Large Vocabulary Speech Recognition Using Deep Tensor Neural Networks

Dong Yu[1], *Li Deng*[1], *Frank Seide*[2]

[1]Microsoft Research, Redmond, WA, USA
[2]Microsoft Research Asia, Beijing, China
{dongyu,deng,fseide}@microsoft.com

Abstract

Recently, we proposed and developed the context-dependent deep neural network hidden Markov models (CD-DNN-HMMs) for large vocabulary speech recognition and achieved highly promising recognition results including over one third fewer word errors than the discriminatively trained, conventional HMM-based systems on the 300hr Switchboard benchmark task. In this paper, we extend DNNs to deep tensor neural networks (DTNNs) in which one or more layers are double-projection and tensor layers. The basic idea of the DTNN comes from our realization that many factors interact with each other to predict the output. To represent these interactions, we project the input to two nonlinear subspaces through the double-projection layer and model the interactions between these two subspaces and the output neurons through a tensor with three-way connections. Evaluation on 30hr Switchboard task indicates that DTNNs can outperform DNNs with similar number of parameters with 5% relative word error reduction.

Index Terms: automatic speech recognition, tensor deep neural networks, CD-DNN-HMM, large vocabulary

1. Introduction

Recently we proposed and developed the context-dependent deep neural network (DNN) hidden Markov models (HMMs) (CD-DNN-HMMs) for large vocabulary speech recognition [1][2][3][4][5]. CD-DNN-HMM is a special artificial neural network (ANN) HMM hybrid system [7][8][9] proposed two decades ago for speech recognition. In CD-DNN-HMMs, DNNs replace Gaussian mixture models (GMMs) and directly approximate the emission probabilities of the tied triphone states. CD-DNN-HMMs have achieved 16% [1][2] and 33% [3][4][5][6] relative recognition error reduction over strong, discriminatively trained CD-GMM-HMMs, respectively, on a voice search (VS) task [10] and the Switchboard (SWB) phone-call transcription task [11].

In this work, we extend the DNN to a novel deep tensor neural network (DTNN) in which one or more layers are double-projection and tensor layers (see details in Section 2). The basic idea of the DTNN comes from our realization that many factors, such as noisy speech, noise and speaker, interact with each other to predict the output. To represent these interactions, we project the input to two nonlinear subspaces through the double-projection layer and model the interactions between these two subspaces and the output neurons through a tensor with three-way connections. In addition to the proposal of this new type of deep model, our original contributions in this work also include a novel approach to reduce the tensor layer to a conventional sigmoid layer, the development of the learning algorithms associated with DTNN, and empirical evaluations of the new deep model on large vocabulary speech recognition.

We will introduce DTNN, explain how to simplify it, and describe the detailed learning algorithms in Section 2. The experimental results on Switchboard task are presented in Section 3. We will conclude the paper in Section 4.

2. Deep Tensor Neural Network

Deep tensor neural network (DTNN) is a new type of deep neural network (DNN) with one or more layers replaced with double-projection and tensor layers, to be defined shortly. In this section we describe DTNN and the related learning and scoring algorithms. We denote the input to the DTNN as x, an $I \times 1$ vector, and the output as y, a $C \times 1$ vector.

