Deep Convex Net: A Scalable Architecture for Speech Pattern Classification

Li Deng and Dong Yu

Microsoft Research, Redmond, WA, USA

{deng, dongyu}@microsoft.com

Abstract

We recently developed context-dependent DNN-HMM (Deep-Neural-Net/Hidden-Markov-Model) for large-vocabulary speech recognition. While achieving impressive recognition error rate reduction, we face the insurmountable problem of scalability in dealing with virtually unlimited amount of training data available nowadays. To overcome the scalability challenge, we have designed the deep convex network (DCN) architecture. The learning problem in DCN is convex within each module. Additional structure-exploited fine tuning further improves the quality of DCN. The full learning in DCN is batch-mode based instead of stochastic, naturally lending it amenable to parallel training that can be distributed over many machines. Experimental results on both MNIST and TIMIT tasks evaluated thus far demonstrate superior performance of DCN over the DBN (Deep Belief Network) counterpart that forms the basis of the DNN. The superiority is reflected not only in training scalability and CPU-only computation, but more importantly in classification accuracy in both tasks.

Index Terms: deep learning, scalability, convex optimization, neural network, deep belief network, phone state classification, batch-mode training, parallel computing

1. Introduction

Automatic speech recognition (ASR) has been the subject of a significant amount of research and commercial development in recent years. Recent research in ASR has explored deep, layered architectures, motivated partly by the desire to capitalize on some analogous properties in the human speech generation and perception systems; e.g., [1][2]. In these studies, learning of model parameters has been one of the most prominent and difficult problems. In parallel with the development in ASR research, recent progresses made in learning methods from neural network research has also ignited interest in exploration of deep-structured models; e.g. [3]. One particular advance is the development of effective learning techniques for Deep Belief Networks (DBNs), which are densely connected, directed belief networks with many hidden layers. In general, DBNs can be considered as a complex nonlinear feature extractor with many layers of hidden units and at least one layer of visible units, where each layer of hidden units learns to represent features that capture higher order correlations in the original input data [3]-[8]

While DBNs have been shown to be extremely powerful in connection with performing recognition and classification tasks including speech recognition [4]-[7], training DBNs has proven to be more difficult computationally. In particular, conventional techniques for training DBNs involve the utilization of a stochastic gradient descent learning algorithm. Although stochastic gradient descent has been shown to be powerful for fine-tuning weights assigned to a DBN, such learning algorithm is extremely difficult to parallelize across machines, causing learning at large scale to be difficult. It has been possible to use one single, very powerful GPU machine to train a DNN-HMM for speech recognizers with dozens to a few hundreds of hours of speech training data with remarkable results [5]. To scale up this success with thousands or more hours of training data, we have been encountering seemingly insurmountable difficulty with the current DNN architecture used in our work in the recent past [5][6][7][8].

The main thrust of the research reported in this paper is a new deep learning architecture, referred to as Deep Convex Network (DCN), which squarely attacks the learning scalability problem. The organization of this paper is as follows. In Section 2, we provide an overview of the DCN architecture, and focus on how it integrates some key ideas from DBN, boosting, and extreme learning machine. In Section 3, an accelerated optimization algorithm we developed recently is outlined, which "fine-tunes" the DCN weights capitalizing on the structure in each module of the DCN. We show experimental results on static classification tasks defined on MNIST (image) and TIMIT (speech), with the accuracy of DCN exceeding that of DBN on both tasks.

2. The DCN Architecture

A DCN includes a variable number of layered modules, wherein each module is a specialized neural network consisting of a single hidden layer and two trainable sets of weights. More particularly, the lowest module in the DCN comprises a first linear layer with a set of linear input units, a non-linear layer with a set of non-linear hidden units, and a second linear layer with a set of linear output units. For instance, if the DCN is utilized in connection with recognizing an image, the input units can correspond to a number of pixels (or extracted features) in the image, and can be assigned values based at least in part upon intensity values, RGB values, or the like corresponding to the respective pixels. If the DCN is utilized in connection with speech recognition, the set of input units may correspond to samples of speech waveform, or the extracted features from speech waveforms, such as power spectra or cepstral coefficients. Note the use of speech waveform as the raw features to a speech recognizer is not a crazy idea. An early study for an HMM-like system (i.e., the hidden filter) that models speech waveform directly as the observation can be found in [9]. And many years later the use of more powerful Restricted Boltzmann Machine (RBM) overcomes some difficulty encountered earlier [10].

