Learning Small-Size DNN with Output-Distribution-Based Criteria

Jinyu Li[1], *Rui Zhao*[2], *Jui-Ting Huang*[1], *and Yifan Gong*[1]

[1]Microsoft Corporation, One Microsoft Way, Redmond, WA 98052
[2]Microsoft Search Technology Center Asia, Beijing, China

{jinyli; ruzhao; jthuang; ygong}@microsoft.com

Abstract

Deep neural network (DNN) obtains significant accuracy improvements on many speech recognition tasks and its power comes from the deep and wide network structure with a very large number of parameters. It becomes challenging when we deploy DNN on devices which have limited computational and storage resources. The common practice is to train a DNN with a small number of hidden nodes and a small senone set using the standard training process, leading to significant accuracy loss. In this study, we propose to better address these issues by utilizing the DNN output distribution. To learn a DNN with small number of hidden nodes, we minimize the Kullback–Leibler divergence between the output distributions of the small-size DNN and a standard large-size DNN by utilizing a large number of un-transcribed data. For better senone set generation, we cluster the senones in the large set into a small one by directly relating the clustering process to DNN parameters, as opposed to decoupling the senone generation and DNN training process in the standard training. Evaluated on a short message dictation task, the proposed two methods get 5.08% and 1.33% relative word error rate reduction from the standard training method, respectively.

Index Terms: DNN, device, output distribution, model compression, learning

1. Introduction

Context-dependent deep neural network hidden Markov model (CD-DNN-HMM) has been shown, by many groups [1][2][3][4][5][6], to outperform the conventional Gaussian mixture model (GMM)-HMMs on many automatic speech recognition (ASR) tasks. The outstanding performance comes with much higher runtime cost because DNNs use much more parameters than the traditional GMM-HMM systems. Several technologies have been proposed to improve the runtime of CD-DNN-HMM so that it can be deployed on servers with high accuracy. Low rank matrices are used at the output layers [7] and all layers [8] to reduce the number of DNN parameters and CPU cost. In [9], several engineering methods such as 8-bit quantization for SSE evaluation are employed, and the frame skipping or prediction technology is used in [10].

Given widely used mobile devices such as smart phones and wearable devices, the industry has strong interests to have DNN systems on devices. However, even with the technologies mentioned above, the large computational cost is still very challenging to the deployment of CD-DNN-HMM on devices due to the limited processing power of devices. A common way to fit CD-DNN-HMM on devices is to reduce the DNN model size by reducing the number of nodes in hidden layers and the number of senone targets in the output layer [11]. Although the DNN model size is reduced, significant increase in word error rate is also observed [11].

In this paper, we explore a better way to reduce the DNN model size with less accuracy loss than the straight-forward method in [11]. To that end, we utilize the property of DNN output distribution by minimizing the Kullback–Leibler (KL) divergence between the output distributions of the small-size DNN and the large-size DNN. We also use the equivalence between the log-linear model and the Gaussian model to cluster the large senone set into a small one. This paper is organized as follows. In Section 2, we review the basic DNN training procedures. We describe the proposed methods in Section 3 and evaluate them in Section 4. Finally, we summarize our study and conclude the paper in Section 5.

2. CD-DNN-HMM

A deep neural network (DNN) can be considered as a conventional multi-layer perceptron (MLP) with many hidden layers (thus deep). The three major components contributing to the excellent performance of CD-DNN-HMM are: modeling senones directly even though there might be thousands of senones; using DNNs instead of shallow MLPs; and using a long context window of frames as the input.

Denote the input and output of the DNN as x and o, the input vector at layer *l* as v^l ($v^0 = x$), the weight matrix as W^l, and bias vector as a^l. Then for a DNN with L hidden layers, the output of the *l*-th hidden layer is

$$v^{l+1} = \sigma\left(z(v^l)\right), \qquad 0 \le l < L \tag{1}$$

where $z(v^l) = W^l v^l + a^l$ and $\sigma(x) = 1/(1+e^x)$ is the sigmoid function applied element-wise. The posterior probability is

$$P(o = s|x) = softmax(z(v^L)), \tag{2}$$

where s belongs to the set of senones (tied triphone states).

