Variable-Component Deep Neural Network for Robust Speech Recognition

Rui Zhao[1], *Jinyu Li*[2], and *Yifan Gong*[2]
[1]Microsoft Search Technology Center Asia, Beijing, China
[2]Microsoft Corporation, One Microsoft Way, Redmond, WA 98052
{ruzhao; jinyli; ygong}@microsoft.com

10.21437/Interspeech.2014-154

Abstract

In this paper, we propose variable-component DNN (VCDNN) to improve the robustness of context-dependent deep neural network hidden Markov model (CD-DNN-HMM). This method is inspired by the idea from variable-parameter HMM (VPHMM) in which the variation of model parameters are modeled as a set of polynomial functions of environmental signal-to-noise ratio (SNR), and during the testing, the model parameters are recomputed according to the estimated testing SNR. In VCDNN, we refine two types of DNN components: (1) weighting matrix and bias (2) the output of each layer. Experimental results on Aurora4 task show VCDNN achieved 6.53% and 5.92% relative word error rate reduction (WERR) over the standard DNN for the two methods, respectively. Under unseen SNR conditions, VCDNN gave even better result (8.46% relative WERR for the DNN varying matrix and bias, 7.08% relative WERR for the DNN varying layer output). Moreover, VCDNN with 1024 units per hidden layer beats the standard DNN with 2048 units per hidden layer with 3.22% WERR and a half computational/memory cost reduction, showing superior ability to produce sharper and more compact models.

Index Terms: variable component, variable parameter, variable output, robust speech recognition

1. Introduction

Recently, the context-dependent deep neural network hidden Markov model (CD-DNN-HMM) [1][2][3][4][5][6] has shown its superiority over the traditional Gaussian mixture model (GMM)-HMM model in automatic speech recognition and has been widely used in real speech recognition productions. Naturally, improving the robustness of CD-DNN-HMM further becomes the next goal of the research [7]. Although deep neural network (DNN) has been shown to be noise robust even without any explicit noise compensation technique [8][7], there is still room for improvement as shown in work [9][10][11]. Until now, most of these work focus on the robust feature for DNN.

In this paper, we propose a model-based noise-robust method called variable-component DNN (VCDNN). This method is inspired by the idea from the variable-parameter HMM (VPHMM) method [12][13][14][15][16]. For both DNN-HMM and GMM-HMM systems, one widely used model-based noise-robust method is to include noisy speech under various conditions into the training data, which is called multi-condition training [17][18]. Although experimental results have shown that multi-condition training consistently gives high recognition accuracies [19], it has some problems: (1) the various training environments are modeled with fixed set of parameters, leading to "flat" distributions. So for the testing speech produced in a particular environment, such "flat" model would not be the optimal matched model. (2) It is difficult to collect training data to cover all possible types of environments, so the performance on unseen noisy environments remains unpredictable. VPHMM was proposed to solve these problems.

In VPHMM, HMM parameters, such as state emission parameters (GMM mean and variance) or adaptation matrices, are modeled as a polynomial function of a continuous environment-dependent variable (e.g. SNR). At the recognition time, a set of GMM means and variances specific to the given value of the environment variable is instantiated and used for recognition. Even if the testing environment is not seen in the training, the estimated GMM parameters can still work well because the change of means and variances in terms of the environment variable can be predicted by polynomials.

In our proposed VCDNN method, we want to have any component in the DNN to be modeled as a set of polynomial functions of an environment variable. In this study, we investigate two types of variation: variable-parameter DNN (VPDNN) in which the weight matrix and bias are variable dependent, and variable-output DNN (VODNN) in which the output of each hidden layer is variable dependent. As in VPHMM, the variable-dependent components are computed online for the environment condition detected in the testing data using their associated polynomial functions during recognition. Experimental results on the Aurora4 task show VPDNN and VODNN can have 6.52% and 5.92% relative WERR over standard DNN trained with multi-conditional training method, respectively. VPDNN and VODNN are also shown to work even better under the unseen SNR conditions, where 8.46% and 7.08% relative WERR over standard DNN are obtained respectively. Besides, VPDNN with 1024 units per hidden layer could beat the standard DNN with 2048 units per hidden layer with 3.22% relative WERR and a half computational/memory cost reduction.

