CUMULATIVE MOVING AVERAGED BOTTLENECK SPEAKER VECTORS FOR ONLINE SPEAKER ADAPTATION OF CNN-BASED ACOUSTIC MODELS

Tsubasa Ochiai[1,2], *Marc Delcroix*[1], *Keisuke Kinoshita*[1], *Atsunori Ogawa*[1]
Taichi Asami[3], *Shigeru Katagiri*[2], *Tomohiro Nakatani*[1]

[1] NTT Communication Science Laboratories, NTT corporation, Kyoto, Japan
[2] Graduate School of Engineering, Doshisha University, Kyoto, Japan
[3] NTT Media Intelligence Laboratories, NTT Corporation, Yokosuka, Japan

ABSTRACT

Adapting acoustic models to speakers have shown to greatly improve performance for many tasks. Among the adaptation approaches, exploiting auxiliary features characterizing speakers or environments has received great attention because they allow rapid adaptation, i.e. adaptation with limited amount of speech data such as a single utterance. However, the auxiliary features are usually computed in batch mode, which causes some inevitable latency. In this paper we explore an extension of the auxiliary feature-based adaptation to online processing. We employ auxiliary features obtained from bottleneck speaker vectors and extend their computation to online processing using cumulative moving averaging. We test our proposed approach for deep CNN-based acoustic models, using context adaptive networks to exploit the auxiliary features. Experimental results on the CHiME-3 task demonstrate that the proposed approach can realize online speaker adaptation.

Index Terms— Acoustic model adaptation, Online adaptation, Bottleneck speaker vectors, Convolutional neural network

1. INTRODUCTION

Adapting an acoustic model to speakers is known to improve automatic speech recognition (ASR) performance in many tasks. The widespread use of deep learning-based acoustic models has led to the development of various acoustic model adaptation techniques, which can be classified into three main categories, i.e. feature transformation [1–3], acoustic model parameter adaptation or retraining with error-backpropagation [4–7], and exploiting auxiliary input features [8–13]. The first two approaches require adaptation data with labels or transcriptions. Consequently, two ASR decoding passes are needed for unsupervised adaptation. Moreover, it usually requires a relatively large amount of adaptation data to become effective.

An auxiliary input feature approach consists of training a deep neural network (DNN)-based acoustic model with acoustic features and auxiliary features. Typically, i-vectors or bottleneck speaker vectors are used for speaker adaptation. By doing so, the network learns to adapt its parameters given the auxiliary information about the speaker. This approach presents the advantage that the auxiliary features can be obtained without labels, making adaptation possible with a single ASR decoding pass. Moreover, auxiliary features such as i-vectors or bottleneck speaker vectors can be extracted from a limited amount of speech data, making utterance-level adaptation possible. However, current appoaches for extracting the auxiliary features assume utterance-level (or speaker-level) batch processing to compute the auxiliary features. Therefore, it is impossible to start the decoding procedure of the speech recognition system until the end of the utterance. This inevitably creates some latency before the system can output the recognition results.

In this paper, we propose to extend auxiliary feature-based speaker adaptation to online processing. Online speaker adaptation means that adaptation is performed by exploiting only past speech samples, which can thus reduce drastically the latency of the ASR system. To realize online speaker adaptation, we investigate online extension of bottleneck speaker vector computation. Conventional approaches for bottleneck speaker vectors train a DNN with a bottleneck layer to predict speaker labels. The prediction is performed on a frame-by-frame basis and the bottleneck speaker vectors are obtained by averaging over the frames of a single utterance, which generates a single vector per utterance [12]. This can be naturally extended to online processing by using moving averaging. The resulting bottleneck speaker vectors will then change frame-by-frame.

We investigate this approach with deep convolutional neural network (CNN)-based acoustic models. For CNNs, simply adding the auxiliary features to the input of the convolutional layer is not possible because the auxiliary features present a different time-frequency structure than the input acoustic features. We have recently proposed context adaptive CNN (CA-CNN) [14] to exploit auxiliary features within the CNN. A CA-CNN factorizes one of its convolutional layers into sublayers. The output of the factorized layer is obtained as the summation of the contribution of the different sublayers weighted by the interpolation coefficients derived from the auxiliary features. We investigate online bottleneck speaker vectors with CA-CNN for speaker adaptation on the CHiME-3 task. Our experimental results confirm that online speaker adaptation is possible and that adaptation may become effective after 3 seconds or less of input speech data.