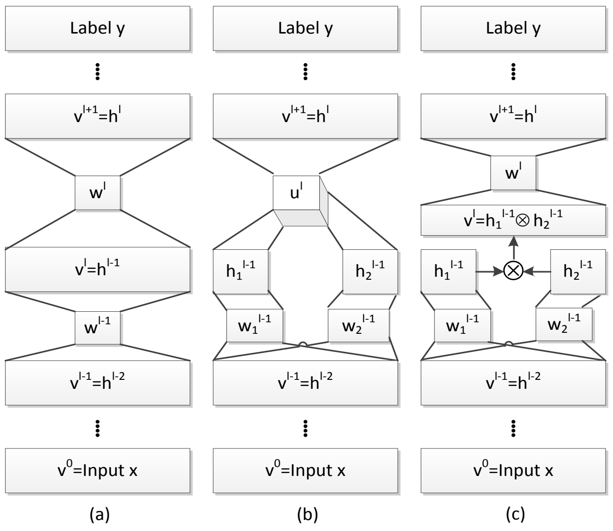

Figure 1: *Architectural illustrations of DNN and DTNN. (a) DNN. (b) DTNN: hidden layer h^{l-1} consists of two parts: h_1^{l-1} and h_2^{l-1}. Hidden layer h^l is a tensor layer to which the connection weights u^l form a three-way tensor. (c) An alternative representation of (b): tensor u^l is replaced with matrix w^l when v^l is defined as the cross product $h_1^{l-1} \otimes h_2^{l-1}$.*

2.1. Tensor Layer and Double-Projection Layer

Figure 1 illustrates and compares DNN and DTNN. More specifically, sub-figure (a) is a conventional DNN with a total of $L+1$ layers and sub-figure (b) is the corresponding DTNN where hidden layer h^l is replaced with a tensor layer. Note that the immediate lower layer, hidden layer h^{l-1}, is separated into two parts h_1^{l-1} (a $K_1^{l-1} \times 1$ vector) and h_2^{l-1} (a $K_2^{l-1} \times 1$ vector). These two parts connect to hidden layer h^l through the three-way tensor u^l which is represented with a cube in the figure. The two hidden layer vectors h_1^{l-1} and h_2^{l-1} may be augmented with value 1 as the last element when connected to hidden layer h^l. However, this is unnecessary since the same effect may be achieved by initializing weights and biases so that one of the units always outputs 1.

Sub-figure (c) is an alternative view of the same DTNN shown in sub-figure (b). By defining

$$v^l = \mathrm{vec}\left(h_1^{l-1}\left(h_2^{l-1}\right)^T\right), \tag{1}$$

where vec(·) is the column-vectorized representation of the matrix, we can organize and rewrite tensor u^l into matrix w^l as represented by a rectangular in (c). This rewriting allows us to bridge between the conventional layers and the tensor layers and to define the same interface in describing these two different types of layers. For example, in sub-figure (c) hidden layer h^l can now be considered as a conventional layer and can be learned using the conventional backpropagation (BP) algorithm. Hidden layer h^{l-1}, however, is still a special layer that contains two output parts h_1^{l-1} and h_2^{l-1} that are determined by two separate weight matrices w_1^{l-1} and w_2^{l-1}. We call this kind of layer double-projection (DP) layer.

In the DTNN shown in Figure 1, hidden layer h^{l-2} is a conventional layer. In other words, h^{l-2} connects with h_1^{l-1} and h_2^{l-1} through weight matrices w_1^{l-1} and w_2^{l-1} instead of tensors. However, nothing would prevent hidden layer h^{l-2} from being a DP layer, in which case it can also be separated into two parts h_1^{l-2} and h_2^{l-2} and these two parts may connect to h_1^{l-1} and h_2^{l-1} through tensors u_1^{l-1} and u_2^{l-1}. Fortunately, by defining input v^{l-1} to hidden layer h^{l-1} as

$$v^{l-1} = \mathrm{vec}\left(h_1^{l-2}\left(h_2^{l-2}\right)^T\right), \tag{2}$$

tensors u_1^{l-1} and u_2^{l-1} can be rewritten as matrices.

In summary, there are three types of layers in DTNNs: the conventional sigmoid layer, the DP layer, and the softmax layer that connects the final hidden layer to labels. In all the layers the input is always v^l for layer h^l. For the first layer we have

$$v^l = v^0 = x. \tag{3}$$

For layers $l > 0$, if the previous layer is a conventional sigmoid layer $v^l = h^{l-1}$, otherwise v^l is defined using Eq. (1). Note v^l is a $K_v^l \times 1$ column vector.