This paper is organized as follows. In Section 2 and 3, we briefly introduce CD-DNN-HMM and VPHMM, respectively. Then, in Section 4, the proposed VCDNN will be described in detail. In Section 5, the experimental results on Aurora4 will be presented. And the conclusion will be given in Section 6.

2. CD-DNN-HMM

In the framework of CD-DNN-HMM, the log likelihood of tied context dependent HMM states (will be called senone in the rest of this paper) is calculated using DNN instead of GMM in the conventional GMM-HMM systems. DNN can be considered as a multi-layer perceptron (MLP) consisting of one input layer, one output layer and many hidden layers. Each node in the output layer represents one senone.

Usually, a sigmoid function is chosen as the activation function for hidden layers of DNN and the output of the l-th hidden layer o^l is given by:

$$o^l = f_{sigm}(u^l) \quad (1)$$

$$u^l = (W^l)^T o^{l-1} + b^l \quad (2)$$

where o^{l-1} is the input of the l-th layer, W^l and b^l are the weighting matrix and bias of the l-th layer, respectively. $f_{sigm}(x) = 1/(1+e^x)$.

The activation function of the output layer (layer L) is a softmax function

$$o_k^L = \frac{exp(u_k^L)}{\sum_{i=1}^{S} exp(u_i^L)} \quad (3)$$

Hence, the senone posterior probability $P(s_k|x)$ is:

$$P(s_k|x) = \frac{exp(u_k^L)}{\sum_{i=1}^{S} exp(u_i^L)} \quad (4)$$

where x is the input feature vector of DNN, s_k is the senone responding to unit k of top layer, and S is the total number of senones. The first layer's input $o^0 = x$. The senone emission likelihood of HMM $p(x|s)$ is then calculated according to

$$p(x|s) = P(s|x) \cdot p(x)/P(s) \quad (5)$$

$P(s)$ is the prior probability of senone s. $p(x)$ is independent of s and can be ignored during HMM decoding.

In DNN training, the commonly used optimization criterion is the cross-entropy between the posterior distribution represented by the reference labels $\hat{P}(s|x)$ and the predicted distribution $P(s|x)$. The objective function is:

$$F_{CE} = -\sum_{i=1}^{S} \hat{P}(s_i|x) \log\big(P(s_i|x)\big) \quad (6)$$

The reference label is typically decided based on the forced-alignment results:

$$\hat{P}(s_i|x) = \begin{cases} 1 & if\ x\ is\ aligned\ to\ senone\ s_i \\ 0 & else \end{cases} \quad (7)$$

Then equation (6) is simplified as:

$$F_{CE} = -\log\big(P(s'|x)\big) \quad (8)$$

where s' is the reference senone for the speech input x.

With the above objective function, a DNN can be trained with the method introduced in [1], which consists of unsupervised pre-training and supervised fine-tuning. The algorithm used in the fine-tuning stage is error back propagation, where the weighting matrix W and bias b of layer l are updated with:

$$\hat{W}^l = W^l + \alpha o^{l-1} (e^l)^{\mathrm{T}} \quad (9)$$

$$\hat{b}^l = b^l + \alpha e^l \quad (10)$$

α is the learning rate. o^{l-1} and e^l are the input and error vector of layer l respectively. e^l is calculated by propagating the error from its upper layer:

$$e_i^l = \left[\sum_{k=1}^{N_{l+1}} w_{ik}^{l+1} e_k^{l+1}\right] f'_{sigm}\big(u_i^l\big) \quad (11)$$

w_{ik}^{l+1} is the element of weighting matrix W^{l+1} in i-th row and k-th column for layer $l+1$, and e_k^{l+1} is the k-th element of error vector e^{l+1} for layer $l+1$. N_{l+1} is the units number in layer $l+1$. $f'_{sigm}\big(u_i^l\big)$ is the derivative of sigmoid function. The error of the top layer (i.e. output layer) is the derivative of the objective function defined in equation (8).

$$e_s^L = -\frac{\partial F_{CE}}{\partial u_s^L} = (\delta_{ss'} - o_s^L) \quad (12)$$

$\delta_{ss'}$ is the Kronecker delta function.

