A DEEP ARCHITECTURE WITH BILINEAR MODELING OF HIDDEN REPRESENTATIONS: APPLICATIONS TO PHONETIC RECOGNITION

*Brian Hutchinson**

EE Dept, University of Washington
brianhutchinson@ee.washington.edu

Li Deng, Dong Yu

Microsoft Research Redmond
{deng,dongyu}@microsoft.com

ABSTRACT

We develop and describe a novel deep architecture, the Tensor Deep Stacking Network (T-DSN), where multiple blocks are stacked one on top of another and where a bilinear mapping from hidden representations to the output in each block is used to incorporate higher-order statistics of the input features. A learning algorithm for the T-DSN is presented, in which the main parameter estimation burden is shifted to a convex sub-problem with a closed-form solution. Using an efficient and scalable parallel implementation, we train a T-DSN to discriminate standard three-state monophones in the TIMIT database. The T-DSN outperforms an alternative pretrained Deep Neural Network (DNN) architecture in frame-level classification (both state and phone) and in the cross-entropy measure. For continuous phonetic recognition, T-DSN performs equivalently to a DNN but without the need for a hard-to-scale, sequential fine-tuning step.

***Index Terms*—** deep learning, higher-order statistics, tensors, stacking model, phonetic classification and recognition

1. INTRODUCTION

Recently, a deep classification architecture built upon blocks of single-hidden-layer neural networks (SHLNN) was proposed in [1, 2]. It was named the Deep Convex Network since the lower-layer weights connecting the input and hidden layers in the SHLNNs may be initialized effectively with a restricted Boltzmann machine and the learning of the upper-layer weights can then be formulated as a convex optimization problem with a closed-form solution. The model can also be, perhaps more appropriately, named the Deep Stacking Network (DSN) when we emphasize the mechanism in this network for building up the deep architecture that shares the same philosophy of "stacked generalization" [3]. The new deep architecture presented in this paper, which we call the Tensor Deep Stacking Network (T-DSN), improves and extends the DSN architecture in two significant ways. First, the information about higher-order statistics in the data, which was not represented in DSN, is now embedded into T-DSN via a bilinear model with a tensor representation of three-way interactions of the network weights. Second, while the T-DSN retains the same linear-nonlinear interleaving structure as DSN in building up the deep architecture, it shifts the major estimation problem in the learning algorithm from the outer non-convex optimization component to the inner convex one with a closed-form solution.

One major motivation for developing the DSN is the lack of scalability and parallelization in the learning algorithms for the Deep Neural Network (DNN) [4, 5]. In [1, 6], it was shown that all computational steps of the learning algorithm for DSN are batch-mode based, and are thus amenable to parallel implementation on a cluster of CPU/GPU nodes. The same computational advantage is retained for T-DSN introduced in this paper; we are able to parallelize the gradient computation and function evaluation to permit scaling to larger training sets. Section 4 discusses how the T-DSN generalizes the DSN. Experimental evaluation for phone classification and recognition is presented in Section 5, and we close with a discussion of the model and future work in Section 6.

*This work was conducted during an internship at Microsoft Research.

2. THE TENSOR DEEP STACKING NETWORK

In this section, we first briefly review the DSN, and then describe the general architecture of the T-DSN and its properties.

2.1. A Review of DSN

The Deep Stacking Network (DSN) is a scalable deep architecture amenable to parallel weight learning [1]. The DSN is trained in a supervised, block-wise fashion, without the need for back-propagation over all blocks. The DSN blocks are stacked to form the overall deep network.

Each DSN block is a SHLNN. It has an upper-layer weight matrix $\mathbf{U}$ that connects the sigmoidal nonlinear hidden layer $\mathbf{h}$ to the linear output layer $\mathbf{y}$, and a lower-layer weight matrix $\mathbf{W}$ that links the input and hidden layers. Let the target vectors $\mathbf{t}$ be stacked into the columns of $\mathbf{T}$ and assume the lower-layer weights $\mathbf{W}$ are known. Then learning the upper-layer weight matrix $\mathbf{U}$ can be formulated as a convex optimization problem and has a closed-form solution:

$$\mathbf{U}^T = \mathbf{T}\mathbf{H}^\dagger, \tag{1}$$

where

$$\mathbf{H}^\dagger = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}, \qquad \mathbf{H} = \sigma(\mathbf{W}^T\mathbf{X}), \tag{2}$$