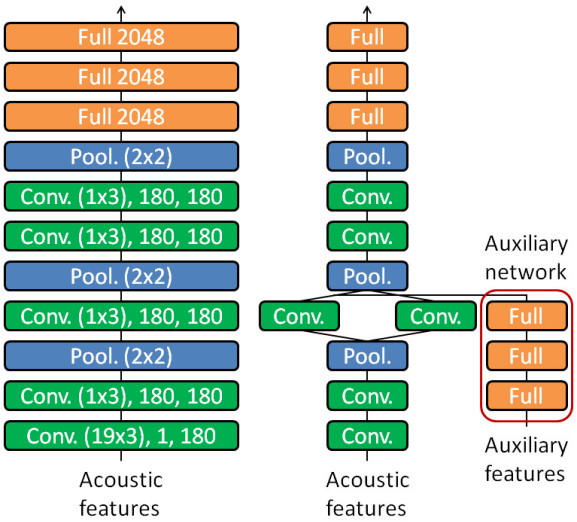
The remainder of this paper is as follows. In Section 2, we review auxiliary feature-based adaptation and conventional approaches for extracting utterance-level i-vectors and utterance averaged bottleneck speaker vectors. Section 3 describes our proposed online bottleneck speaker vector extraction, and we discuss its relation with prior works in Section 4. Finally, we present experimental results in Section 5 and conclude the paper in Section 6.

2. CONVENTIONAL AUXILIARY FEATURE-BASED ADAPTATION

2.1. Overview

Auxiliary input feature approach, consists of training a DNN-based acoustic model with acoustic features and auxiliary features, which represent the context information such as speaker or environment. Typically, utterance-level i-vectors or utterance averaged bottleneck

This work was done while Tsubasa Ochiai was an intern at NTT.



(a) Baseline CNN (b) CA-CNN

Fig. 1. Schematic diagram of (a) baseline CNN, (b) the corresponding CA-CNN.

speaker vectors are used for the speaker adaptation [9, 12].

The conventional approach consists of giving the auxiliary features as an additional input of a fully connected layer of the DNN. This implies compensation of the bias term of the corresponding hidden layer based on the auxiliary features.

However, for the CNNs, simply adding the auxiliary features to the input of the convolutional layer is difficult because the auxiliary features present a different time-frequency structure than the input acoustic features. We have recently proposed CA-CNN to exploit auxiliary features within a convolutional layer. This is realized by factorizing the convolutional layers, where the auxiliary features are used as the input to an auxiliary network that is used to calculate interpolation coefficients of each factorized sub-layers.

The structure of the CA-CNN, which we used in the experiment of this paper, is illustrated in Figure. 1. Because of the multilayer structure, the factorized layer can be allocated to any of the convolutional layers. As an example, in the figure, we allocate the factorized layer to the third convolutional layer.

In the following for discussion simplicity, we assume that the factorized layer is allocated to the i-th convolutional layer. The parameters of the factorized layer can thus be obtained as follows:

$$\hat{\mathbf{w}}^i_{n,m} = \sum_{k=1}^{K} \alpha_k \mathbf{w}^i_{n,m,k}, \tag{1}$$

$$\hat{b}^i_m = \sum_{k=1}^{K} \alpha_k b^i_{m,k}, \tag{2}$$

where, $\hat{\mathbf{w}}^i_{n,m}$ and $\hat{b}^i_m$ are the adapted filter and bias of the i-th convolutional layer associated with the n-th input and m-th output feature maps, $\mathbf{w}^i_{n,m,k}$ and $b^i_{m,k}$ are the basis filter and bias of the k-th factorized sublayer, K is the number of the factorized sublayers, and α_k is the interpolation coefficient of k-th factorized sublayer.

$\alpha_k (k = 1, 2, \cdots, K)$ is calculated as the outputs of the small-size feed forward network, which is referred to as the auxiliary network. The auxiliary network and the factorized sublayers are trained jointly with the other part of the network, i.e., main CNN-based network.