We compute the HMM's state emission probability density function $p(x|o=s)$ by converting the state posterior probability $P(o=s|x)$ to

$$p(x|o=s) = \frac{P(o=s|x)}{P(o=s)} \cdot p(x), \tag{3}$$

where $P(o=s)$ is the prior probability of state s, and $p(x)$ is independent of state and can be dropped during evaluation.

In our implementation, the CD-DNN-HMM inherits the model structure, including the phone set, the HMM topology, and tying of context-dependent states, directly from the CD-GMM-HMM system. The senone labels used for training the DNNs are extracted from the forced alignment generated using the CD-GMM-HMM. The training criterion is to minimize cross entropy which is reduced to minimize the negative log likelihood because every frame has only one target label s_t:

$$-\sum_t log\big(P(s_t|x_t)\big). \tag{4}$$

The DNN parameters are optimized with back propagation using stochastic gradient descent. In the following, we refer the DNN training process described in this section as the standard training process.

3. Small-Size DNN Learning

In this section, we study whether there is a better way to reduce the DNN model size with less accuracy loss than the straight-forward method in [11]. We will address this problem in two areas: 1) reducing the number of nodes in every hidden layer 2) reducing the number of senones in the output layer.

3.1 Learning DNN with a small number of hidden nodes

The accuracy gap is large between DNNs with large and small numbers of hidden nodes [11] if we train them with the standard training method in Section 2. To reduce the gap, our idea of learning DNN with small numbers of hidden nodes is motivated by the model compression concept [12] which trains a compact model to approximate the function learned by a large model. The concept was extended in [13] to learn a shallow neural network (SNN) with one single hidden layer to approximate the performance of a DNN by minimizing the difference of $z(v^L)$ values from a SNN and a DNN on the TIMIT database. To mimic a deep architecture, they train a SNN with greater or equal amounts of parameters. In [13], the number of parameters in the SNN ranges from 12 million to 180 million, whereas the DNN only has 12 million parameters. Different from their work, our learning task is to boost the performance of a small-size DNN by learning from a DNN with larger capacity (numbers of hidden nodes). One important property of such function learning is that no label is required during training. This enables us to explore the use of additional un-transcribed data to learn a better approximation of complex functions from a large-size DNN.

In this paper, we propose to directly minimize the KL divergence between the output distribution of the small-size DNN and the large-size DNN by leveraging large amounts of un-transcribed data to get a better small-size DNN than using the standard training method with only transcribed data.

We denote the posterior distribution of the large-size and small-size DNNs as $P_L(s|x)$ and $P_S(s|x)$, respectively. The KL divergence between these two distributions is

$$\sum_t \sum_{i=1}^{N} P_L(s_i|x_t) log\left(\frac{P_L(s_i|x_t)}{P_S(s_i|x_t)}\right) \qquad (5)$$

where N is the total number of senones.

To learn a small-size DNN that approximates the given large-size DNN, only the parameters of the small-size DNN needs to be optimized. Minimizing the above KL divergence is equivalent to minimizing the cross entropy

$$-\sum_t \sum_{i=1}^{N} P_L(s_i|x_t) log P_S(s_i|x_t) \qquad (6)$$

because $P_L(s_i|x_t) log P_L(s_i|x_t)$ has no impact on the small-size DNN parameter optimization. The proposed small-size DNN training criterion in Eq. (6) is a general form of the standard DNN training criterion in Eq. (4) where for every frame only one dimension of $P_L(s_i|x_t)$ equals to 1 and the others equal to 0. In contrast, in Eq. (6) every dimension of $P_L(s_i|x_t)$ has its non-zero (although may be very small) value.

