FACTORIZED DEEP NEURAL NETWORKS FOR ADAPTIVE SPEECH RECOGNITION

Dong Yu[1]*, Xin Chen*[2]*, Li Deng*[1]

[1]Speech Research Group, Microsoft Research, Redmond, WA, USA
[2]Department of Computer Science, University of Missouri, Columbia, Missouri, USA
dongyu@microsoft.com, xinchen@mail.missouri.edu, deng@microsoft.com,

ABSTRACT

Recently, we have shown that context-dependent deep neural network hidden Markov models (CD-DNN-HMMs) can achieve very promising recognition results on large vocabulary speech recognition tasks, as evidenced by over one third fewer word errors than the discriminatively trained conventional HMM-based systems on the 300hr Switchboard benchmark task. In this paper, we propose and describe two types of factorized adaptive DNNs, improving the earlier versions of CD-DNN-HMMs. In the first model, the hidden speaker and environment factors and tied triphone states are jointly approximated; while in the second model, the factors are firstly estimated and then fed into the main DNN to predict tied triphone states. We evaluated these models on the small 30hr Switchboard task. The preliminary results indicate that more training data are needed to show the full potential of these models. However, these models provide new ways of modeling speaker and environment factors and offer insight onto how environment invariant DNN models may be constructed and subsequently trained.

Index Terms — automatic speech recognition, deep neural networks, factorized DNN, CD-DNN-HMM

1. INTRODUCTION

Recently significant progress has been made in applying artificial neural network (ANN) hidden Markov model (HMM) hybrid systems [1][2][3][4][5][6][7][8] for speech recognition. Two main factors contributed to the resurrection of the interests: the discovery of the strong modeling ability of deep neural networks (DNNs) and the availability of high-speed general purpose graphical processing units (GPGPUs) for efficiently training DNNs.

A notable advance is the context-dependent DNN-HMMs (CD-DNN-HMMs) [4][5][6], in which DNNs replace Gaussian mixture models (GMMs) and directly approximate the emission probabilities of the tied triphone states. CD-DNN-HMMs have recently been shown to be highly promising. They have achieved 16% [4][5] and 33% [6][7][8] relative recognition error reduction over strong, discriminatively trained CD-GMM-HMMs, respectively, on a voice search (VS) task [9] and the Switchboard (SWB) phone-call transcription task [10].

Speech signals have long been considered as complicated non-linear combination of factors such as speech itself, speaker, channel, and the acoustic environment. In this work, we extend the DNN to its factorized versions so that we may separate these factors and thus be better able to model inherently sophisticated interactions among these factors.

More specifically, we introduce and describe two factorized DNNs: joint and disjoint models. For the former, we jointly model the hidden factors and the tied triphone states. Each factor in this model is either on or off and can be combined with each other; i.e., several factors can be on at the same time. In other words, a K-factor hidden layer (i.e., with K units) can represent a total of 2^K possible combinatorial factors. In the disjoint model, on the other hand, the factors and tied triphone states are modeled separately using different DNNs. In one specific model that we have implemented, the hidden factor layer can only model K factors and one can only select one of these factors at a time. In other words, each factor in the disjoint model can be considered as a composite of several factors in the joint model.

We will introduce the basic DNN in Section 2 and describe the two factorized DNNs in detail in Section 3. Preliminary experimental results on Switchboard task are illustrated in Section 4. The paper is concluded with discussions in Section 5.

2. DEEP NEURAL NETWORKS

As a brief review, a DNN is a multi-layer perceptron (MLP) with many hidden layers. The lower layers of the DNN are sigmoid layers in which the j-th node of each layer converts the input vector $\boldsymbol{x}$ into output $y_j = \sigma\left(\sum_i W_{ij} x_i\right)$. This can be interpreted as a non-linear transformation of the input feature $\boldsymbol{x}$. Alternatively each node can be considered as taking binary values 0 and 1 following Bernoulli distribution. The mean field value of each node's output is sent to the next layer as the input. The last layer of a DNN transforms a number of Bernoulli distributed units into a multinomial distribution using the softmax operation

$$\mathrm{p}(y = k|\mathbf{x}; \theta) = \frac{exp\left(\sum_{i=1}^{H} W_{ik} x_i + a_k\right)}{Z(\boldsymbol{x})}, \tag{1}$$

where $y = k$ denotes the input been classified into the kth class, and W_{ik} is the weight between input unit x_i at the last layer and class label k.