The hidden layer of the *lowest module* of a DCN comprises a set of non-linear units that are mapped to the input units by way of a first, lower-layer weight matrix, which we denote by $\boldsymbol{W}$. For instance, the weight matrix may comprise a plurality of randomly generated values between zero and one, or the weights of an RBM trained separately. The non-linear units may be sigmoidal units that are configured to perform non-linear operations on weighted outputs from the input units (weighted in accordance with the first weight matrix $\boldsymbol{W}$).

The second, *linear* layer in any module of a DCN includes a set of output units that are representative of the targets of classification. For instance, if the DCN is configured to perform digit recognition (e.g., the digits 1-10), then the

plurality of output units may be representative of the values 1, 2, 3, and so forth up to 10 with a 0-1 coding scheme. If the DCN is configured to perform ASR, then the output units may be representative of phones, HMM states of phones, or context-dependent HMM states of phones in a way that is similar to [5][6]. The non-linear units in each module of the DCN may be mapped to a set of the linear output units by way of a second, upper-layer weight matrix, which we denote by $\boldsymbol{U}$. This second weight matrix can be learned by way of a batch learning process, such that learning can be undertaken in parallel. Convex optimization can be employed in connection with learning $\boldsymbol{U}$. For instance, $\boldsymbol{U}$ can be learned based at least in part upon the first weight matrix $\boldsymbol{W}$, values of the coded classification targets, and values of the input units.

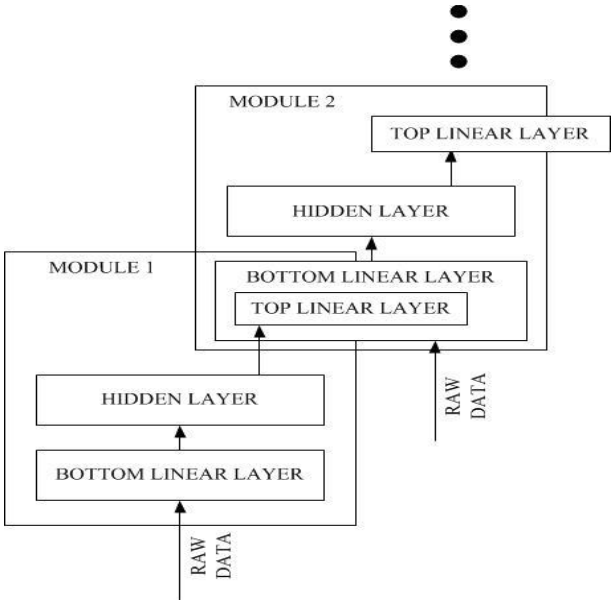

Fig.1: *Block diagram showing two of many modules in a DCN and their connection; note the overlapping of two linear layers in the two adjacent modules*

As indicated above, the DCN includes a set of serially connected, overlapping, and layered modules, wherein each module includes the aforementioned three layers -- a first linear layer that includes a set of linear input units whose number equals the dimensionality of the input features, a hidden layer that comprises a set of non-linear units whose number is a tunable hyper-parameter, and a second linear layer that comprises a plurality of linear output units whose number equals that of the target classification classes (e.g., the total number of context-dependent phones clustered by a decision tree used in [5][6]). The modules are referred to herein as being layered because the output units of a lower module are a subset of the input units of an adjacent higher module in the DCN. More specifically, in a second module that is directly above the lowest module in the DCN, the input units can include the output units of the lower module(s). The input units can additionally include the raw training data – in other words, the output units of the lowest module can be appended to the input units in the second module, such that the input units of the second module also include the output units of the lowest module. A block diagram showing two of many modules in a DCN and their connection is shown in Fig. 1. The sharing or overlapping of the two adjacent modules in a DCN is represented explicitly in the overlapping portion of the two large boxes labeled as MODULE 1 and MODULE 2, respectively, in Fig. 1. This particular way of serially connecting any adjacent modules in the DCN has been motivated partly by our earlier work on the deep-structured conditional random field [11].