The training steps of a small-size DNN guided by a large-size DNN are summarized as

1. Use the standard DNN training method to train a large-size DNN.
2. Use unsupervised pre-training to initialize a small-size DNN.
3. Then use a much larger amount of *un-transcribed* data to train the small-size DNN with the following steps.
   a. For each mini-batch, do forward propagation of both large-size and small-size DNNs to calculate $P_L(s_i|x_t)$ and $P_S(s_i|x_t)$.
   b. For that mini-batch, calculate the error signal of Eq. (6), and then do back propagation for the small-size DNN.
   c. Repeat Step 3.a & 3.b until convergence.

With the same amount of transcribed data, we don't expect the proposed method to outperform the standard training method. The advantage of the proposed method is that training of the small-size DNN (a student) doesn't need any transcription because its supervised signal $P_L(s|x)$ is obtained simply by passing training data through the large-size DNN (a teacher). Without the need for transcriptions, the small-size DNN trained based on optimizing Eq. (6) can use much more training data than trained with Eq. (4) for standard DNN training. With more training data to cover the feature space, we hope to learn a better small-size DNN. This training criterion is particularly useful for the industry scenario, where the amount of un-transcribed data is much larger than the amount of transcribed data due to the deployment feed-back loop. In the meantime, the high-performance large-scale DNN can be trained regardless of the memory/computation limitation for deployment on devices.

In contrast to directly using the continuous outputs of the teacher DNN with Eq. (6), an alternative way is to use an approximation to 1-of-K representation for $P_L(s_i|x_t)$ by assigning 1 to the senone class with the largest posterior and 0 to the others and then use Eq. (4) to train the small-size DNN. We believe this is not optimal because our goal is to let the student and teacher DNNs have as close as possible continuous outputs in Eq. (5).

3.2 Learning DNN with a small senone set

We also want to generate a small senone set with a better accuracy than using the standard senone generation method which splits the decision tree by maximizing the increase of likelihood evaluated on single Gaussians and forms the final senone set with the leaf nodes in the decision tree [14].

The potential problem with using the standard likelihood-based decision tree splitting process to obtain a senone set for DNN modeling is that the senone set is determined by single Gaussian distributions for MFCC or PLP feature, and hence the process to generate the senone set and the process to train a DNN are not consistent. Our proposed method partially addresses this concern by first generating a large senone set with the standard decision-tree splitting method and then merging similar senones using some criterion related to the DNN. In other words, we aim to perform clustering of a large set of senones based on DNN-related features. To this end, we show that the senone targets at the softmax layer of the DNN can be represented by Gaussian models and then convert the task of senone clustering into the task of Gaussian clustering.

It was shown in [15] that a log-linear model is equivalent to a Gaussian model. The general form of log-linear model is

$$P(s|x) = \frac{1}{Z(x)} \exp\left(\sum_i \lambda_{si} f_i(x)\right) \qquad (7)$$

where $f_i(x)$ is the *i*-th feature function for input *x*, λ_{si} is the weight for the *s*-th class and *i*-th feature, and $Z(x)$ is for the normalization. The mapping from a log-linear model to a Gaussian model $\mathcal{N}(x; \mu_s, \Sigma_s)$ is [15]

$$\Sigma_s = -\frac{1}{2}(\lambda_{s2} + \Delta\lambda_2)^{-1} \quad \text{and} \quad \mu_s = \Sigma_s \lambda_{s1} \tag{8}$$

where λ_{s2} and λ_{s1} are the second- and first-order weights, $\Delta\lambda_2$ is a constant to make the variance matrix Σ_s positive definite.

The softmax function in Eq. (2) can be considered as a log-linear model with

$$f_1(x) = \left[{v^L}^T, 1\right]^T, \quad \lambda_{s1} = [W_s, b_s], \quad \lambda_{s2} = 0 \tag{9}$$

where W_s, b_s are the weights and bias for senone s. Hence the corresponding Gaussian model for senone *s* is

$$\Sigma_s = \Sigma, \quad \text{and} \quad \mu_s = \Sigma[W_s, b_s] \tag{10}$$

Here the Gaussian distributions of all senones have the same variance matrix Σ, and the input vector to these Gaussians is $\left[{v^L}^T, 1\right]^T$. As every senone is represented by a Gaussian distribution, for any pair of senones s_i and s_j, we use the symmetric KL divergence as their distance measure, which is the metric to cluster a large set of senones into a small set.