Due to the deep structure and the complicated nonlinear surface introduced by the large number of hidden layers, it is important to employ effective training strategies. A popular trick is to initialize the parameters of each layer greedily and generatively by treating each pair of layers in DNNs as a restricted Boltzmann machine (RBM) before joint optimization of all the layers [11][12]. This learning strategy enables discriminative training to start from well initialized weights and is used in this study.

To learn the DNNs, we first train a Gaussian-Bernoulli RBM generatively in which the visible layer is the continuous input vector constructed from $2n + 1$ frames of speech features, in which n is the number of look-forward and look-backward frames.

We then use Bernoulli-Bernoulli RBMs for the remaining layers. When pre-training the next layer, $E(h_j|\mathbf{x};\theta) = p(h_j = 1|\mathbf{x};\theta)$ from the previous layer is used as the visible input vector based on the mean-field theory. This process continues until the last layer at which time error back-propagation (BP) is used to fine-tune all the parameters jointly by maximizing the frame-level cross-entropy between the true and the predicted probability distributions over class labels.

3. FACTORIZED DEEP NEURAL NETWORKS

Factorized DNNs are DNNs in which at least one layer is factorized. In speech recognition, a factor can be a cluster of speakers or a special acoustic environment (e.g., noise/channel condition and SNR). In this section, we will propose and discuss two such models: a joint model and a disjoint model. For the sake of discussion, we assume the input to the factorized layer is $\boldsymbol{x}$ (an $I \times 1$ vector) and the output of the same layer is $\boldsymbol{y}$ (an $L \times 1$ vector). We use y to indicate y-th element of $\boldsymbol{y}$. Although our discussion is based on the softmax layer, the model can be easily extended to sigmoid layers through error back-propagation.

3.1 Joint Factorized Model

In the joint factorized model, borrowing from [13], we approximate the joint probability as

$$p(\boldsymbol{y},\boldsymbol{h}|\boldsymbol{x}) = \frac{\exp(\boldsymbol{h}^T\boldsymbol{W}_y\boldsymbol{x})}{\sum_{y',\boldsymbol{h}'}\exp(\boldsymbol{h}'^T\boldsymbol{W}_{y'}\boldsymbol{x})}, \tag{2}$$

where $\boldsymbol{h}$ is a $K \times 1$ binary vector indicating hidden condition (e.g., speaker or environment), and $\boldsymbol{W}_y$ is a $K \times I$ matrix. We thus have

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}) &= \sum_{\boldsymbol{h}\in\{0,1\}^K} p(\boldsymbol{y},\boldsymbol{h}|\boldsymbol{x}) \\ &\propto \sum_{\boldsymbol{h}\in\{0,1\}^K} \exp(\boldsymbol{h}^T\boldsymbol{W}_y\boldsymbol{x}) \\ &= \sum_{\boldsymbol{h}\in\{0,1\}^K} \exp\left(\sum_{i,k} h_k W_{yik} x_i\right) \\ &= \prod_k \left(\exp\left(\sum_{i,h_k=0} h_k W_{yik} x_i\right) + \exp\left(\sum_{i,h_k=1} h_k W_{yik} x_i\right)\right) \\ &= \prod_k \left(1+\exp\left(\sum_i W_{yik}x_i\right)\right) \\ &= \prod_k \left(1+\exp(\boldsymbol{W}_{y:k}^T\boldsymbol{x})\right), \end{aligned} \tag{3}$$

where $\boldsymbol{W}_{y:k}$ is an $I \times 1$ vector along dimension i. The partition function is

$$\Omega = \sum_{y'}\prod_k\left(1+\exp\left(\sum_i W_{y'ik}x_i\right)\right). \tag{4}$$

As shown in Figure 1, in this factorized joint model, both the speaker or environmental condition and the output (e.g., tied triphone state) are unknown (hidden) and are jointly estimated. To predict the tied triphone states, we sum over all possible speaker or environmental conditions. Fortunately, the factorization trick carried out in (3) renders such gigantic summation feasible; i.e., the combinatorially large sum is reduced to a product.