In Fig. 2, we show information flow within each module of the DCN that is not at the lowest layer. The part of the input units in any non-bottom module corresponding to the raw training data can be mapped to the hidden units by the first weight matrix described earlier, denoted here in Fig. 2 as $\boldsymbol{W}_{\text{rbm}}$ (In our experiments reported in Section 4, the use of separately trained RBM to initialize these weights gave much better results than all other ways of initialization). The portion of the input units in the module corresponding to the output units of the lower module can be mapped to the common set of hidden units by a new weight matrix, which may be initialized by, for example, random numbers. We denote this set of weights by $\boldsymbol{W}_{\text{ran}}$ in Fig. 2. Thereafter, the aforementioned second weight matrix $\boldsymbol{U}$, which connects between the hidden units and the linear output units of this module, can be learned by way of convex optimization. This operation is represented by the box labeled with "Learning Component" in Fig. 2.

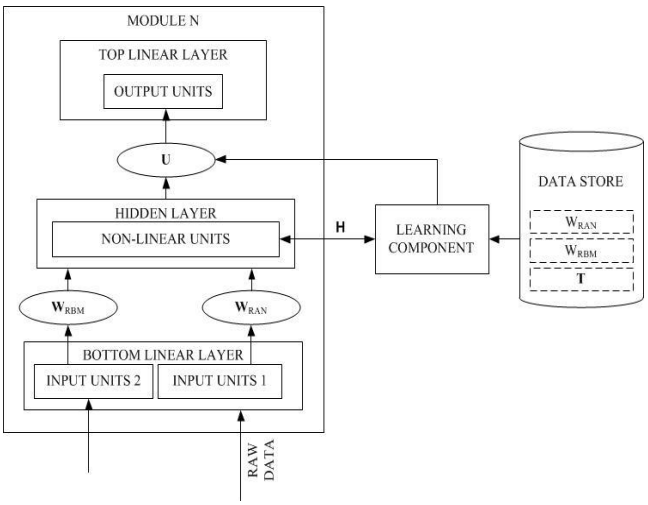

Fig. 2. *Information flow in one typical module, which is not at the lowest layer, of the DCN.*

The pattern discussed above of including output units in a lower module as a portion of the input units in an adjacent higher module in the DBN and thereafter learning a weight matrix that describes connection weights between hidden units and linear output units via convex optimization can continue for many modules -- e.g., tens to hundreds of modules in our experiments. A resultant learned DCN may then be deployed in connection with an automatic classification task such as frame-level speech phone or state classification. Connecting DCN's output to an HMM or any dynamic programming device enables continuous speech recognition. Details of this final step can be found in [5] and will not be dealt with in the remaining part of this paper.

3. DCN Fine Tuning in Batch Mode

Unlike DBN, the "fine tuning" algorithm of DCN weights we developed recently is confined within each module, rather than across all layers globally. It is batch-mode based, rather than stochastic; hence it is naturally parallelizable. Further, it makes direct use of the DCN structure where the strong constraint is imposed between the upper layer's weights, $\boldsymbol{U}$, and the lower layer's weights, $\boldsymbol{W}$, within the same module as the weighted pseudo-inverse:

$$\boldsymbol{U} = (\boldsymbol{H}\Lambda\boldsymbol{H}^T)^{-1}\boldsymbol{H}\Lambda\boldsymbol{T}^T. \quad (1)$$

Here, $\boldsymbol{H}$ is the output vectors of the hidden units:

$$\boldsymbol{H} = \sigma(\boldsymbol{W}^T \boldsymbol{x}\ ), \qquad (2)$$

Λ is the weight matrix constructed to direct the optimization's search direction, and $\boldsymbol{T}$ is the classification target vectors.