For the conventional sigmoid layer,

$$h^l = \sigma\left(z^l(v^l)\right) = \sigma((w^l)^T v^l + a^l), \tag{4}$$

where w^l is the weight matrix, a^l is the bias, $\sigma(x) = 1\,/\,((1 + \exp(-x)))$ is the sigmoid function applied element-wise, and $z^l(v^l) = (w^l)^T v^l + a^l$ is the activation vector given input v^l.

For the DP layer,

$$h_i^l = \sigma\left(z_i^l(v^l)\right) = \sigma\left(\left(w_i^l\right)^T v^l + a_i^l\right) \tag{5}$$

where $i = \{1,2\}$ indicates the part number. Each DP layer projects the input v^l onto two non-linear subspaces h_1^l and h_2^l. The first and second order statistics of these two projections are then used as the input feature to the adjacent higher layer as quantified by Eq. (1). In other words, DP layers can capture higher order statics.

The softmax layer converts the last hidden layer into a multinomial distribution defined by

$$p(y|v^L) = \frac{\exp\left(\left(w_y^L\right)^T v^L + a_y^L\right)}{\sum_{y'} \exp\left(\left(w_{y'}^L\right)^T v^L + a_{y'}^L\right)}, \tag{6}$$

where w_y^L is a column vector of the weight matrix w^L.

2.2. Learning DTNN Parameters

As with DNN, the DTNN model parameters are optimized to maximize the cross entropy

$$D = \sum_y \tilde{p}(y|x) \log p(y|x), \tag{7}$$

where $\tilde{p}(y|x)$ is the target distribution that is typically the empirical distribution observed from the training set. The parameters can be learned using the BP algorithm.

The gradients associated with the softmax layer and the conventional sigmoid layer are the same as in DNNs. More specifically, for the softmax layer

$$\frac{\partial D}{\partial w^L} = v^L\left(e^L(x)\right)^T, \tag{8}$$

$$\frac{\partial D}{\partial a^L} = e^L(x), \tag{9}$$

where w^L is the $K_v^L \times C$ weight matrix, a^L is the $C \times 1$ bias column vector, and $e^L(x)$ is a $C \times 1$ error column vector with $e_i^L(x) = \left(\tilde{p}(y = i|x) - p(y = i|x)\right)$. For other layers with $l < L$ we define $e^l(x) = \frac{\partial D}{\partial v^{l+1}}$.

In the softmax layer, the error can be propagated to the immediately previous layer according to

$$e^{L-1}(x) = \frac{\partial D}{\partial v^L} = w^L e^L(x) \tag{10}$$

Similarly, for the conventional sigmoid layer, we have

$$\frac{\partial D}{\partial w^l} = v^l\left(\mathrm{diag}\left(\sigma'\left(z^l(v^l)\right)\right)e^l(x)\right)^T, \tag{11}$$

$$\frac{\partial D}{\partial a^l} = \mathrm{diag}\left(\sigma'\left(z^l(v^l)\right)\right)e^l(x), \tag{12}$$

and

$$e^{l-1}(x) = \frac{\partial D}{\partial v^l} = w^l \mathrm{diag}\left(\sigma'\left(z^l(v^l)\right)\right)e^l(x), \tag{13}$$

where $\sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right)$ is the gradient of the sigmoid function applied element-wise, and diag(.) is the diagonal matrix determined by the operant.

