Deep Convex Networks for Image and Speech Classification

Li Deng DENG@MICROSOFT.COM
Microsoft Research, Redmond, WA 98052, USA

Dong Yu DONGYU@MICROSOFT.COM
Microsoft Research, Redmond, WA 98052, USA

Abstract

To overcome the scalability challenge associated with Deep Belief Network (DBN), we have designed a novel deep learning architecture, deep convex network (DCN). The learning problem in DCN is convex within each layer. 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 handwriting image recognition task (MNIST) and on phone state classification (TIMIT) demonstrate superior performance of DCN over DBN not only in training efficiency but also in classification accuracy. DCN gives the error rate of *0.83%*, the lowest without the use of additional training data produced by elastic distortion. The corresponding error rate by the best DBN which we have carefully tuned is *1.06%*. On the TIMIT task, DCN also outperforms DBN but with a relatively smaller percentage so far.

1. Introduction

While Deep Belief Network (DBN) has been shown to be extremely powerful in connection with performing recognition and classification tasks including speech recognition and image classification [1]-[6], training DBN has proven to be more difficult computationally. In particular, conventional techniques for training DBN at the fine tuning phase involve the utilization of a stochastic gradient descent learning algorithm, which is extremely difficult to parallelize across machines. This makes learning at large scale practically impossible. For example, it has been possible to use one single, very powerful GPU machine to train DBN-based speech recognizers with dozens to a few hundreds of hours of speech training data with remarkable results. It is very difficult, however, to scale up this success with thousands or more hours of training data.

The goal 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. In this paper, we demonstrate the power of DCN in a benchmark image recognition task in both the algorithm's scalability and in classification accuracy in comparison with DBN.

2. The Architecture of Deep Convex Network

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.

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

Extended summary

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. 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.

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). 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.

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.

3. Fine Tuning for Deep Convex Network

Unlike DBN, the "fine tuning" algorithm of DCN weights that 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:

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

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

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

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

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

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

4. Experiments on Image Classification

We evaluated the DCN and the associated learning algorithms on the MNIST database of binary images of handwritten digits [7]. The digits have been size-normalized to fit in a 20x20 pixel box while preserving their aspect ratio and centered in a 28x28 image by computing and translating the center of mass of the pixels. The task is to classify each 28x28 image into one of the 10 digits. The MNIST training set is composed of 60,000 examples from approximately 250 writers. The test set is composed of 10,000 patterns. The sets of writers of the training set and test set are disjoint.

In our experiments, a small fraction of the training data are held as validation set to tune hyper parameters in the DCN. Unlike the speech classification experiments (see next Section), we found that the properties of the validation and the test sets in MNIST are strikingly similar to each other.

Figure 1 shows the basic architecture of the DCN used in the MNIST experiment described in this section. Only three modules are illustrated, with the number of units in each layer of each module indicated. Each arrow in the

figure corresponds to a weight matrix in the DCN. They may be either upper layer or lower layer weights in a DCN module. The input layer in the lowest module consists of 784 linear units, which is the number on pixels (28x28) in each raw image. Hidden layers with 3000 sigmoidal nonlinear units are shown here. The input layer in the higher modules of the DCN consists of 784 units appended by multiples of 10, where 10 (number of classes in the MNIST task) is the number of output units in each module of the DCN.

Figure 2 gives results of percentage error rate as a function of the layers (modules) in the DCN of Figure 1 and of the iteration number of the fine tuning algorithm applied to each layer. Note in the horizontal axis, "0" indicates the beginning of a new layer (see the illustrative vertical bars), and "1", "2", "3"… signifies the epoch number in the running of the batch-mode fine tuning algorithm described in Section 3. Note that immediately before the vertical bars, i.e., when a new layer was added, there tended to be large error reduction.

In this experiment, we used restricted Boltzmann machine (RBM) to initialize the weight matrix in the lowest layer/module of the DCN, which was then subject to fine tuning. When moving up to the second layer, the fine-tuned weight matrix was copied, mixed with a new weight matrix, initialized by random numbers, which corresponds to the output units from the lower layer. This combined weight matrix was then again subject to fine tuning in the second layer, producing a new set of outputs in the second layer. And this process was continued until the top layer of the DCN.

5. Experiments on Speech Classification

We now report our more recent experiments where similar kinds of DCN to Section 4 were applied to the most popular speech database of TIMIT. The speech data was analyzed using a 25-ms Hamming window with a 10-ms fixed frame rate. We represented the speech using first- to 12th-order Mel frequency cepstral coefficients (MFCCs) and energy, along with their first and second temporal derivatives. The data were normalized to have zero mean and unit variance. All experiments used a context window of 11 frames. This gives rise to a total of 39*11=429 elements in each feature vector, or a super-frame, as the input to the single-hidden-layer neural network. For the neural network output, we used 183 target class labels (i.e., three states for each of the 61 phones), coded in binary zero or one, which we call "phone states".