We use the batch-mode gradient descent to fine tune $\boldsymbol{W}$, where the gradient of the mean square error, $\boldsymbol{E}$, after constraint (1) is imposed is given by

$$\frac{\partial E}{\partial \boldsymbol{W}} = 2\boldsymbol{X}\Big[\boldsymbol{H}^{\boldsymbol{T}} \circ (\boldsymbol{1} - \boldsymbol{H})^{\boldsymbol{T}} \circ [\boldsymbol{H}^{+}(\boldsymbol{H}\Lambda\boldsymbol{T}^T)(\boldsymbol{T}\boldsymbol{H}^{+}) - \Lambda\boldsymbol{T}^T(\boldsymbol{T}\boldsymbol{H}^{+})]\Big],$$

and $\boldsymbol{H}^{+} = \Lambda\boldsymbol{H}^T(\boldsymbol{H}\Lambda\boldsymbol{H}^T)^{-1}$.

More detail of this DCN fine tuning algorithm is provided in [13].

4. Experimental Evaluation

4.1. NMIST experiments and results

Comprehensive experiments have been conducted to evaluate the performance of the DCN architecture and the related learning algorithms on the benchmark MNIST database; see [12] for detail of this task. In brief, the MNIST consists of binary images of handwritten digits, and is one of the most common classification tasks for evaluating machine learning algorithms. We only briefly summarize our strong results on MNIST here in Table 1.

Table 1*: Classification error rate comparison: DBN vs. DCN*

DBN [3] (Hinton's)	DBN (MSR's)	DCN (Fine-tuning)	DCN (no Fine-tuning)	Shallow (D)CN (Fine-tuned single layer)
1.20%	1.06%	0.83%	0.95%	1.10%

4.2. TIMIT experiments

We now focus on our more recent experiments where we apply the same DCNs and the related learning algorithms developed on the MNIST task to the speech database of TIMIT. Standard MFCC feature was used, but with a longer than usual context window of 11 frames. This gives rise to a total of 39*11=429 elements in each feature vector, which we call a "super-frame", as the input to each module of the DCN. For the DCN output, we used 183 target class labels as "phone states". The 183 target labels correspond to all states in the 61 phone-like units defined in TIMIT.

The standard training set of TIMIT consisting of 462 speakers was used for training the DCN. The total number of super-frames in the training data is about 1.12 million. The standard development set of 50 speakers, with a total of 122,488 super-frames, was used for cross validation. Results are reported using the standard 24-speaker core test set consisting of 192 sentences with 7,333 phone tokens and 57,920 super-frames.

The algorithms presented in this paper all are batch-mode based. This is because, as an example of convex optimization with the global optimum, the pseudo-inverse is carried out necessarily involving the full training set. However, in our experiments where the full training set of TIMIT is represented by a very large 429 by 1.12M matrix, the various batch-mode matrix multiplications required by the algorithms easily cause a single computer to run out of memory. (We had not implemented our learning algorithms over parallel machines at the time of carrying out the reported experiments here). To overcome the CPU memory limitation problem, we block the training data into many mini-batches, and use mini-batch training instead of full batch training. After the final mini-batch data are consumed in each training epoch, we then use a routine for block matrix multiplication and inversion, which incurs some undesirable but unavoidable waste of computation with a single CPU, to combine the full training data in implementing the estimation formula of Eq. (1) in order to achieve approximate effect of batch-mode training to our best ability.

4.3. TIMIT results

The DCN, as well as DBN, has the strength mainly as a static pattern classifier. HMM or dynamic programing is a convenient tool to help port the strength of a static classifier to handle dynamic patterns, as we recently demonstrated with DNN [5][6]. (We are nevertheless clearly aware that the unique elasticity of temporal dynamic of speech as explained in [1] would require temporally-correlated models better than HMM for the ultimate success of ASR, and integrating such a model with the DCN to form the coherent dynamic DCN is by itself a more challenging new research beyond the scope of this paper.) Therefore, as our first step of experimentation, we focus on evaluation of the static classification ability of DCN here. To this end, we choose frame-level phone-state classification error rate as the main evaluation criterion. In this case, we have a total of 183 state classes, three for each of the 61 phone labels defined in the TIMIT training set. The actual state labels are obtained by HMM forced alignment. We also show frame-level phone classification error rates, when the errors in the state within the same phone are not counted, for 61 phone classes.