However, the gradients as well as their derivations are more complicated for the DP layer, which we derive now. Note for the DP layer we have

$$\begin{aligned} v^{l+1} &= \mathrm{vec}\left(h_1^l\left(h_2^l\right)^T\right) \\ &= \left(h_2^l \otimes I_{K_1^l}\right)\mathrm{vec}\left(h_1^l\right) = \left(h_2^l \otimes I_{K_1^l}\right)h_1^l \\ &= \left(I_{K_2^l} \otimes h_1^l\right)\mathrm{vec}\left(\left(h_2^l\right)^T\right) = \left(I_{K_2^l} \otimes h_1^l\right)h_2^l, \end{aligned} \tag{14}$$

where $\otimes$ is cross product and I is identity matrix. $v^{l+1}$is thus a $\left(K_1^l \times K_2^l\right) \times 1$ column vector whose elements are $v_{j+k\cdot K_1^l}^l = h_{1,j}^l h_{2,k}^l$, where we assume matrix and vector index is 0 based. This leads to the gradients

$$\frac{\partial (v^{l+1})^T}{\partial h_1^l} = \frac{\partial \left(\left(h_2^l \otimes I_{K_1^l} \right) h_1^l \right)^T}{\partial h_1^l} = \left(h_2^l \right)^T \otimes I_{K_1^l} \tag{15}$$

whose $\left(i, j + k \cdot K_1^l\right)$-th element is $\delta(i = j) h_{2,k}^l$, and

$$\frac{\partial (v^{l+1})^T}{\partial h_2^l} = \frac{\partial \left(\left(I_{K_2^l} \otimes h_1^l \right) h_2^l \right)^T}{\partial h_2^l} = I_{K_2^l} \otimes \left(h_1^l \right)^T \tag{16}$$

whose $\left(i, j + k \cdot K_1^l\right)$-th element is $\delta(i = k) h_{1,j}^l$.

Noting that for the parts $i \in \{1,2\}$

$$\frac{\partial \left(h_i^l\right)^T}{\partial \mathrm{vec}\left(w_i^l\right)} = v^l \otimes \mathrm{diag}\left(\sigma'\left(z_i^l(v^l)\right)\right), \tag{17}$$

$$\frac{\partial \left(h_i^l\right)^T}{\partial a_i^l} = \mathrm{diag}\left(\sigma'\left(z_i^l(v^l)\right)\right), \tag{18}$$

and

$$\frac{\partial \left(h_i^l\right)^T}{\partial v^l} = w_i^l \mathrm{diag}\left(\sigma'\left(z_i^l(v^l)\right)\right). \tag{19}$$

By defining $e_i^l(x) = \frac{\partial D}{\partial h_i^l}$ we get

$$e_i^l(x) = \frac{\partial D}{\partial h_i^l} = \frac{\partial h^l}{\partial h_i^l} \frac{\partial D}{\partial h^l} \tag{20}$$

More specifically,

$$e_1^l(x) = \left(\left(h_2^l\right)^T \otimes I_{K_1^l} \right) e^l(x) = \left[e^l(x)\right]_{K_1^l, K_2^l} h_2^l, \tag{21}$$

$$e_2^l(x) = \left(I_{K_2^l} \otimes \left(h_1^l\right)^T \right) e^l(x) = \left(\left[e^l(x)\right]_{K_1^l, K_2^l} \right)^T h_1^l. \tag{22}$$

The gradients needed for BP algorithm in the DP layer are thus

$$\begin{aligned} \frac{\partial D}{\partial w_i^l} &= \left[\frac{\partial D}{\partial \mathrm{vec}\left(w_i^l\right)} \right]_{K_v^l, K_i^l} \\ &= \left[\frac{\partial h_i^l}{\partial \mathrm{vec}\left(w_i^l\right)} \frac{\partial D}{\partial h_i^l} \right]_{K_v^l, K_i^l} \\ &= \left[\left(v^l \otimes \mathrm{diag}\left(\sigma'\left(z_i^l(v^l)\right)\right) \right) e_i^l(x) \right]_{K_v^l, K_i^l} \\ &= v^l \left(\mathrm{diag}\left(\sigma'\left(z_i^l(v^l)\right)\right) e_i^l(x) \right)^T, \end{aligned} \tag{23}$$

$$\frac{\partial D}{\partial a_i^l} = \frac{\partial h_i^l}{\partial a_i^l} \frac{\partial D}{\partial h_i^l} = \mathrm{diag}\left(\sigma'\left(z_i^l(v^l)\right)\right) e_i^l(x) \tag{24}$$