The steps of learning a DNN with a small senone set are:

1. Train a DNN with a large senone set using the standard training procedure.
2. Convert the output layer of this large-senone-set DNN into Gaussian models according to Eq. (10).
3. Cluster the large set of Gaussians into a small set according to the symmetric KL divergence, and assign senone IDs for each cluster in the small set.
4. In the original senone alignment for training data, replace the large-set senone IDs with the small-set senone IDs obtained in Step 3, and then retrain the DNN.

With Eq. (10), senones in the large set are related to the well-trained DNN. We hope that clustering using DNN-related Gaussian parameters can achieve a better accuracy than the standard clustering procedure that uses single Gaussians generated from the traditional MFCC or PLP feature.

4. Experimental Evaluation

The proposed methods are evaluated using a Microsoft internal Windows Phone short message dictation task. The transcribed training data has 375 hours of US-English audio. The test set is extracted from the live data of the Windows Phone task. The input feature to CD-DNN-HMM system is a 29-dimension log-filter-bank feature with up to second-order derivatives. We augment the feature vectors with previous and next 5 frames (5-1-5). The system uses 6k senones, determined by the baseline CD-GMM-HMM system.

4.1 Experiments on DNN with a small number of hidden nodes

In this section, we report the results of learning a DNN with a small number of hidden nodes from a large-size DNN which cannot be deployed on devices. Because our proposed method in Section 3.1 is to train a small-size DNN by learning the output distribution of a large-size DNN, we sometimes call the large-size DNN as the teacher DNN, and the small-size DNN as the student DNN.

Using 375 hours of transcribed US-English audio data, we first train 5-layer DNNs with 2048 nodes and 512 nodes in each hidden layer, respectively, with the standard cross-entropy criterion in Eq. (4). The output layer is the same, with 6k senones. These two DNNs are model A and B in Table 1, with 16.32% and 19.90% WER, respectively.

Then, we use the steps described in Section 3.1 to learn small-size DNNs with 375 hours (hr), 750 hours, and 1500 hours of un-transcribed data from the same Windows Phone short message dictation task. They are denoted as model C, D, and E in Table 1, respectively. All these DNNs are initiated from the same pre-trained model without using any transcribed data. It is interesting to see that model C trained with only un-transcribed data gets slightly better WER than model B trained with the same data with transcription. It is possibly because the small-size DNN cannot have the capacity to learn well the hard labels generated from forced-alignment. Instead, the supervised signal from the large-size DNN seems to be an easier target for this small-size DNN to learn. We also observe that by adding 375, 750, and 1500 hours of un-transcribed data, the WERs of the small-size DNN keep dropping linearly. Therefore, we can infer that adding even more data will likely further reduce the accuracy gap between the small-size and large-size DNNs. However, the student DNN can never surpass the teacher DNN. It will get converged after seeing too much un-transcribed data. Currently, with 1500 hours of un-transcribed data, the small-size DNN learned with our proposed method can get relative 5.08% WER reduction from its counterpart trained with the standard process in Section 2.

Table 1: *Comparison of large-size and small-size DNNs trained with Eq. (4) and small-size DNNs learned with Eq.* (6). *The training data is US-English Windows Phone data.*

Model	**Training**	**WER**
A	2kx5 trained with 375hr transcribed data	16.32
B	512x5 trained with 375hr transcribed data	19.90
C	512x5 learned from model A with 375hr un-transcribed data	19.55
D	512x5 learned from model A with 750hr un-transcribed data	19.28
E	512x5 learned from model A with 1500hr un-transcribed data	18.89