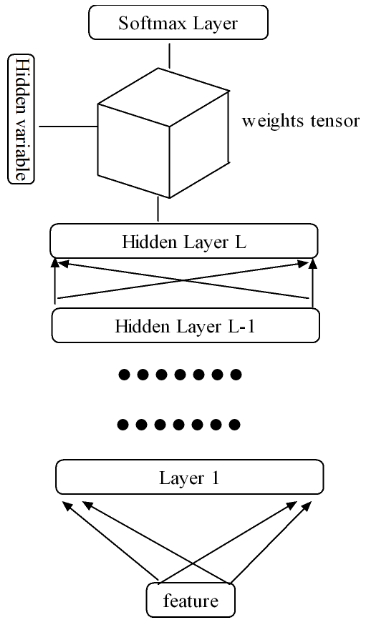

Figure 1: *Illustration of a typical architecture of the joint factorized DNN model*

We now turn to the learning problem. Note that $\boldsymbol{W}$ is an $L \times I \times K$ tensor and can be huge. To reduce the total number of weight parameters, we can restrict the number of hidden conditions K or the input dimensionality I. Alternatively, we can further assume that $\boldsymbol{W}$ is determined using F factors as

$$W_{yik} = \sum_f W_{if}^x W_{yf}^y W_{kf}^h. \tag{5}$$

Or in a matrix format

$$\boldsymbol{W} = \sum_f \boldsymbol{W}_{:f}^x \otimes \boldsymbol{W}_{:f}^y \otimes \boldsymbol{W}_{:f}^h, \tag{6}$$

where $\otimes$ is outer product. We then obtain

$$\begin{aligned} \log p(\boldsymbol{y}|\boldsymbol{x}) &= \log\prod_k\left(1+\exp\left(\sum_i W_{yik}x_i\right)\right) - \log\Omega \\ &= \sum_k \log\left(1+\exp\left(\sum_i\sum_f W_{if}^x W_{yf}^y W_{kf}^h x_i\right)\right) - \log\Omega \\ &= a_y - \log\sum_{y'}\exp a_{y'}, \end{aligned} \tag{7}$$

where

$$\begin{aligned} a_y &= \sum_k \log\left(1+\exp\left(\sum_i W_{yik}x_i\right)\right) \\ &= \sum_k \log\left(1+\exp\left(\sum_f\left(\sum_i x_i W_{if}^x\right) W_{yf}^y W_{kf}^h\right)\right). \end{aligned} \tag{8}$$

The model parameters are learned by maximizing the conditional log likelihood $\log p(\boldsymbol{y}|\boldsymbol{x})$, whose gradient with regard to $W_{\bar{y}ik}$ is

$$\begin{aligned}\frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{\bar{y}ik}} &= \frac{\partial\left(a_y - \log \sum_{y'} \exp a_{y'}\right)}{\partial W_{\bar{y}ik}} \\ &= \left(\delta_{\bar{y}y} - p(\bar{y}|\boldsymbol{x})\right) \sigma\left(\sum_{i'} W_{\bar{y}i'k} x_{i'}\right) x_i \\ &= \left(\delta_{\bar{y}y} - p(\bar{y}|\boldsymbol{x})\right) \sigma\left(W_{\bar{y}:k}^T x\right) x_i,\end{aligned} \tag{9}$$

where $\sigma(a) = \frac{1}{1+\exp(-a)}$ is sigmoid function, and we have used the fact that