3. VPHMM

In traditional GMM-HMM systems, the speech distribution under different environment is modeled by the same set of parameters (Gaussian mean and variance). The variation of these parameters caused by environment (such as SNR) changes has been studied in [12], and the result shows the distribution of the speech feature value is a continuous function of SNR. Hence traditional GMM-HMM is imperfect because it doesn't model the acoustic environment (e.g. SNR) changes.

To solve this problem, it is better to change the model parameters according to the environment. This is the motivation of VPHMM which models GMM parameters as a polynomial function of SNR, i.e., the Gaussian component m is modeled as $N(y;\ \mu(m,v), \sum(m,v))$. $\mu(m,v)$ and $\sum(m,v)$ are polynomial functions of environment variable v. For example, $\mu(m,v)$ can be denoted by

$$\mu(m,v) = \sum_{j=0}^{J} c_j(m) v^j \quad (13)$$

where $c_j(m)$ is a vector with the same dimension as input feature vector and corresponds to the j-th order environment variable. The choice of a polynomial function is based on its good approximation property to continuous functions, its simple derivation operations, and the fact that the change of means and variances in terms of the environment is smooth and can be modeled by low order polynomials.

In the training of VPHMM, $c_j(m)$ (and other parameters) can be estimated based on the maximum likelihood criterion with the EM algorithm. In the testing stage, the Gaussian mean and variance are calculated with the estimated SNR value of the testing speech. Even if the testing SNR is not seen in the training, the polynomial function can help to calculate appropriate model parameters, so VPHMM can work well in unseen environments.

4. Variable-Component DNN

The basic idea in variable-component DNN (VCDNN) is similar to VPHMM: to refine the DNN components by modeling their variation against environment changes, which is not explicitly taken into consideration in a standard DNN. In this study, we specifically work on two types of components of DNN: (a) weighting matrix and bias, (b) the output of each layer. To make it clear, we will call VCDNN on weighting matrix and bias as VPDNN, and VCDNN on the output of each layer as VODNN in the rest of this paper.

4.1. VPDNN

In VPDNN, the weighting matrix W and bias b of layer l is modeled as a function of environment variable v:

$$W^l = f_w^l(v)\ ,\ b^l = f_b^l(v) \quad 0 < l \le L \quad (14)$$

Here, we use a polynomial function for both f_w^l and f_b^l based on its advantages and effectiveness shown in VPHMM. SNR is selected as the environment variable. So we have:

$$W^l = \sum_{j=0}^{J} H_j^l v^j \quad 0 < l \le L \quad (15)$$

$$b^l = \sum_{j=0}^{J} p_j^l v^j \quad 0 < l \le L \quad (16)$$

J is the polynomial function order. H_j^l is a matrix with the same dimensions as W^l and p_j^l is a vector with the same

dimension as b^l. The flowchart of one layer in VPDNN is show in figure 1.

In the training of VPDNN, we need to learn H_j^l and p_j^l instead of W^l and b^l in standard DNN. From equation (15) and (16) we can see that if we set *J*=0, VPDNN is equivalent to standard DNN, so we don't need to learn H_j^l and p_j^l from the scratch. We can update them based on a standard DNN in the fine-tuning stage, and their initial values are:

$$H_0^l = W_{sta}^l \quad p_0^l = b_{sta}^l \tag{17}$$

$$H_j^l = 0 \quad p_j^l = 0 \quad j > 1 \tag{18}$$

W_{sta}^l and b_{sta}^l are weighting matrix and bias of the layer l in standard DNN.

Combining equation (15) (16) and the error back propagation algorithm introduced in Section 2, we can get the updating formulas for H_j^l and p_j^l:

$$\widehat{H}_j^l = H_j^l + \alpha o^{l-1} (e^l)^{\mathrm{T}} v^j \tag{19}$$

$$\hat{p}_j^l = p_j^l + \alpha e^l v^j \tag{20}$$

In the recognition stage, the weighting matrix W and bias b of each layer are instantiated according to (15) (16) with the estimated SNR of the testing data. Then the senone posterior can be calculated in the same way as in standard DNN.