The results in Table 2 are obtained by a typical run of the DCN program when 6,000 hidden units are used in each module of DCN, where "X (Y)" in the first column denotes the X^{th} layer of DCN (counted from bottom up) and Y^{th} epoch in the fine-tuning optimization. The hyper-parameters are tuned using the development set defined in TIMIT. We used a single-hidden-layer RBM that was trained in the same way as in [4][5] to initialize weights $\boldsymbol{W}$ at the lowest module of the DCN before applying fine tuning as described in Section 3. We have found empirically that if random noise is used for the initialization, then the error rate becomes at least 30% relative higher than presented in Table 2.

Fine-tuned weights from lower modules are used to initialize the weights at higher modules. Then, they are appended with random weights associated with the output units from the immediately lower module before fine tuning at the current module.

Table 2. *Frame-level classification error rates of phones (61 classes) and states (183 classes) as a function of the number of stacked DCN modules; RBM is used to initialize lowest-level network weights.*

Layer (Epoch)	Train State Err %	Dev. State Err %	Test Phone Err %	Test State Err %
1 (1)	27.19	49.50	39.18	49.83
...	...	...	...	...
1 (8)	21.20	46.00	36.12	46.30
2 (1)	13.01	44.44	34.87	44.88
3 (1)	7.96	44.30	34.64	44.70
4 (1)	5.14	44.22	34.67	44.65
5 (1)	3.51	44.11	34.56	44.53
6 (1)	2.57	44.25	34.83	44.70
7 (1)	1.95	44.25	34.74	44.69

The most notable observation from Table 2 is that as the layers gradually add up, the error rates for training, development, and core test sets continue to drop until over-fitting occurs at Layer 6 in this example. There has been very little published work on frame-level phone or phone state classification. The closest work we have been able to find reported over 70% phone state error rate with an easier 132 phone state classes (than our 183 state classes) but with a more difficult speech database. We ran the DBN system of [7] on the same TIMIT data and found the corresponding frame-level phone state error rate be to 45.04% (which gave 22% phonetic recognition error rate after running a decoder with a standard bi-gram phonetic "language" model as reported in [7]). This frame-level error rate achieved by DBN is slightly higher than the DCN's error rate of 44.53% shown in Table 2.

In Table 3 is a summary of the results, with different hyper-parameters than in Table 2. It shows the dependency of frame-level classification error rates on the number of hidden units, which is fixed for all modules of the DCN in our current implementation. We also fold the 61 classes in the original TIMIT label set into the standard 39 classes. The corresponding results are presented in Table 3 also. These results are obtained without the use of phone-bound state alignment. That is, there is no left-to-right constraint, and frame-level decision is made. These results are obtained also without any phone-level "language" model.

Table 3. *Frame-level classification percent error rates of phones (61 or folded 39 classes), and of phone states (183 classes) as a function of the size of hidden layer units in DCN.*

Size of Hidden Units	Frame-level Test Phone Err % (39 classes)	Frame-level Test Phone Err % (61 classes)	Frame-level Test State Err % (181 classes)
3000	27.11	35.97	46.08
4000	26.37	35.27	45.39
6000	25.44	34.12	44.24
7000	25.22	34.04	44.04

5. Summary and Conclusions

We recently developed a DNN-based architecture for large-vocabulary speech recognition. While achieving remarkable success with this approach, we face the scalability problem in practical applications, e.g. voice search. In this paper we present a novel DCN architecture aimed to enable scalability. Experimental results on both MNIST and TIMIT tasks demonstrate higher classification accuracy than DBN. The superiority of DCN over DBN is particularly strong in the MNIST task so long as we use a much deeper DCN than could be computationally afforded by the conventional DBN architecture and learning. While the basic module of the DCN reported in this paper is similar to the extreme learning machine in the literature (e.g., [14]), any simple or weak classifier can be embedded in the DCN architecture to make it stronger.