$$\begin{aligned}\frac{\partial a_y}{\partial W_{\bar{y}ik}} &= \delta_{\bar{y}y} \frac{\partial \log\left(1 + \exp\left(\sum_{i'} W_{\bar{y}i'k} x_{i'}\right)\right)}{\partial W_{\bar{y}ik}} \\ &= \delta_{\bar{y}y} \frac{\exp\left(\sum_{i'} W_{\bar{y}i'k} x_{i'}\right)}{1 + \exp\left(\sum_{i'} W_{\bar{y}i'k} x_{i'}\right)} x_i \\ &= \delta_{\bar{y}y} \frac{1}{1 + \exp\left(-\sum_{i'} W_{\bar{y}i'k} x_{i'}\right)} x_i \\ &= \delta_{\bar{y}y} \sigma\left(\sum_{i'} W_{\bar{y}i'k} x_{i'}\right) x_i.\end{aligned} \tag{10}$$

If parameterization of (5) is used, we further have

$$\begin{aligned}\frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{if}^x} &= \sum_{\bar{y},k} \frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{\bar{y}ik}} \frac{\partial W_{\bar{y}ik}}{W_{if}^x} \\ &= \sum_{\bar{y},k} \frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{\bar{y}ik}} \frac{\partial \sum_f W_{if}^x W_{\bar{y}f}^y W_{kf}^h}{W_{if}^x} \\ &= \sum_{\bar{y},k} \frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{\bar{y}ik}} W_{\bar{y}f}^y W_{kf}^h \\ &= \boldsymbol{W}_{:f}^{y,T} \boldsymbol{A}_{:i:} \boldsymbol{W}_{:f}^h,\end{aligned} \tag{11}$$

where $A_{\bar{y}ik} = \frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{\bar{y}ik}}$ and $\boldsymbol{A}_{:i:}$ is an $L \times K$ slice of the tensor $\boldsymbol{A}$ along i. Similarly,

$$\begin{aligned}\frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{\bar{y}f}^y} &= \sum_{i,k} \frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{\bar{y}ik}} W_{if}^x W_{kf}^h \\ &= \boldsymbol{W}_{:f}^{x,T} \boldsymbol{A}_{\bar{y}::} \boldsymbol{W}_{:f}^h,\end{aligned} \tag{12}$$

and

$$\begin{aligned}\frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{kf}^h} &= \sum_{\bar{y},i} \frac{\partial \log p(y|\boldsymbol{x})}{\partial W_{\bar{y}ik}} W_{if}^x W_{\bar{y}f}^y \\ &= \boldsymbol{W}_{:f}^{x,T} \boldsymbol{A}_{::k} \boldsymbol{W}_{:f}^y.\end{aligned} \tag{13}$$

The error can then be propagated to lower layers by calculating

$$\begin{aligned}\frac{\partial \log p(y|\boldsymbol{x})}{\partial \boldsymbol{x}} &= \frac{\partial\left(a_y - \log \sum_{y'} \exp a_{y'}\right)}{\partial \boldsymbol{x}} \\ &= \frac{\partial a_y}{\partial \boldsymbol{x}} - \frac{\partial \log \sum_{y'} \exp a_{y'}}{\partial \boldsymbol{x}} \\ &= \frac{\partial a_y}{\partial \boldsymbol{x}} - \frac{\sum_{\bar{y}} \exp a_{\bar{y}} \frac{\partial a_{\bar{y}}}{\partial \boldsymbol{x}}}{\sum_{y'} \exp a_{y'}} \\ &= \sum_{\bar{y}} \left(\delta_{\bar{y}y} - p(\bar{y}|\boldsymbol{x})\right) \sum_k \sigma\left(\boldsymbol{W}_{\bar{y}:k}^T \boldsymbol{x}\right) \boldsymbol{W}_{y:k},\end{aligned} \tag{14}$$