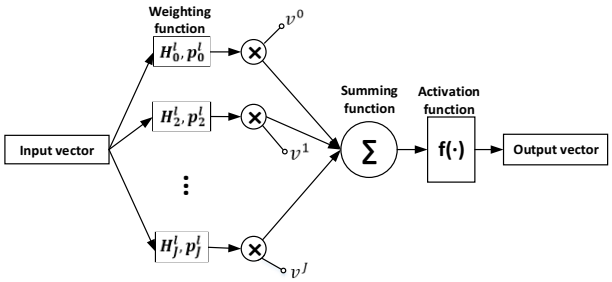

Figure 1. *Flowchart of one layer in VPDNN*

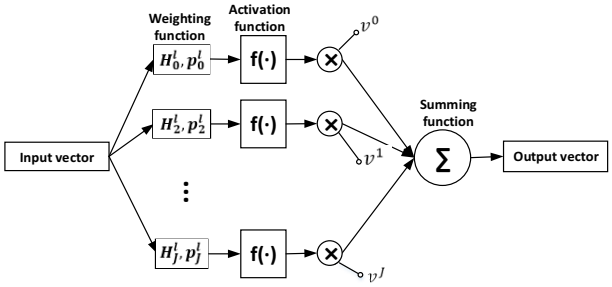

Figure 2. *Flowchart of one layer in VODNN*

4.2. VODNN

In VODNN, we assume the output of each hidden layer could be described by a polynomial function of environment variable v.

$$o^l = \sum_{j=0}^{J} f_{sigm}(u_j^l) v^j \qquad 0 < l < L \tag{21}$$

where

$$u_j^l = (H_j^l)^T o^{l-1} + p_j^l \qquad 0 < l < L \tag{22}$$

The framework of one layer in VODNN is shown in figure 2. As in VPDNN, H_j^l and p_j^l are updated based on standard DNN with the same initial values given in (17) (18). Similarly, the updating formulas could be obtained by combining (21) (22) and error back propagation algorithm:

$$\widehat{H}_j^l = H_j^l + \alpha o^{l-1} (e_j^l)^{\mathrm{T}} v^j \tag{23}$$

$$\hat{p}_j^l = p_j^l + \alpha e_j^l v^j \tag{24}$$

where

$$e_{i(j)}^l = \left[\sum_{n=0}^{J} \sum_{k=1}^{N_{l+1}} h_{ik(n)}^{l+1} e_{k(n)}^{l+1}\right] f'_{sigm}(u_{ij}^l) \tag{25}$$

$e_{i(j)}^l$ is the i-th element of error vector e_j^l for layer l, and $h_{ik(n)}^{l+1}$ is the element of matrix H_n^{l+1} in i-th row and k-th column for layer $l+1$.

In the recognition stage of VODNN, the output of each hidden layer is calculated according to (21) with the estimated SNR of the testing data. And the output of top layer, i.e. the senone posterior, is calculated according to equation (4) and (2) with the environment independent parameter W^L and b^L.

One thing that needs to be mentioned is that SNR v should be normalized for both VPDNN and VODNN because its numerical value range is too big compared with the DNN components. In this paper, we use the sigmoid function for the purpose of SNR normalization. It not only narrows the numerical value range but also makes the impact of very high SNRs similar. This would be reasonable since, for example, 40dB and 60dB SNR won't make obvious difference to speech recognition. This also applies to the very low SNR cases.

5. Experimental Results

The proposed methods are evaluated with Aurora 4 [20], a noise-robust medium-vocabulary task based on Wall Street Journal corpus (WSJ0). Aurora 4 has two training sets: clean and multi-condition. Each of them consists of 7138 utterances (about 14 hours of speech data). For the multi-condition training set, half of the data was recorded with a Sennheiser microphone and the other was with a secondary microphone. Besides, 6 types of noises (car, babble, restaurant, street, airport, and train) were added with SNRs from 10 to 20dB. The subset recorded with the Sennheiser microphone was called as channel wv1 data and the other part as channel wv2 data.