2.2. Utterance-level Auxiliary Features

2.2.1. Utterance-level i-vectors

I-vectors are often used as auxiliary features for speaker adaptation. The i-vector is a low-dimensional vector representation, which represents the speaker information.

The i-vector approach models the speaker variability based on a Gaussian mixture model (GMM). The speaker dependent GMM's supervector $\mathbf{M}$ is represented as follows,

$$\mathbf{M} = \mathbf{m} + \mathbf{T}\mathbf{i}, \tag{3}$$

where $\mathbf{m}$ is the GMM's supervector of the speaker independent GMM, which is referred to as universal background model (UBM), $\mathbf{T}$ is a low-rank matrix which spans the total-variability subspace, and $\mathbf{i}$ is the low-dimensional i-vector representation. The parameters of the i-vector extractor is trained based on the Expectation Maximization (EM) algorithm.

Given a sequence of the acoustic feature vectors $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{T^u}\}$, the i-vector representation for utterance u, which is referred to as $\mathbf{i}^u$, is calculated as follows,

$$\mathbf{i}^u = \left(\mathbf{I} + \mathbf{T}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{N}(u)\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{F}(u), \tag{4}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{\Sigma}$ is the diagonal covariance matrix, which represents the residual variability not captured by $\mathbf{T}$, and T^u is the total frame length of the utterance u. $\mathbf{N}(u)$ and $\mathbf{F}(u)$ are the zero-order and first-order statistics, which are calculated on the given utterance. Further details of the i-vector algorithm can be found in [15].

2.2.2. Utterance Averaged Bottleneck Speaker Vectors

Utterance averaged bottleneck speaker vectors have been used as an alternative to i-vectors for speaker adaptation. To create the utterance averaged bottleneck speaker vectors, we first extract the framewise bottleneck speaker vectors. This is done using a bottleneck DNN (BN-DNN), which is trained to classify the training speaker's indices and the additional silence label. In the BN-DNN, the last hidden layer is a bottleneck layer, whose dimension is basically smaller than one of the other hidden layers.

After the training procedure, the BN-DNN is expected to extract speaker information in the bottleneck layer. Therefore, the linear output of the bottleneck layer is used as the framewise bottleneck speaker vector. Since the framewise bottleneck speaker vectors are very noisy, the vectors are usually averaged over a single utterance. The utterance averaged bottleneck speaker vector for utterance u, which is referred to as $\tilde{\mathbf{a}}^u$, is given as follows:

$$\tilde{\mathbf{a}}^u = \frac{1}{T^u}\sum_{\tau=1}^{T^u} \mathbf{a}^u_\tau, \tag{5}$$

where $\mathbf{a}^u_\tau$ is the framewise bottleneck speaker vector for the τ-th input frame of utterance u.

2.3. Latency problem of conventional approaches

In the conventional auxiliary feature-based adaptation approach, the acoustic model can be adapted by using the speech data of a single utterance. Compared to the other approaches, it can be said that it is a rapid adaptation approach.

However, to calculate the utterance-level auxiliary features, it is necessary to wait until the end of the utterance. Namely, even in the conventional auxiliary feature-based approach, it is impossible to

start the decoding procedure of the speech recognition system until the end of the utterance. It leads to the latency before the speech recognition system can output the recognition results.

Therefore, for the situation where the response speed of the speech recognition system is important, e.g., smart phone applications, it is not suitable to use the conventional auxiliary feature-based approach using the utterance-level auxiliary features.

To decrease the latency introduced by the computation of the utterance-level auxiliary features, we extend the auxiliary feature-based approach to perform the online processing, which means that it is possible to start the decoding procedure from the beginning of the utterance.

3. PROPOSED ONLINE BOTTLENECK SPEAKER VECTORS

To perform the online style decoding, we have to create the auxiliary features using only the speech frames prior to the current time.

Framewise bottleneck speaker vectors can be used as the online auxiliary features because they are calculated by using only the current frame. However, such framewise bottleneck speaker vectors are unstable to represent the speaker information, because the length of the speech frame for the current time point is too short (about 0.2 seconds in the experiment of this paper) to capture the speaker characteristics precisely. Therefore, like the conventional utterance averaged bottleneck speaker vectors, it is assumed that it is important to apply some kind of the averaging technique.