and $\mathbf{X}$ is the data matrix, whose columns are the input vectors to the SHLNN. In the lowest layer, $\mathbf{X}$ contains only the raw input data, while in higher layers the input data may be concatenated with one or more output representations from the previous layers. The lower-layer weight matrix $\mathbf{W}$ can be optimized using an accelerated gradient descent [7, 8] algorithm to minimize the squared error $f = \|\mathbf{U}^T\mathbf{H} - \mathbf{Y}\|_F$. Embedding the solution of Eq.1 into the objective and deriving the gradient, we obtain

$$\nabla_{\mathbf{W}} f = 2\mathbf{X}\left[\mathbf{H}^T \circ (\mathbf{1} - \mathbf{H}^T) \circ \left[\mathbf{H}^\dagger(\mathbf{H}\mathbf{T}^T)(\mathbf{T}\mathbf{H}^\dagger) - \mathbf{T}^T(\mathbf{T}\mathbf{H}^\dagger)\right]\right], \tag{3}$$

where $\mathbf{1}$ is the matrix of all ones and $\circ$ denotes element-wise multiplication.

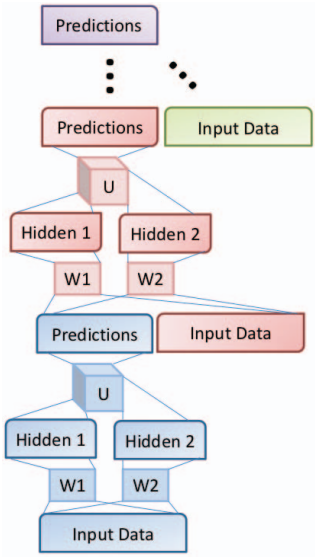

Fig. 1. Example T-DSN architecture with two complete blocks.

2.2. An Architectural Overview of T-DSN

In Fig. 1, we illustrate the modular architecture of a T-DSN, where two complete blocks, one in red and one in blue, are stacked one on another. The stacking operation of the T-DSN is exactly the same as that for the DSN described in [1]. Unlike the DSN, however, each T-DSN block has two sets of lower-layer weight matrices $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$. They connect the input layer with two parallel sigmoidal hidden layers "Hidden 1" and "Hidden 2" in Fig. 1. Each T-DSN block also contains a three-way, upper-layer weight tensor $\mathcal{U}$ that connects the two hidden layers with the output layer.

Note if the T-DSN is used for regression or for classification, then the basic architecture shown in Fig. 1 is sufficient. However, if T-DSN is to be interfaced with a hidden Markov model (HMM) for structured prediction such as continuous phonetic or word recognition as is the focus of this paper, it is desirable to convert the final output in Fig. 1 into posterior probabilities via an additional softmax layer. Our experiments reported in Section 4 are obtained with a softmax layer added to the top of Fig. 1.

2.3. Bilinear Predictions from Two Parallel Hidden Layers

We now elaborate on the key aspect of the T-DSN: modeling three-way interactions among the two parallel hidden layers and the output prediction layer in each T-DSN block. In place of the DSN's linear mapping from the hidden units $\mathbf{h}$ to the output units $\mathbf{y}$, the T-DSN uses a bilinear relationship from the two hidden representations, $\mathbf{h}_{(1)}$ and $\mathbf{h}_{(2)}$. The upper layer is thus parameterized by a weight tensor, $\mathcal{U}$. Formally, the predictions $\mathbf{y}$ from a T-DSN are defined as

$$\mathbf{y} = \begin{bmatrix} \mathbf{h}_{(1)}{}^T \mathbf{U}_1 \mathbf{h}_{(2)} \\ \vdots \\ \mathbf{h}_{(1)}{}^T \mathbf{U}_C \mathbf{h}_{(2)} \end{bmatrix}, \quad \text{where} \quad \mathbf{h}_{(\mathbf{j})} = \sigma(\mathbf{W}_{(\mathbf{j})}{}^T \mathbf{x})^T \qquad (4)$$