The test set contains 14 sub sets. 2 of them are clean and the other 12 are noisy. The noisy test sets were recorded with the same types of microphone as in multi-condition training set. Also, the same 6 types of noise as in multi-condition training set were added with SNRs between 5 and 15 dB.

The acoustic feature of baseline CD-DNN-HMM system is 24-dimensional log Mel filter-bank features plus their first- and second-order derivative features, totally 72 dimensions. The dimension of the DNN input layer is 792, formed from a context window of 11 frames. Its output layer contains 1209 units, which means there are 1209 senones in the HMM system. The DNN has 5 hidden layers with 2048 units in each layer.

In the experiments, we first examined the VCDNN's performance in terms of the order of polynomial. Both standard DNN and VCDNN are trained with the wv1 data from multi-condition training set. The test data are clean and 6 noisy wv1 sub sets. The results are given in Table 1 which shows the first-order VPDNN and VODNN achieved 6.53% and 5.92% relative word error rate reduction (WERR) over the standard DNN, respectively. However the second-order and third-order VCDNN did not show obvious gain compared with the first order one. This indicates that the first-order polynomial is good enough to model the variation caused by SNR changes within the DNN framework. Therefore, the first-order polynomial will be used in the following experiments. Given VPDNN is a little better than VODNN, in the following, we will only discuss VPDNN in detail.

Table 2 shows the breakdown results for different noise conditions and SNRs of the first order VPDNN. It can be seen that VPDNN works well in most noise environments. Besides, it gets even better result (8.47% relative WERR) in unseen SNR conditions (from 5dB to 10dB) compared with the seen conditions (>10dB). This indicates that DNN has a strong power to model the various environments it has seen, but for the unseen environments, there is more room for improvement. The similar result is also observed for VODNN (7.08% relative WERR for 5dB < SNR < 10dB, 4.26% relative WERR for SNR > 10dB).

Table 1. *The performance of VCDNN in terms of the order of polynomial*

WER(%)	VPDNN	VODNN
0 order (standard DNN)	10.26	10.26
1st order	9.59	9.65
2nd order	9.58	9.63
3rd order	9.58	9.62

Table 2. *Breakdown results for first-order VPDNN*

	5dB-10dB		> 10dB	
WER(%)	standard DNN	VPDNN	standard DNN	VPDNN
Clean	-	-	6.00	5.30
Street	16.19	14.67	8.89	9.00
Babble	13.46	11.72	7.63	7.32
Airport	12.72	11.44	8.20	8.08
Train	15.96	14.53	8.60	8.49
Car	6.13	6.10	5.67	5.67
restaurant	18.94	17.89	9.12	8.67
Average	13.85	12.68	7.52	7.23
Relative WERR(%)		8.47		3.79

Table 3. *Comparison of standard DNN and first-order VPDNN with different sizes*

WER(%)	2048 units / hidden layer	1024 units / hidden layer
standard DNN	10.26	10.50
VPDNN	9.59	9.93

At last, we examined the VCDNN's performance with less number of parameters using VPDNN. We evaluated VPDNN with 1024 units for each hidden layer to compare with the standard DNN with 2048 units per hidden layer. The results are given in Table 3. Evaluated with all the test sets with wv1 data, we can see that the first-order VPDNN with 1024 units per hidden layer achieves 3.22% relative WERR compared with the standard DNN with 2048 units per hidden layer, while the computational and memory costs are reduced by half. This will benefit the application scenario such as device dictation that only limited computational resource is available [21].

6. Conclusions and Future Works

In this paper, we have proposed a noise-robust method named VCDNN for CD-DNN-HMM speech recognition systems. In this method, the DNN components are modeled as a polynomial function of the speech SNR value, and during recognition, DNN components are instantiated according to the estimated SNR of the testing speech. We tried two kinds of implementation: VPDNN and VODNN. In VPDNN, the weighting matrix and bias of each layer are modeled as variables of SNR value, while in VODNN the output of each hidden layer are modeled as the variables of SNR.