In this paper, we propose the moving average-based bottleneck speaker vectors to realize the online streaming adaptation. There are some variations of the moving average methods. In this paper, we simply use the cumulative moving average, which is the average of all features up until the current time point t. Our choice of cumulative moving average is governed by the fact that the online bottleneck speaker vectors converge to the conventional utterance averaged bottleneck speaker vectors at the end of the utterance. This makes comparison with the conventional utterance averaged method easier.

Our proposed online bottleneck speaker vector for utterance u at the time t, which is referred to as $\tilde{\mathbf{a}}_t^u$, is given as follows:

$$\tilde{\mathbf{a}}_t^u = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{a}_\tau^u \tag{6}$$

$$= \frac{(t-1)\tilde{\mathbf{a}}_{t-1}^u + \mathbf{a}_t^u}{t} \tag{7}$$

Note that when the online bottleneck speaker vectors are calculated in the form of Eq. (7), the computational cost is constant in each time step. For the online adaptation scenario, such a constant computational cost is a preferable property from the viewpoint of the response speed of the speech recognition system.

4. RELATION TO PRIOR WORKS

Our work is related to previous works on speaker adaptation with bottleneck speaker vectors such as [11,12], except that we extend the concept to online processing and use it in the context of CA-CNN.

There are several related works on online i-vectors computation. [16] proposed to employ the online i-vectors for speaker diarization, where the i-vectors were extracted for every set of 10 input acoustic features.

Online i-vectors have also been used in the context of ASR in [17] but with a different purpose. In [17], instead of using all frames of the utterance to calculate the statistics, the statistics of the online i-vectors are calculated using only past frames at each time point, and the i-vectors are sequentially generated by using the statistics. Such online i-vectors were used during training to increase the variety of the i-vectors for training samples. During the testing stage, the i-vectors were extracted offline using a sufficient amount of speech data (60$\sim$ seconds). To the best of our knowledge, there has been no report of results with online i-vectors used during the testing stage. In principle, the online i-vectors could also be used for online adaptation scenario in a similar way as our proposed online bottleneck speaker vectors. We will compare our approach with the online i-vectors.

5. EXPERIMENTS

5.1. Settings

5.1.1. Speech corpus and acoustic feature representation

We tested our proposed scheme on the CHiME-3 noisy speech corpus [18]. The CHiME-3 is a noisy speech recognition task, consisting of speech recorded with a six-microphone tablet device in four environments, i.e., cafe (CAF), street junction (STR), public transport (BUS) and pedestrian area (PED).

The training set consists of 1600 real utterances of 4 speakers and 7138 simulated utterances of 83 speakers. The development (dev) set consisted of 1640 real utterances and 1640 simulated utterances of 4 speakers. The evaluation (eval) set consisted of 1320 real utterances and 1320 simulated utterances of the other 4 speakers. In this experiment, we used the audio data from one microphone, which is located in the center of the tablet device. Therefore, the total length of the training data was about 18 hours.

The acoustic feature vector consists of 80 log mel filterbank coefficients. It was normalized so that its mean and variance became 0 and 1, respectively. Then 19 concatenated acoustic feature vectors (1520 dimensions) were used as input to the acoustic models.

5.1.2. Evaluated configurations

The architecture of our baseline CNN and the corresponding CA-CNN are shown in Figure 1. The output consists of 5976 output units corresponding to the HMM states. We used sigmoid activations for all layers and used the dropout regularization [19] for the fully connected layers.

In the CA-CNN, the auxiliary network consists of 2 hidden layers with sigmoid activations, each of which has 50 hidden units, and an output layer with linear activation. The output of the auxiliary network consists of 2 output units, which corresponds to 2 acoustic context classes (sublayers). We chose 2 acoustic context classes as it performed slightly better than using 4 or more classes. The dimension of all auxiliary features (i-vectors or bottleneck feature vectors) was set to 50. Therefore, the auxiliary network has 50 input and 2 output units.