and $\mathbf{U}_k \in \mathbb{R}^{L_1 \times L_2}$ are class-dependent matrix slices of the tensor $\mathcal{U}$. The connection to the DSN can be illuminated by some changes in notation. First, note that $\mathbf{h}_{(\mathbf{1})}{}^T \mathbf{U}_k \mathbf{h}_{(2)} = \text{vec}(\mathbf{U}_k)^T \text{vec}(h_{(1)} h_{(2)}^T) = \sum_{i=1}^{L_1} \sum_{j=1}^{L_2} u_{kij} h_{(1)i} h_{(2)j}$; that is, it is a weighted sum of the products between each pair of elements of $\mathbf{h}_{(1)}$ and $\mathbf{h}_{(2)}$. If we let $\tilde{\mathbf{h}} \in \mathbb{R}^{L_1 L_2}$ denote a vector containing all pairs of products between $\mathbf{h}_{(1)}$ and $\mathbf{h}_{(2)}$, and $\tilde{\mathbf{u}}_k$ the vector of corresponding weights, then $\mathbf{h}_{(1)}^T \mathbf{U}_k \mathbf{h}_{(2)} = \tilde{\mathbf{u}}_k^T \tilde{\mathbf{h}}$. Stacking the $\tilde{\mathbf{u}}_\mathbf{k}$ into a matrix $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_\mathbf{1}\ \tilde{\mathbf{u}}_2\ \cdots\ \tilde{\mathbf{u}}_\mathbf{C}]$, and $\tilde{\mathbf{h}}_\mathbf{n}$ for each data point into a matrix $\tilde{\mathbf{H}} = [\tilde{\mathbf{h}}_\mathbf{1}\ \tilde{\mathbf{h}}_2\ \cdots\ \tilde{\mathbf{h}}_\mathbf{N}]$, it follows that $\mathbf{y} = \tilde{\mathbf{U}}\tilde{\mathbf{h}}$ and, in matrix form, that $\mathbf{Y} = \tilde{\mathbf{U}}\tilde{\mathbf{H}}$.

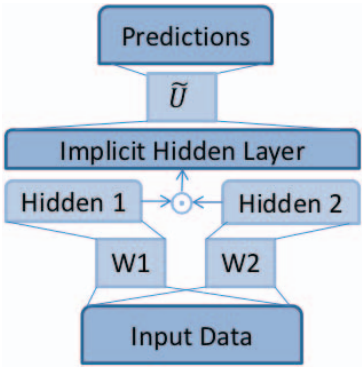

Fig. 2. Equivalent architecture to the bottom (blue) block of T-DSN in Fig. 1 where the tensor is unfolded into a large matrix.

This leads to the same prediction equation as in DSN, but with a transformed hidden representation $\tilde{\mathbf{h}}$ that contains multiplicative interactions between $\mathbf{h}_{(1)}$ and $\mathbf{h}_{(2)}$, permitting second-order statistics of the input data to be included in an abstract and parsimonious manner. Fig. 2 gives an equivalent architecture of the bottom block in Fig. 1, illustrating how the two hidden layers are expanded into an implicit hidden layer with all pairwise products. The relationship between the matrices of explicit (low-dimensional) hidden units, $\mathbf{H}_{(i)} = \sigma(\mathbf{W}_{(\mathbf{i})}{}^T \mathbf{X})$, and matrix of implicit (high-dimensional) hidden units $\tilde{\mathbf{H}}$, is given by

$$\tilde{\mathbf{H}} = \mathbf{H}_{(1)} \odot \mathbf{H}_{(2)}.$$

By $\odot$ we denote the Khatri-Rao product [9], which is a column-wise Kronecker product, and achieves the effect of multiplying all pairs of values within each column.

3. LEARNING T-DSN PARAMETERS

Due to the equivalence of the architectures shown in Fig.1 and Fig. 2, learning $\tilde{\mathbf{U}}$, the unfolded representation of tensor $\mathcal{U}$, given the implicit hidden layer's output becomes the same as that in the DSN. Since the implicit hidden layer's output is a deterministic function of the lower-layer weight matrices, we only need to determine $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$ to train a T-DSN block.

