigram language models for the stronger baseline system with dynamic feature learning (DFL).

	bigram	trigram
NiN + DFL	11.58	8.74
CA-CNN + DFL ($i = 4$)	10.78	8.23

5.4. Results with dynamic feature learning

Finally we have tested the proposed CA-CNN with a stronger baseline that includes 2 additional convolutional layers with filters of 5×1 placed in the bottom of the network. This configuration enables learning the dynamic features [25]. In this case, the input feature consists of 19 frames of 40 dimension log mel features without Δ and $\Delta\Delta$. Table 2 shows the results with the dynamic feature learning (DFL) baseline for the bigram and trigram language models. The proposed CA-CNN can provide significant gains of up to 6% relative improvement even with this stronger baseline.

6. Conclusion

In this paper we have investigated the use of context adaptive DNNs to realize rapid adaptation of deep CNNs. We have demonstrated that the proposed CA-CNN was effective to realize adaptation of NiN based deep CNN architectures and achieved up to 6 % relative improvement over a strong baseline. In addition, we have demonstrated experimentally that the acoustic context classes learned by the CA-CNN were related to gender information. These results appear promising and suggest that the CA-CNN network can learn meaningful classes.

In future work we will investigate if this behavior can still be observed when using different auxiliary features such as noise estimates to realize environmental adaptation. We also plan to test the proposed CA-CNN for larger tasks, where we may potentially learn more complex acoustic context class definitions.

7. References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

[12] Y. Miao and F. Metze, "On speaker adaptation of long short-term memory recurrent neural networks," in *Proc. of INTERSPEECH'15*, 2015, pp. 1101–1105.

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

[14] 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.

[15] 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*, 2013, pp. 7942–7946.

[16] 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.

[17] Y. Miao, L. Jiang, H. Zhang, and F. Metze, "Improvements to speaker adaptive training of deep neural networks," in *Proc. of SLT'14*, 2014, pp. 165–170.

[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] O. Abdel-Hamid and H. Jiang, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition," in *Proc. of INTERSPEECH'13*, 2013, pp. 1248–1252.

[21] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A.-R. Mohamed, G. Dahl, and B. Ramabhadran, "Deep convolutional neural networks for large-scale speech tasks," *Elsevier, Neural Networks*, vol. 64, pp. 39 – 48, 2015.

[22] G. Saon, H. J. Kuo, S. J. Rennie, and M. Picheny, "The IBM 2015 English conversational telephone speech recognition system," *arXiv*, 2015. [Online]. Available: http://arxiv.org/abs/1505.05899

[23] T. N. Sainath, O. Vinyals, A. W. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *ICASSP'2015*, 2015, pp. 4580–4584.

[24] T. Yoshioka, N. Ito, M. Delcroix, A. Ogawa, K. Kinoshita, M. Fujimoto, C. Yu, W. J. Fabian, M. Espi, T. Higuchi, S. Araki, and T. Nakatani, "The NTT CHiME-3 system: advances in speech enhancement and recognition for mobile multi-microphone devices," in *Proc. of ASRU'15*, 2015, pp. 436–443.

[25] T. Yoshioka, K. Ohnishi, F. Fang, and T. Nakatani, "Noise robust speech recognition using recent developments in neural networks for computer vision," in *Proc. of ICASSP'16*, 2016.

[26] 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.

[27] M. Delcroix, K. Kinoshita, C. Yu, A. Ogawa, T. Yoshioka, and T. Nakatani, "Context adaptive deep neural networks for fast acoustic model adaptation in noisy conditions," in *Proc. of ICASSP'16*, 2016.

[28] A. Mohamed, G. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," in *Proc. of ICASSP'12,*, 2012, pp. 4273–4276.

[29] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013. [Online]. Available: http://arxiv.org/abs/1312.4400

[30] 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.

[31] 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.

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

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," *Proc. Int. conf. Computer Vision*, 2015.

[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. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in *Proc. of Workshop on Machine Learning Systems in NIPS*, 2015.