All models were randomly initialized, and directly optimized using the cross entropy criterion. We used an initial learning rate of 0.08, a momentum of 0.9 and a batch size of 128. At each training epoch, the learning rate was halved if the frame accuracy did not improve for the dev set. The training procedure was stopped after 25 epochs. For online adaptation experiments we used the acoustic models trained with utterance-level auxiliary features because in preliminary experiments it performed better than when using online auxiliary features during training.

In this experiment, we compare 5 types of auxiliary features, utterance-level and online i-vectors, and utterance-level, framewise

Table 1. Experimental results (word error rate [%])

System	dev	eval
baseline CNN	11.38	19.48
i-vector (utt)	10.66	19.75
i-vector (online)	10.85	20.78
bottleneck (utt)	10.56	18.41
bottleneck (frame)	12.32	19.96
bottleneck (online)	10.99	18.88

and online bottleneck speaker vectors. The utterance-level and online i-vectors were computed using the Kaldi toolkit [20]. For the bottleneck speaker feature extraction, we use a fully connected network with 3 hidden layers with sigmoid activations. The input consists of the same input acoustic features used for the main CNN-based network. The network was trained to estimate the 83 training speaker labels added with an additional class for silence frames. The last hidden layer consisted of bottleneck layer consisting of 50 units. All the other hidden layers had 1024 units. Note that the computation of the online i-vectors and online bottleneck speaker vectors assures that the feature obtained at the end of the utterance corresponds to the corresponding utterance-level feature.

For all experiments, we used a trigram language model for decoding. The results were evaluated in terms of word error rate (WER) for the real data of the dev and eval sets.

5.2. Results

Table 1 shows the WERs for the dev and eval sets for our baseline deep CNN system, and for the CA-CNN systems using utterance-level and online i-vectors, and bottleneck speaker vectors. For bottleneck speaker vectors, we compared results for the framewise vectors (not averaged), utterance-level averaging and the proposed online extraction (cumulative moving averaging). We obtained these results based on the configuration of the CA-CNN (the factorized layer position) that performed best on the dev set. Note that there is a great discrepancy in the nature of the dev and eval data sets, which is clearly observed from the large performance gap of the baseline system on the two data sets. This discrepancy can be explained from the speaking style differences between the speakers of the dev and eval sets [18].

From Table 1, we confirmed that using bottleneck speaker vectors for auxiliary features is effective for adaptation. The utterance-level bottleneck speaker vectors performed slightly worse than i-vectors on the dev set, but outperformed i-vectors on the eval set. Using framewise bottleneck speaker vectors degrades performance compared to the baseline, which confirms that it is challenging to capture speaker information from a very short context and proves that the averaging operation is essential. The proposed online bottleneck speaker vectors could improve performance compared to the baseline but performed slightly worse than the utterance-level ones. Such loss in performance is expected given the poor performance of framewise bottleneck speaker vectors.

In our experiments, we confirmed that the systems using i-vectors and online i-vectors could improve performance over the baseline on the dev set but performed significantly worse on the eval set. This is probably due to difference in the distributions of speech features between the dev and eval speakers caused speaking style differences [18]. We will include in our future works further investigations of online i-vectors on different data sets.

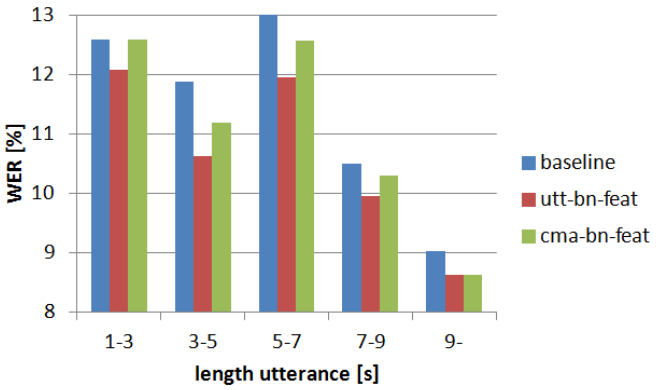


Fig. 2. Relationship between total length of utterances and recognition performance (word error rate [%]).