To train a T-DSN block using first order methods, we need to calculate the gradients of the squared error objective function with respect to $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$. These gradients have a similar form to that of the DSN (Eq. 3), but must be modified to account for the Khatri-Rao product. Using the fact that $\partial(\mathbf{H}_{(1)} \odot \mathbf{H}_{(2)}) = (\partial \mathbf{H}_{(1)}) \odot \mathbf{H}_{(2)} + \mathbf{H}_{(1)} \odot (\partial \mathbf{H}_{(2)})$, and letting $\mathbf{\Theta}$ denote $\tilde{\mathbf{H}}^\dagger (\tilde{\mathbf{H}} \mathbf{T}^T)(\mathbf{T}\tilde{\mathbf{H}}^\dagger) - \mathbf{T}^T (\mathbf{T}\tilde{\mathbf{H}}^\dagger)$ we can modify the derivation in [1] to obtain the following gradients:

$$\nabla_{\mathbf{W}_1} f = 2\mathbf{X}\left[\mathbf{H}_{(1)}^T \circ (1 - \mathbf{H}_{(1)}^T) \circ \mathbf{\Psi}_{(1)}\right], \qquad (5)$$

and

$$\nabla_{\mathbf{W}_2} f = 2\mathbf{X}\left[\mathbf{H}_{(2)}^T \circ (1 - \mathbf{H}_{(2)}^T) \circ \mathbf{\Psi}_{(2)}\right] \qquad (6)$$

where $\Psi_{(1)ij} = \sum_{k=1}^{L_2} h_{(2)kj}\Theta_{((i-1)L_2+k),j}$ and $\Psi_{(2)ij} = \sum_{k=1}^{L_1} h_{(1)kj}\Theta_{((k-1)L_1+i),j}$. The $\mathbf{\Psi}$ matrices have the effect of bridging the high dimensional representation used in $\mathbf{\Theta}$ with the low dimensional representation in $\mathbf{H}_{(i)}$, and are needed due to the Khatri-Rao product. In contrast, the DSN has only a single hidden representation $\mathbf{H}_{(1)}$, which has the same dimension as $\tilde{\mathbf{H}}$, so $\mathbf{\Theta}$ is used directly in place of $\mathbf{\Psi}$.

With the above gradients, we optimize the objective via the L-BFGS method using the Poblano optimization toolbox [10]. Typically, a T-DSN block is trained with 10 to 15 iterations and with up to 7 line-search function evaluations per iteration. Weight matrices $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$ are initialized with random values in the range $[-1, 1]$.

From Eqs. 5 and 6, it is clear that the bulk of the gradient computation is in matrix operations, including matrix multiplies and element-wise matrix products. To speed up computation and to reduce the memory requirements, we have successfully parallelized these matrix operations to run on a CPU cluster. The ability to parallelize training in this manner is a key reason for the scalability of T-DSN training.

4. CONNECTIONS BETWEEN T-DSN AND DSN

The T-DSN can be reduced to a DSN by forcing one of the two parallel hidden unit sets in each T-DSN block to have size one. Although the DSN can be considered to be a special, extremely asymmetrical, case of T-DSN, we have found empirically that the more symmetric the numbers of the two sets of hidden units in the T-DSN are, the better the classification performance. We conjecture that this is due to the fact that the symmetric case maximizes the ratio of implicit feature dimension over explicit feature dimension, and thus makes the best use of the parameters. The key advantage of the non-degenerated T-DSN (i.e., roughly equal number of hidden units in each set) over the degenerated one (i.e., DSN) is the new ability to capture higher-order feature interactions or correlations. In addition, the T-DSN typically has only 50 to 100 units in each of the two sets, which is substantially smaller than the size of the hidden layer in a DSN with a typical size of 3000. Hence, the parameter balance is drastically shifted from the lower-layer weights toward the upper-layer ones, the latter being much easier to optimize due to the closed-form solution. The significantly smaller hidden representation size in T-DSN has the further strength of bottlenecking the data. This aids the "stackability" in the deep architecture by providing stacking flexibility. That is, one can concatenate the raw data not only with the module's prediction layer, but instead with $\mathbf{h}_{(1)}$ and $\mathbf{h}_{(2)}$, or even concatenating all these three sets without dramatically increasing the input dimension in the higher-level blocks. Such flexibility is difficult to achieve by the DSN.

The T-DSN objective is identical to that of the DSN, with $\mathbf{U}$ replaced by $\tilde{\mathbf{U}}$ and $\mathbf{H}$ replaced by $\tilde{\mathbf{H}}$. This similarity allows us to exploit much of the learning machinery developed for the DSN to train the T-DSN, including the parallel implementation discussed previously.