In Table 1, the small-size DNN models (C, D, and E) are learned from model A, which is trained with the cross-entropy criterion. A natural question is that if the large-size teacher DNN in our proposed framework continues to improve its accuracy by some other techniques whether it could be translated to a better small-size student DNN using our KL-based training criterion. To answer this question, we trained a large size (2kx5) DNN model using the sequential training criterion [16] with 375 hours of transcribed data. This model is denoted as model F in Table 2, with 13.93% WER. Then, we use the steps described in Section 3.1 to learn a small-size DNN with 750 hours of un-transcribed audio. The new small-size DNN is denoted as model H in Table 2, with 16.66% WER which has relative 13.59% WER reduction from model D in Table 1. Therefore, even the teacher DNN is trained using a different criterion, the student DNN can learn very well from the teacher using the cross-entropy criterion in Eq. (6). This opens a door for further improvement by using an even more powerful teacher DNN which could be an ensemble of multiple DNNs or a giant-size DNN much wider and deeper than the standard DNN used on servers.

Model H gets 2.91% relative WER reduction from model G in Table 2 which is trained with sequential training criterion using 375 hours of transcribed data. The improvement is consistent with the gain obtained by model D from model B in Table 1, meaning that the cross-entropy learning with Eq. (6) using un-transcribed data can outperform the sequential learning with transcribed data for the small-size DNN.

Table 2: *Small-size DNN learning from a sequentially-trained large-size DNN. The training data is US-English Windows Phone data.*

Model	**Training**	**WER**
F	2kx5 trained with 375hr transcribed data using sequential training criterion	13.93
G	512x5 trained with 375hr transcribed data using sequential training criterion	17.16
H	512x5 learned from model F with 750hr un-transcribed data	16.66

Next, we conducted an experiment by learning the student DNN using 600 hours of German audio. The teacher DNN is model A in Table 1. With Eq. (6) the German audio is forward propagated through model A to generate $P_L(s_i|x_t)$ as the supervised signal to learn the student DNN which gets 21.71% WER on the US-English test set. This is interesting because the data used to learn the student DNN is not English data. It suggests that for the teacher DNN, the output space with 600 hours of German audio should be overlapped with the output space of English audio in some extent, and the distribution learning of Eq. (6) guides the student DNN well in those overlapped areas so that the student DNN can get reasonable WER on English task even trained with German audio. On the other hand, because the output space of German and English audio is not identical, we still see a clear gap between the student DNNs learned from German and English audio. Therefore, in order to build the best student DNN, the training data should be consistent with the target task.

4.2 Experiments on DNN with a small senone set

In this section, we compare two small-senone-set generation methods. Note that the experiments conducted in this section are independent with the experiments in Section 4.1. We want to train a DNN with only 1k senone set. The DNN still has 5 hidden layers, with 2048 nodes in each hidden layer. The first one uses the standard decision-tree-based process to generate a 1k senone set [14]. The second one uses the steps in Section 3.2 by clustering seones in the original 6k senone set into a 1k senone set. Table 3 lists the results, showing that our proposed method gives slight improvement from the standard method with 1.33% relative WER reduction. We believe this is because the clustering from 6k senone to 1k senone is closely related to DNN training. This should be better than the standard method which splits the decision tree by using the likelihood from single Gaussians. However, the slight improvement may indicate that the number of senones is more crucial to the final WER than how the senone set is obtained.

Table 3: *Comparison of two different small-senone set generation methods.*

Training	**WER**
Standard generation of 1k senone set	18.78
1k senone set merged from 6k senone set with steps in Section 3.2	18.53