where we have used the fact that

$$\begin{aligned}\frac{\partial a_y}{\partial \boldsymbol{x}} &= \frac{\partial \sum_k \log\left(1 + \exp\left(\boldsymbol{W}_{y:k}^T \boldsymbol{x}\right)\right)}{\partial \boldsymbol{x}} \\ &= \sum_k \frac{\exp\left(\boldsymbol{W}_{y:k}^T \boldsymbol{x}\right)}{1 + \exp\left(\boldsymbol{W}_{y:k}^T \boldsymbol{x}\right)} \boldsymbol{W}_{y:k} \\ &= \sum_k \frac{1}{1 + \exp\left(-\boldsymbol{W}_{y:k}^T \boldsymbol{x}\right)} \boldsymbol{W}_{y:k} \\ &= \sum_k \sigma\left(\boldsymbol{W}_{y:k}^T \boldsymbol{x}\right) \boldsymbol{W}_{y:k}.\end{aligned} \tag{15}$$

Note that the implementation of the learning algorithm based on the above gradient computation is tricky and needs to be smart. This is because many of the gradients above share part of the intermediate results. We only need to calculate these intermediate results once. Otherwise, the computation would be prohibitively large.

3.2 Disjoint Factorized Model

Unlike the joint factorized model, in the disjoint model as defined by the condition of

$$p(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}) = p(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{x}) p(\boldsymbol{h}|\boldsymbol{x}), \tag{16}$$

the two conditionals of $p(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{x})$ and $p(\boldsymbol{h}|\boldsymbol{x})$ are estimated separately first and then multiplied. For example, as adopted in this study, $p(\boldsymbol{h}|\boldsymbol{x})$ can be estimated using a separate DNN and its computation may use additional information (e.g., alignment information if second pass decoding is used). A special implementation (or architecture) of the disjoint factorized model is illustrated in Figure 2.

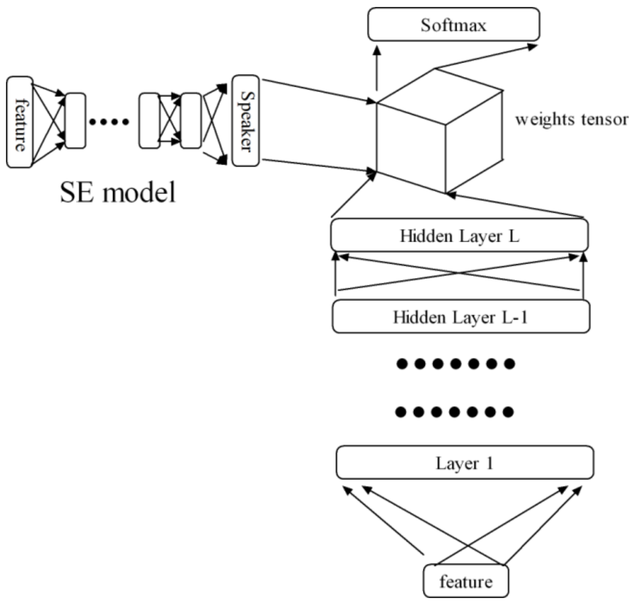

Figure 2: *Illustration of a typical architecture of the disjoint factorized DNN model*

On the other hand, $p(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{x})$ can be estimated using the softmax model

$$p(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{x}) = \frac{\exp\left(\boldsymbol{h}^T \boldsymbol{W}_y \boldsymbol{x}\right)}{\sum_{y'} \exp\left(\boldsymbol{h}^T \boldsymbol{W}_{y'} \boldsymbol{x}\right)}. \tag{17}$$

Note that although the same softmax formula is used as for (2), the meaning is very different. In (2) we approximated the probability of the pair $(\boldsymbol{y}, \boldsymbol{h})$ given the input $\boldsymbol{x}$, and thus the partition function has an additional (huge) sum over combinatorial $\boldsymbol{h}$. In contrast, in (17) we approximate only the probability of $\boldsymbol{y}$ when we know both input $\boldsymbol{x}$ and factor $\boldsymbol{h}$. The marginal conditional probability is thus

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{h}} p(\boldsymbol{y}|\boldsymbol{h},\boldsymbol{x})\,p(\boldsymbol{h}|\boldsymbol{x}) = \sum_{\boldsymbol{h}} \frac{\exp(\boldsymbol{h}^T \boldsymbol{W}_y \boldsymbol{x})}{\sum_{y'} \exp(\boldsymbol{h}^T \boldsymbol{W}_{y'} \boldsymbol{x})} p(\boldsymbol{h}|\boldsymbol{x}). \quad (18)$$