The future directions of our work include: 1) full exploration of the rich flexibility in the architecture and module type provided by the basic DCN framework presented in this paper; 2) addition of a dynamic programing based decoder on top of the final layer of the DCN to enable continuous phonetic or speech recognition; 3) learning (rather than tuning) of hyper-parameters in DCN; 4) development of speaker and environment adaptation techniques for DCN; and 5) development of a temporal DCN which integrates generative dynamic models of speech (e.g., [15][16]) with the DCN architecture presented in this paper.

6. Acknowledgements

We are grateful to many helpful discussions with, and valuable suggestions and encouragements by John Platt, Geoff Hinton, Dave Wecker, and Alex Acero. We also thank G.B. Huang for discussions on many possible basic modules of the DCN.

7. References

[1] L. Deng, D. Yu, and A. Acero. "Structured speech modeling," IEEE Trans. Audio, Speech & Language Proc., vol. 14, no. 5, pp. 1492-1504, September 2006.

[2] N. Morgan. "Deep and Wide: Multilayers in Automatic Speech Recognition," IEEE Trans. on Audio, Speech, and Language Processing, 2011 (in press).

[3] G. Hinton and R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks", *Science,* Vol. 313. no. 5786, pp. 504 – 507, 2006.

[4] A. Mohamed, G. Dahl, G. Hinton, "Deep belief networks for phone recognition", NIPS Workshop on Deep Learning for Speech Recognition and Related Applications, Dec. 2009.

[5] G. 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, 2011 (in press).

[6] D. Yu, L. Deng, and G. Dahl, "Roles of Pre-Training and Fine-Tuning in Context-Dependent DBN-HMMs for Real-World Speech Recognition," NIPS Workshop on Deep Learning and Unsupervised Feature Learning, December 2010.

[7] A. Mohamed, D. Yu, and L. Deng, "Investigation of Full-Sequence Training of Deep Belief Networks for Speech Recognition," in *Interspeech,* September 2010.

[8] L. Deng, M. Seltzer, D. Yu, A. Acero, A. Mohamed, and G. Hinton. "Binary Coding of Speech Spectrograms Using a Deep Auto-encoder," in *Interspeech,* Sept. 2010.

[9] H. Sheikhzadeh and L. Deng. "Waveform-Based Speech Recognition Using Hidden Filter Models: Parameter Selection and Sensitivity to Power Normalization, IEEE Trans. on Speech and Audio Processing, Vol. 2, pp. 80-91, 1994.

[10] N. Jaitly and G. Hinton. "Learning a Better Representation of Speech Sound Waves Using Restricted Boltzmann Machines," *in Proc. ICASSP*, 2011, Prague.

[11] D. Yu, S. Wang, and L. Deng. "Sequential Labeling Using Deep-Structured Conditional Random Fields," IEEE J. Selected Topics in Sig. Proc., Vol. 4(6), pp.965-973, Dec. 2010.

[12] Y. LeCun, L. Bottou, Y., Bengio, and P. Haffner "Gradient-Based Learning Applied to Document Recognition", *Proc. IEEE,* Vol. 86, pp. 2278-2324, 1998.

[13] D. Yu and L. Deng, "Accelerated Parallelizable Neural Networks Learning Algorithms for Speech Recognition," Proc. *Interspeech 2011,* accepted.

[14] G. B. Huang, Q-Y. Zhu, and C.K. Siew. "Extreme Learning Machine: Theory and Applications", Neurocomputing, vol. 70, pp. 489-501, 2006.

[15] J. Baker, et. al. "Research Developments and Directions in Speech Recognition and Understanding," *IEEE Sig. Proc. Mag.,* vol. 26, pp. 75-80, May 2009.

[16] L. Deng, "Computational Models for Speech Production," Chapter in *Computational Models of Speech Pattern Processing*, pp. 199-213, Springer, 1999.