Experimental results on the Aurora4 task show the first-order VPDNN and VODNN yield 6.52% and 5.92% relative WERR from the standard DNN trained with multi-conditional training method, respectively. They achieved better WERR from the standard DNN under unseen SNR conditions than under the seen SNR conditions. This indicates that DNN has a strong power to model the various environments it has observed, but for the unseen environments, there is more room for improvement. With the polynomial function, VCDNN can very well predict the DNN components used for unseen condition. Therefore, VCDNN can generalize very well for unseen environments. Moreover, the first-order VCDNN with 1024 units per layer could get 3.22% relative WERR and a half computational/memory cost reduction over the standard DNN with 2048 units per layer, showing superior ability to produce sharper and more compact models.

Even with the first-order SNR variable, VPDNN and VODNN proposed in this paper doubled the number of parameters from the standard DNN. The impact of SNR to DNN should be in a low dimension space. Therefore, we should be able to use only limited number of parameters to handle it. One way is to use the SNR variable as a factor in the input or output layer [8][22] so that only very limited number of DNN weights are connected with the SNR variable. Also, we may combine the SNR variable with other factors to do factorized training [22].We are working on these directions and will report results later.

ACKNOWLEDGEMENT

We would like to thank Dr. Mike Seltzer in Microsoft for providing the SNR table of Aurora 4 utterances.

7. References

[1] D. Yu, L. Deng, and G. Dahl, "Roles of pretraining and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition," in *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[2] 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. Workshop on Automatic Speech Recognition and Understanding*, pp. 30–35, 2011.

[3] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[4] N. Jaitly, P. Nguyen, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition", in *Proc. Interspeech*, 2012.

[5] 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 Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

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

[7] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach, "An overview of noise-robust automatic speech recognition, in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014.

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

[9] B. Li and K. C. Sim, "Noise adaptive front-end normalization based on vector Taylor series for deep neural networks in robust speech recognition," in *Proc. ICASSP*, pp. 7408–7412, 2013.

[10] B. Li, Y. Tsao, and K. C. Sim, "An investigation of spectral restoration algorithms for deep neural networks based noise robust speech recognition," in *Proc. Interspeech*, pp. 3002–3006, 2013.

[11] M. Delcroix, Y. Kubo, T. Nakatani, and A. Nakamura, "Is speech enhancement pre-processing still relevant when using deep neural networks for acoustic modeling," in *Proc. Interspeech*, pp. 2992–2996, 2013.

[12] X. Cui and Y. Gong, "A study of variable-parameter Gaussian mixture hidden Markov modeling for noisy speech recognition," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 15, no. 4, pp. 1366–1376, 2007.

[13] D. Yu, L. Deng, Y. Gong, and A. Acero. "A novel framework and training algorithm for variable-parameter hidden Markov models," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 17, no. 7, 1348-1360, 2009.

[14] N. Cheng, X. Liu, and L. Wang, "Generalized variable parameter HMMs for noise robust speech recognition," in *Proc. Interspeech*, pp. 481–484, 2011.

[15] M. Radenen and T. Artieres, "Contextual hidden Markov models," in *Proc. ICASSP*, pp. 2113–2116, 2012.

[16] Y. Li, X. Liu, and L. Wang, "Feature space generalized variable parameter HMMs for noise robust recognition," in *Proc. Interspeech*, pp. 2968–2972, 2013.

[17] R. Lippmann, E. Martin, and D. Paul, "Multi-style training for robust isolated-word speech recognition," in *Proc. IEEE Int. Conf. Acoustic Speech Signal Processing*, pp. 705–708, 1987.

[18] M. Blanchet, J. Boudy and P. Lockwood, "Environment adaptation for speech recognition in noise," in *Proc. EUSIPCO*, 1992, pp. 391–394.

[19] Y. M. Cheng et al., "A robust front-end algorithm for distributed speech recognition," in *Proc. EUR. Conf. Speech Communication and Technology*, 2001

[20] N. Parihar and J. Picone, "Aurora working group: DSR front end LVCSR evaluation AU/384/02," *Tech. Rep.,* Institute for Signal and Information Processing, Mississippi State Univ., 2002.

[21] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria," in *Proc. Interspeech*, 2014.

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