and

$$\begin{aligned} e^{l-1}(x) &= \frac{\partial D}{\partial v^l} = \sum_{j \in \{1,2\}} \frac{\partial h_j^l}{\partial v^l} \frac{\partial D}{\partial h_j^l} = \sum_{j \in \{1,2\}} \frac{\partial h_j^l}{\partial v^l} e_i^l(x) \\ &= \sum_{j \in \{1,2\}} w_i^l \mathrm{diag}\left(\sigma'\left(z_i^l(v^l)\right)\right) e_i^l(x) \end{aligned} \tag{25}$$

2.3. Comparisons with Other Tensor Networks

Tensors have been used in neural networks in the past. More recently, Memisevic et al. [12] proposed to gate the softmax layer with a hidden factor layer and the tensor was used to model the joint probability of factors and labels. Yu, Chen, and Deng [14] extended the gated softmax layer to DNNs and also proposed a tensor-based architecture that uses separately predicted gating factors. Hutchinson, Deng and Yu [13] replaced the single sigmoid hidden layer with a tensor layer in the stacking networks. The DTNN proposed in this work is different from all the above prior arts in that it uses double-projection layers to automatically factorize information which is later combined through the tensor layers, and that the DP layers and tensor layers can be flexibly incorporated into the DNN architecture. This work also provides a unified way to train DNN and DTNN by mapping the input feature of each layer to a vector and the tensor to a matrix.

3. Experimental Results

We have evaluated the proposed DTNN 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 mainly because training DNN and DTNN is still time consuming for large data sets due to a lack of efficient cross machine parallel algorithms.

The system uses a 39-dimensional feature that was reduced using HLDA from the mean-variance normalized 13-dimensional PLP features and up to third-order derivatives. The common left-to-right 3-state speaker-independent crossword triphones 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.

The GMM-HMM baseline system has a mixture of 40 Gaussians in each HMM state. It was 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.

Both the CD-DNN-HMM and CD-DTNN-HMM systems replace the Gaussian mixtures with likelihoods derived from the DNN and DTNN posteriors, respectively. The input to the DNN and DTNN contains 11 (5-1-5) frames of the HLDA-transformed features. The baseline DNN has 429 input nodes, 1504 output nodes and 5 hidden layers with 2048 nodes (a 429-2048x5-1504 architecture). Since input and output layers are the same we ignore them when describing the DNN architecture from now on.

We use the notation of two numbers enclosed in the parentheses to denote the DP layers in a DTNN. As an example, (96:96) denotes a DP layer with 96 units in each of the two sub-hidden parts. Thus, (64:64)x1-2kx4 denotes a DTNN that contains a DP layer with 64 units at each part, followed by 4 conventional sigmoid hidden layers each of which has 2k units. A DTNN whose hidden layers are (96:96)x5 has a similar number of parameters in total to the baseline conventional DNN. In our experiments, the conventional DNNs are pre-trained with the DBN-pretraining algorithm before they are fine-tuned using the BP algorithm. However, we have not developed similar pretraining algorithms for DTNNs. DTNNs are thus trained using the BP algorithm presented in this section starting from randomly initialized weights. Pretraining typically provides

0.3%-0.5% absolute WER reduction on a 5-hidden layer DNN.

Table 1 compares the effect of different DTNN configurations. To reduce the overall training time we trained DTNNs for only 10 epochs, in which the first 5 epochs were carried out using a learning rate of 0.0003 per sample and the remaining 5 epochs with a learning rate of 0.000008 per sample.

Note that even with this highly sub-optimal learning strategy, a DNN with 5 hidden layers (shaded row in the table) already significantly outperforms the CD-GMM-HMM trained using the BMMI criterion. The results in Table 1 are organized so that all configurations above the shaded line underperform the conventional DNN and all the configurations below the shaded line outperform DNN.