When the *output layer* of the separate DNN with K output nodes is used to estimate $h_k = p(h = k|\boldsymbol{x})$, h can only take one of K values and thus we have further simplification of

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{k\in[1\,K]} \frac{\exp(\boldsymbol{h}^T \boldsymbol{W}_y \boldsymbol{x})}{\sum_{y'} \exp(\boldsymbol{h}^T \boldsymbol{W}_{y'} \boldsymbol{x})} p(h = k|\boldsymbol{x}) = \sum_{k} \frac{h_k \exp(\boldsymbol{W}_{y:k}^T \boldsymbol{x})}{\sum_{y'} \exp(\boldsymbol{W}_{y':k}^T \boldsymbol{x})}. \quad (19)$$

Equation (19) is equivalent to say we build a separate softmax layer for each type of speaker or environment.

In this disjoint model, the training is straightforward. Since we build a DNN to estimate factors (e.g., speaker or environment) and a cluster of DNNs for tied triphone states, one for each factor trained firstly using all the data and then adapted using only data associated with that factor. No change of leaning algorithms from those reviewed in Section 2 is needed.

4. EXPERIMENTS

In this section we report some of the preliminary results of applying factorized DNNs on the Switchboard task.

The training and development sets contain 30 hours and 6.5 hours of data randomly sampled from the 309-hour Switchboard-I training set. The 1831-segment SWB part of the NIST 2000 Hub5 eval set (6.5 hours) was used as the test set. To prevent speaker overlap between the training and test sets, speakers occurring in the test set were removed from the training and development sets. We evaluated the models only on the 30-hr (instead of 309-hr) training set at this stage due to the high computational cost incurred with factorized models.

The system uses 13-dimensional PLP features with windowed mean-variance normalization and up to third-order derivatives, reduced to 39 dimensions by HLDA. The speaker-independent crossword triphones use the common 3-state topology and share 1504 CART-tied states determined on the conventional GMM system. The trigram language model was trained on the 2000h Fisher-corpus transcripts and interpolated with a written text trigram. Test-set perplexity with the 58k dictionary is 84. Recognition is done in a single-pass without any speaker adaptation.

The GMM-HMM baseline system has 40 Gaussian mixtures trained with maximum likelihood (ML) and refined discriminatively with the boosted maximum-mutual-information (BMMI) criterion. Using more than 40 Gaussians did not improve the ML result.

The CD-DNN-HMM system replaces the Gaussian mixtures with likelihoods derived from the DNN posteriors [1][5][6]. The input to the DNN contains 11 (5-1-5) frames of the HLDA-transformed features. The DNN contains 429-2048-2048-2048-2048-2048-1504 neurons at different layers. The joint and disjoint factorized CD-DNN-HMM systems replace the DNN posteriors with eq. (3) and (19) respectively. All these models use five hidden layers each of which has 2048 hidden units. In the joint model the dimension of the factor $\boldsymbol{h}$ is set to 7 with $2^7 = 128$ possible factor combinations to make training tractable. In the disjoint model, the factor $\boldsymbol{h}$ is the speaker side ID and there are 354 of it in the training set. The factor posteriors themselves are estimated from a separate DNN with 429-128-128-128-354 units at different layers. We chose to use 128 hidden units to make it comparable to the joint factor model when the hidden units are used as the factors (similar to the bottle-neck feature). All DNNs were trained using the minibatch stochastic gradient ascent algorithm with 256 frames in each minibatch. For both joint and disjoint model we only applied the factorized layer at the top layer. In all the experiments, we have used the ML trained GMM system to generate the senone labels for DNN training. To alleviate the overfitting problem due to significantly more parameters in the factorized models, smaller learning rate (one tenth of what we used in [5][6]), L2 regularization and cross validation were used to control the training process. Additional training details can be found in [5][6].