5. EXPERIMENTS

5.1. Experimental setup

We now report our experiments using the popular speech database TIMIT. The speech data is analyzed using a 25-ms Hamming window with a 10-ms fixed frame rate. We represent the speech using first- to 12th-order Mel frequency cepstral coefficients (MFCCs) and energy, along with their first and second temporal derivatives. The data are normalized to have zero mean and unit variance. All experiments used a context window of 11 frames. This gives a total of $39 \cdot 11 = 429$ elements in each feature vector as the raw input to T-DSN. For the prediction at each layer of the T-DSN, we used 183 target class labels (i.e., three states for each of the 61 phones), which we call "phone states," with a one-hot encoding.

We use the standard TIMIT data sets. The training set consists of 462 speakers, with all SA sentences removed. The total number of frames in the training data is 1,124,589. The development set contains 50 speakers, with a total of 122,488 frames, and is used for tuning. Results are reported using the standard 24-speaker core test set consisting of 192 sentences with 7,333 phone tokens and 57,920 frames.

The DSN and T-DSN are both trained in batch-mode, which is practical due to our parallelization of their learning algorithms.

5.2. Experimental results

The results reported in this section are obtained using the main T-DSN architecture illustrated in Fig.1, where the number of stacking blocks is 13, and an additional hidden layer and softmax layer are added to the top of this T-DSN for computing frame-level state posterior probabilities. The two sets of hidden units in each block contain an equal number of 70 nodes.

In Table 1, we show the performance of the T-DSN (first row) in terms of four measures: 1) Frame-level state error rate (Fr State Err); 2) cross entropy (Cr-Ent) in nats; 3) frame-level phone classification error rate (Fr Ph Err), and 4) continuous phonetic recognition error rate (Ph Rec Err). For the state error rate, the total number of classes is 183. In computing the frame-level phone classification error rate, we map the 183 states to 61 phone classes which are then collapsed into 39 classes using a standard phone mapping. Cross entropy is the average of log posterior probabilities over all frames in the test set, computed from the softmax layer. The higher the cross entropy, the better the performance. To obtain the continuous phonetic recognition error rate, we used a phone decoder, which applies dynamic programming over each of the full test sentences. A standard bigram phone language model is used during decoding. The results in Table 1 demonstrate superior performance of T-DSN over DSN in all four measures (Row 1 vs. Row 2).

It is also interesting to compare the performance of T-DSN with two other deep architectures, the conventional DBN-initialized DNN trained with the frame-level cross entropy criterion (labeled as "DNN" in Row 3 in Table 1) and that trained with the full-sequence maximum mutual information (MMI) criterion [11] (final row in Table 1). The original motivation of this work was to make the learning of deep networks scalable by replacing stochastic gradient descent algorithm for fine tuning with the parallelizable batch-mode learning. As both DSN and T-DSN do "fine tuning" only within the block rather than through the entire deep network as carried out for DNN, we expected at best a matching performance to DNN. Surprisingly, while DSN has not matched the low phonetic recognition error rate achieved by DNN, T-DSN produces a slightly lower error rate (22.8% vs. 22.9%). And more surprisingly, for the measures of frame-level error rates and cross entropy, both DSN and T-DSN outperform DNN and even MMI-DNN.

Fig. 3 illustrates the performance of a single T-DSN block, with hidden unit sizes of $L_1 = L_2 = \{70, 80, 98\}$. $L_1 = L_2 = 98$ marks the break-even point, when the T-DSN has the same number of overall parameters (lower and upper layer weights) as a DSN with 3000 hidden units, although the T-DSN still has far fewer *free* (lower layer) parameters. Despite random initialization, the learning

Networks	Fr State Err	Cr-Ent	Fr Ph Err	Ph Rec Err
T-DSN	40.9%	-2.02	21.0%	22.8%
DSN	41.8%	-2.16	22.9%	24.6 %
DNN	45.0%	-2.28	23.5%	22.9 %
MMI-DNN	43.0%	-2.20	23.0%	22.2 %

Table 1. Performance comparison between the T-DSN and three other deep architectures on the TIMIT core test set. The reported performance measures are: frame-level state error rate, cross entropy, frame-level phone classification error rate, and continuous phonetic recognition error rate.