Table 1. *Comparing the effect of different DTNN configurations on the word error rate. In these experiments, DTNNs were trained for only 10 epochs, in which the first 5 epochs were carried out using a learning rate of 0.0003 per sample and the remaining 5 epochs with a learning rate of 0.000008 per sample.*

Configuration	Test WER
CD-GMM-HMM (BMMI)	34.8%
(64:64)x1-2kx4	31.0%
(96:96)x5	28.5%
2kx5 (DNN)	28.3%
2kx2-(64:64)x3	27.9%
2kx2-(64:64)x1-2kx2	27.6%
2kx2-(96:96)x3	27.6%
2kx4-(64:64)x1	27.3%
2kx4-(96:96)x1	27.0%

Examining Table 1, we can make three observations. First, configuration (96:96)x5 in which all layers are DP layers (and hence all layers are tensor layers) performs similarly to the DNN baseline that contains a similar number of parameters even though the DNN was pre-trained while the DTNN was not. Second, the configuration in which only the bottom (first) layer was replaced with the DP layer performs the worst. This may suggest that the DP layer is not able to capture full information when converting from the real valued input features to the binary hidden representations, especially since the upper layers cannot recover from information loss. Third, the configurations that replace the top hidden layer with the DP layer perform the best and achieve more than 1% absolute (or 5% relative) WER reduction over the DNN. This suggests that the DP (and tensor) layers are better suited to operate upon binary features, consistent with the findings from [13].

Table 2. *Comparing the effect of different DTNN configurations on the word error rate. Learning strategy was tuned for DNN and applied to DTNN.*

Configuration	Test WER
CD-GMM-HMM (BMMI)	34.8%
2kx5	27.4%
2kx2-(64:64)x1-2kx2	26.8%
2kx4-(64:64)x1	26.4%
2kx4-(96:96)x1	26.2%

To eliminate the possibility that the training strategy adopted in Table 1 favors DTNNs over DNNs, we tuned the learning strategy, including learning rates and schedule, for DNN and used the tuned learning strategy to train DTNNs. More specifically, DNNs and DTNNs were trained for 15 epochs, in which the first 9 epochs were carried out using a learning rate of 0.0003 per sample and the remaining 6 epochs with a learning rate 0.000008 per sample. The new results are summarized in Table 2. These results further confirmed the effectiveness of DTNN.

4. Conclusions

In this paper we have proposed and implemented a novel deep model called DTNN, in which one or more layers are DP and tensor layers. We have described an approach to map the tensor layers to the conventional sigmoid layers so that the former can be treated and trained in a similar way to the latter. With this mapping we can consider a DTNN as the DNN augmented with DP layers and so the BP learning algorithm of DTNNs can be cleanly derived.

We have evaluated different configurations of the DTNN architecture on the SWB task using 30 hours of training data. The experimental results demonstrate that when the DP layer is placed at the top layer of the DTNN, it performs the best and outperforms the corresponding DNN by more than 5% relative WER reduction.

5. 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,” Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2010.

[2] 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, vol. 20, no. 1, pp. 33-42.

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

[4] 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.

[5] D. Yu, F. Seide, G. Li, L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition,” Proc. ICASSP 2012, pp. 4409-4412.

[6] X. Chen, A. Eversole, G. Li, D. Yu, and F. Seide, “Pipelined Back-Propagation for Context-Dependent Deep Neural Networks”, Interspeech 2012.

[7] 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.

[8] 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, vol. 20, no. 1, pp. 14-22.

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

[10] 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.

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

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

[13] B. Hutchinson, L. Deng, and D. Yu, “A deep architecture with bilinear modeling of hidden representations: Applications to phonetic recognition,” Proc. ICASSP 2012. pp. 4805-4808.

[14] D. Yu, X. Chen, and L. Deng, “Factorized deep neural networks for adaptive speech recognition,” Proc. Int. Workshop on Statistical Machine Learning for Speech Processing, 2012.