The preliminary results are summarized in Table 1. It is expected that the CD-DNN-HMM significantly outperforms the conventional CD-GMM-HMM with a 27% relative WER reduction. Unfortunately both factorized models perform only slightly better than the non-factorized DNN-HMMs and they perform worse than the method of 2-pass feature discriminative linear regression (fDLR) [7] with which a linear transformation of the input feature is estimated to maximize the posterior probability of the senone alignment generated by the first-pass recognition result. The insignificant gain observed in this experiment might be partially due to the fact that factorized DNNs use considerably more parameters and our experiments (work in progress) are so far limited to only 30 hours of training data.

Table 1. Comparisons: CD-GMM-HMM, conventional CD-DNN-HMM, and factorized CD-DNN-HMMs. Trained on 30-hr training set. WER reported on SWB NIST 2000 Hub5 eval set.

Setup	Test WER (%)
CD-GMM-HMM	**34.8**
CD-DNN-HMM	**25.7**
Joint Fac DNN	**25.6**
Disjoint Fac DNN	**25.6**
fDLR (2-pass)	**25.3**

5. DISCUSSIONS

In this paper, we have introduced and described two types of factorized DNNs -- joint and disjoint -- for large vocabulary speech recognition, aimed to accommodate or be adaptive to a wide range of speaker and environmental conditions. The proposed approaches represent new ways of modeling speaker and environment factors and offer insight onto how we may effectively construct and train environment invariant DNN models. We hope the models presented in this paper can trigger new ideas and techniques to further advance the state of the art.

Our preliminary results indicate that given a relatively small amount of training data in our work progressed thus far, the factorized DNNs only slightly outperform the conventional DNN (not statistically significant). Ongoing experiments are further testing out these models. These include the use of more training data, adjustment of the number of factors, and adoption of better training strategy. Our future directions will also integrate the adaptive modeling strategy presented in this paper to new deep architectures beyond DNNs and develop applications beyond speech recognition [14].

6. REFERENCES

[1] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco,“Connectionist Probability Estimators in HMM Speech Recogni-tion,” *IEEE Trans. Speech and Audio Proc.,* January 1994.

[2] A. Mohamed, G. E. Dahl, and G. E. Hinton, “Acoustic modeling using deep belief networks,” *IEEE Trans. on Audio, Speech, and Lang. Proc. Jan. 2012*.

[3] A. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition", in *Proc. Interspeech* 2010, pp. 1692-1695.

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

[5] G.E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pretrained deep neural networks for large vocabulary speech recognition", *IEEE Trans. Audio, Speech, and Lang. Proc. Jan.* 2012.

[6] F. Seide, G. Li and D. Yu, “Conversational speech transcription using context-dependent deep neural networks,” Proc. *Interspeech* 2011, pp. 437-440.

[7] F. Seide, G. Li, X. Chen, D. Yu, “Feature engineering in context-dependent deep neural networks for conversational speech transcription,” Proc. *ASRU* 2011, pp. 24-29.

[8] D. Yu, F. Seide, G. Li, L. Deng, "Exploiting Sparseness In Deep Neural Networks For Large Vocabulary Speech Recognition", Proc. *ICASSP*, March 2012.

[9] D. Yu, Y. C. Ju, Y. Y. Wang, G. Zweig, and A. Acero, “Automated directory assistance system - from theory to practice,” Proc. *Interspeech*, 2007, pp. 2709–2711.

[10] J. Godfrey and E. Holliman, “Switchboard-1 Release 2,” *Linguistic Data Consortium*, Philadelphia, 1997.

[11] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

[12] G. Hinton, S. Osindero, and Y. Teh, “A Fast Learning Algorithm for Deep Belief Nets”, *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[13] R. Memisevic, C. Zach, G. Hinton, and M. Pollefeys. “Gated softmax classication”, *NIPS* 2011.

[14] D. Yu and L. Deng, “Deep learning and its applications to signal and information processing , *IEEE Signal Processing Magazine*, Vol. 28, January 2011.