5. Conclusions and Future Works

In the industry, we have strong interests to put DNNs on devices due to the increasingly popular mobile scenarios. To address the computational and storage limitation on devices, we proposed two methods to effectively learn a DNN with a small number of hidden nodes and a small set of senones. To learn a DNN with a small number of hidden nodes, we first train a large-size DNN and use its output distribution to teach the small-size DNN by minimizing the KL divergence of the output distribution between them. Evaluated with a short message dictation task for Windows Phone, the small-size DNN achieved 5.08% relative WER reduction from the small-size DNN trained with the standard process. As shown in Table 1, the WER of the learned DNN decreases linearly as a function of the amount of un-transcribed data in the early data-adding stage. Distribution learning enables the use of unlimited un-transcribed data, which can be obtained with our industry feed-back loop after product deployment. Therefore, we predict that the small-size DNN can approach the performance of the large-size DNN (although cannot surpass) if enough data is available for learning. It is also shown in Table 2 that accuracy improvement for the teacher DNN due to sequential training can be effectively translated to the student DNN by cross-entropy-based distribution learning. For the small senone set learning, by relating the senone clustering process with a trained DNN, we only get around 1.33% relative WER reduction. This indicates that the number of senones is more important to the final WER than how the senone set is obtained. Given the large performance gap between 6k and 1k senones, we will still use a relative large number of senones in the output layer and only focus on the small-size DNN learning method proposed in Section 3.1.

We are extending the proposed methods in several directions. First, although the proposed learning methods in this paper are for the purpose of training a small-size DNN that can be deployed on devices, these methods can also be used to improve large-size DNNs deployed on servers as long as the teacher DNN has a better accuracy. For example, we can have a teacher DNN which is an ensemble of multiple DNNs or a giant-size DNN which is much wider and deeper than the standard DNN used on servers. Either an ensemble DNN or a giant-size DNN is too complex to be used in deployment. However, they can be served as a good teacher DNN for a standard student DNN using distribution learning. Second, as shown in Section 4.1, the WERs of the small-size DNN keep dropping linearly when adding more un-transcribed data. With more training data, the performance of the small-size DNN will approach the performance of the large-size one although it can never surpass it. Third, to further reduce the number of parameters, the proposed learning method can be combined with other methods by exploring low-rank properties [8] or explicit environment variable modeling [17]. Finally, together with the low-rank method [8] and 16-bit quantization, we can put a DNN with only 4M-bytes size on devices. While the teacher DNN has 18.11% relative WER reduction from the device DNN with standard training and 360 hours of transcribed audio, we are able to improve the device DNN with 8.54% relative WER reduction by the distribution-learning criterion with 2100 hours of un-transcribed audio.

ACKNOWLEDGEMENT

We would like to thank Dr. Rich Caruana in Microsoft for providing valuable discussions.

6. 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] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, “Low-rank matrix factorization for deep neural network training with high-dimensional output targets,” in *Proc. ICASSP*, pp. 6655-6659, 2013.

[8] J. Xue, J. Li, and Y. Gong, “Restructuring of deep neural network acoustic models with singular value decomposition,” in *Proc. Interspeech*, pp. 2365-2369, 2013.

[9] V. Vanhoucke, A. Senior, and M. Z. Mao, “Improving the speed of neural networks on CPUs,” in *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[10] V. Vanhoucke, M. Devin, and G. Heigold. "Multiframe deep neural networks for acoustic modeling." In *Proc. IEEE Acoustics, Speech and Signal Processing*, 2013.

[11] X. Lei, A. Senior, A., A. Gruenstein, and J. Sorensen, “Accurate and compact large vocabulary speech recognition on mobile devices,” in *Proc. Interspeech*, 2013.

[12] C. Bucilu, R. Caruana, and A. Niculescu-Mizil, “Model compression,” In *Proc. of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

[13] L. Ba and R. Caurana, "Do deep nets really need to be deep?" *arXiv preprint arXiv:1312.6184*, 2013.

[14] S. J. Young, J. J. Odell, and P. C. Woodland, “Tree-based state tying for high accuracy acoustic modelling,” in *HLT*, pp. 307–312, 1994.

[15] G. Heigold, H. Ney, P. Lehnen, T. Gass, and R. Schluter, "Equivalence of generative and log-linear models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol.19, no.5, pp.1138-1148, 2011.

[16] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," In *Proc. ICASSP*, 2013.

[17] R. Zhao, J. Li, and Y. Gong, “Variable-component deep neural network for robust speech recognition,” In *Proc. Interspeech*, 2014.