5.3. Discussions

We investigated the effect of adaptation as a function of the length of the utterances to informally evaluate the speed of adaptation. Figure 2 plots the WERs for the dev set as a function of the length of the utterance in seconds for the baseline, utterance-level and online bottleneck speaker vectors. The percentage of utterances in each time interval to the total number of utterances in dev set is about 9.5%, 25.6%, 30.9%, 24.5%, and 9.6%, respectively.

Utterance-level bottleneck speaker vectors improve performance in all conditions. This suggests that utterance-level bottleneck speaker vectors computed with less than 3 seconds of speech can already capture speaker characteristics needed for adaptation.

The proposed online bottleneck speaker vectors outperform the baseline except for utterances shorter than 3 seconds, and perform slightly worse than utterance-level ones. This results suggested that online adaptation can start becoming effective at least after 3 seconds. Further investigations are needed to get more precise evaluation of the amount of data needed for online adaptation.

Moreover, we suspect that the initial silent portions of the utterance may bias the estimation of the online bottleneck speaker vectors, which may explains why online adaptation does not improve over the baseline for short utterances. We will investigate approaches to remove such initial bias in future works.

6. CONCLUSION

This paper proposed an online extension of auxiliary feature-based adaptation. We extended the bottleneck speaker vectors approach to online processing by using cumulative moving averaging, and demonstrated performance competitive with utterance-level bottleneck speaker vectors for CNN-based adaptation using CA-CNN.

In this paper, we evaluated our proposed scheme only for a CNN-based network structure. However, our proposed scheme can be applied to other neural network structures such as deep neural networks (DNNs) and recurrent neural networks (RNNs). Evaluating the effect of our proposed method for such network structures will be part of our future research topic.

In addition, future research directions will include testing other variations of moving averaging (e.g. moving window average) and joint training of the bottleneck feature extractor and the network of the CA-CNN in a similar way as [21]. We are also interested in evaluating the proposed online adaptation with lecture or meeting speech data where several consecutive utterances are spoken by the same speaker.

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

[3] T. Yoshioka, A. Ragni, and M. J. F. Gales, "Investigation of unsupervised adaptation of DNN acoustic models with filter bank input," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6344–6348.

[4] 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 *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7893–7897.

[5] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7947–7951.

[6] 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 *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6359–6363.

[7] T. Ochiai, S. Matsuda, X. Lu, C. Hori, and S. Katagiri, "Speaker adaptive training using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6349–6353.

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

[9] A. Senior and I. Lopez-Moreno, "Improving dnn speaker independence with i-vector inputs," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 225–229.

[10] V. Gupta, P. Kenny, P. Ouellet, and T. Stafylakis, "I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6334–6338.

[11] H. Huang and K. C. Sim, "An investigation of augmenting speaker representations to improve speaker normalisation for dnn-based speech recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4610–4613.

[12] S. Kundu, G. Mantena, Y. Qian, T. Tan, M. Delcroix, K. C. Sim, and Yu K., "Joint acoustic factor learning for robust deep neural network based automatic speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5025–5029.

[13] 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.

[14] M. Delcroix, K. Kinoshita, A. Ogawa, T. Yoshioka, D. Tran, and T. Nakatani, "Context adaptive neural network for rapid adaptation of deep cnn based acoustic models," in *Proc. of INTERSPEECH'16*, 2016, pp. 1573–1577.

[15] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[16] S. Madikeri, I. Himawan, P. Motlicek, and M. Ferras, "Integrating online i-vector extractor with information bottleneck based speaker diarization system," Tech. Rep., Idiap, 2015.

[17] V. Peddinti, G. Chen, D. Povey, and S. Khudanpur, "Reverberation robust acoustic modeling using i-vectors with time delay neural networks," in *Proc. of INTERSPEECH'15*, 2015, pp. 2440–2444.

[18] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third "chime" speech separation and recognition challenge: Dataset, task and baselines," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 504–511.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," in *Journal of Machine Learning Research*, 2014, pp. 1929–1958.

[20] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[21] K. Vesely, S. Watanabe, K. Zmolikova, M. Karafiat, L. Burget, and J.H. Cernocky, "Sequence summarizing neural network for speaker adaptation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5315–5319.