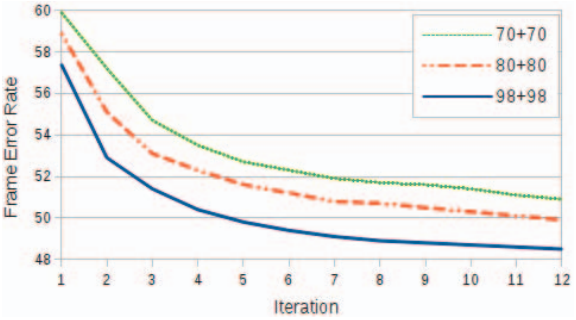

Fig. 3. Single layer T-DSN test set performance, by hidden unit size.

reliably converges to a good solution, suggesting the absence of excessive local optima. In contrast, initializing the DSN with random weights leads to poor and unpredictable single-layer performance, nearly always above 60% frame error rate. This robustness to initialization is an important practical advantage of the T-DSN.

6. DISCUSSION AND CONCLUSION

A new architecture of T-DSN for deep learning is presented, generalizing the earlier DSN architecture. The principal novelty is to split the original large hidden layer (in each block) into two smaller ones, and bilinearly map them to the predictions so as to capture higher order interactions between features. T-DSN retains the computational advantage of DSN in parallelism and scalability while learning the weight parameters. Note the parallelism in learning for T-DSN can be implemented either in a CPU cluster (as carried out in the current study) or in a GPU cluster. A single GPU parallelization speed up over CPU can be between 10-100 times but CPU programming is much easier.

Modeling covariance structure directly on raw speech data, rather than on the more compact hidden feature layers as achieved in T-DSN, was previously proposed in an architecture called mcRBM [12]. One key distinction is the different domains in which the higher-order structure is represented, one in the data and another in the hidden units. In addition, mcRBM cannot be extended to deeper layers due to the model and learning complexity, and it has to rely on factorization to reduce the cubic growth in size of the weight parameters. Factorization incurs very high computational cost, which, together with the need for high cost of Hybrid Monte Carlo in learning, makes it impossible to scale up to large data sets. Fortunately, these difficulties are removed in T-DSN. Specifically, the same interleaving of linear-nonlinear layers inherited from DSN makes it straightforward to stack up deeper layers, and the closed-form solution for the upper-layer weights enables much more efficient training. Because of the relatively smaller sizes in the multiplicative hidden layers, no factorization is needed for the T-DSN's weights.

Compared with our earlier DSN architecture, two further advantages of T-DSN are 1) a potential extension to incorporate speaker and/or environmental factor-gating (by training one of the hidden layers to encode speaker or environmental factors), and 2) a new stacking mechanism where compact dual hidden representations can be concatenated with input data. Our preliminary work on both of these fronts is promising. With the parallelized implementation of T-DSN already in place, we expect meaningful improvements in real-world speech recognition tasks. Further, encouraged by our recent results with DNN and DSN in applications of speech understanding [13] and in speech attribute detection [14], we expect greater success with the use of T-DSN in these and other applications.

7. REFERENCES

[1] L. Deng and D. Yu, "Deep convex networks: A scalable architecture for speech pattern classification," in *Proc. Interspeech*, August 2011.

[2] L. Deng and D. Yu, "Deep convex networks for image and speech classification," in *ICML Workshop on Learning Architectures*, June 2011.

[3] D.H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[4] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, January 2012.

[5] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, January 2012.

[6] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Proc. ICASSP*, 2012.

[7] D. Yu and L. Deng, "Accelerated parallelizable neural network learning algorithm for speech recognition," in *Proc. Interspeech*, August 2011.

[8] D. Yu and L. Deng, "Efficient and effective algorithms for training single-hidden-layer neural networks," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 554–5580, April 2012.

[9] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, September 2009.

[10] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Poblano v1.0: A matlab toolbox for gradient-based optimization," Tech. Rep. SAND2010-1422, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Mar. 2010.

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

[12] G. Dahl, M. Ranzato, A. Mohamed, and G. Hinton, "Phone recognition with the mean-covariance restricted bolzmann machine," in *Proc. NIPS*, December 2010.

[13] G. Tur, L. Deng, D. Hakkani-Tur, and X. He, "Toward deeper understanding: Deep convex networks for semantic utterance classification," in *Proc. ICASSP*, 2012.

[14] D. Yu, S. Siniscalchi, L. Deng, and C. Lee, "Boosting attribute and phone estimation accuracies with deep neural networks for detection-based speech recognition," in *Proc. ICASSP*, 2012.