The standard training set of TIMIT consisting of 462 speakers was used, with all SA sentences removed, for training the neural network consisting of linear input and output units and sigmoid hidden units. 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 used here all are batch-mode based, since the pseudo-inverse is carried out necessarily involving the full training set. However, in our experiments where the full training set 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 CPU to be out of memory (we have not implemented our algorithms over parallel machines while carrying out the reported experiments). To avoid the problem leading to out of memory, we block the training data into a number of mini-batches, and use mini-batch training instead of full batch training. Only after the final mini-batch data are consumed in each training epoch, do we use a routine for block matrix multiplication, which incurs some undesirable but unavoidable waste of computation with a single CPU and Matlab code for the experiments, to combine the full training data.

In Figure 4, we show the results of one experiment where we used 6000 units in each layer of the DBN. Slightly different from the DBN used for the MNIST experiments, no skip-module output layers were used in building the higher modules of the DBN. In Figure 4, the frame-level phone-state classification percent error rate (with 183 state classes), together with three other performance measures, is plotted as a function of the training epoch (i.e. a full sweep of full 1.12M super-frames). These other measures include the frame-level phone classification percent error rate (61 phone classes as well as folded 39 phone classes) for the core test set when the errors in the state within the same phone are not counted.

The results shown in Figure 4 were obtained when the mini-batch size was set to 200,000 in fine tuning, and the step size in gradient descent was set to a value between 0.01 and 0.10. Besides the step size, a few other hyper-parameters are tuned using the dev set.

Like the MNIST task, we also have found empirically in the current TIMIT task that if random numbers were used for the initialization of the weight matrix instead of RBM, then the error rate becomes at least 30% higher than presented in Figure 4 in most cases.

There has been very few published work on frame-level phone or phone state classification. The closest work we have been able to find in [8] reported over 70% phone state error rate with an easier 132 phone state classes (than our 183 state classes) but with more a difficult speech database. We also ran the DBN system of [5] on the same TIMIT data and found the corresponding frame-level phone state error rate be to 45.04% (which gives a 22% phonetic recognition error rate after running a decoder as reported in [5]). This frame-level error rate

achieved by DBN is slightly higher than the DCN's error rate of 43.85% shown in Figure 4.

6. Summary and Future Directions

We recently developed a DBN-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 recognition 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.

Our exploration and experiments on speech recognition only started very recently and the preliminary results presented in Section 5 are encouraging. The future directions of this work include: 1) full exploration of much architectural flexibility provided by the basic DCN framework presented in this paper; 2) addition of a dynamic program based decoder on top of the final layer of the DCN to enable continuous phonetic or speech recognition; 3) learning (rather than tuning) hyper-parameters in DCN; 4) developing speaker and environment adaptation techniques for DCN; and 5) developing temporal DCN which integrates generative dynamic models of speech with the DCN architecture presented in this paper.

References

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

[2] 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.

[3] 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 - Special Issue on Deep Learning for Speech and Language Processing*, 2011 (in press).

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

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

[6] 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.

[7] Y. LeCun, L. Bottou, Y., Bengio, and P. Haffner "Gradient-Based Learning Applied to Document Recognition", *Proc. IEEE*, Vol. 86(11), pp. 2278-2324, 1998.

[8] J. Droppo, M. Seltzer, A. Acero, Y. Chiu. "Towards a non-parametric acoustic model: An acoustic decision tree for observation probability calculation," Interspeech, Sept, 2008.

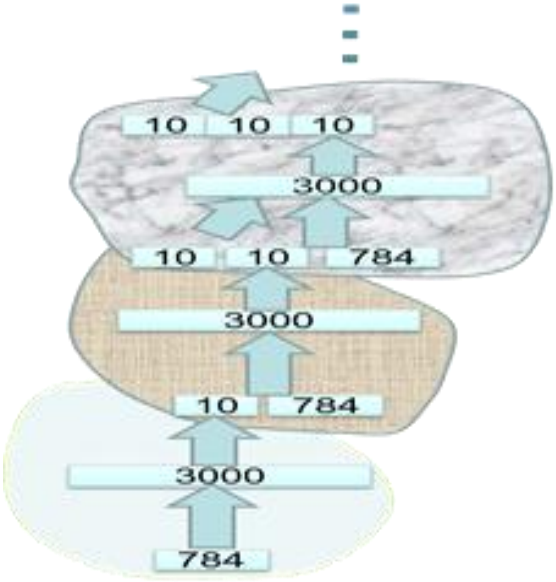

Figure 1:

The basic architecture of DCN used in the MNIST experiments. Three modules of the DCN are shown here, although most experiments reported in this paper involve much deeper DCNs. The input layer in the lowest module consists of 784 linear units, which is the number on pixels (28x28) in each raw image. Hidden layers with 3000 sigmoidal nonlinear units are shown here. The input layer in the higher modules of the DCN consists of 784 units appended by multiple of 10, where 10 (number of classes in the MNIST task) is the number of output units in each module of the DCN. Each arrow in the figure represents a weight matrix in the DCN.

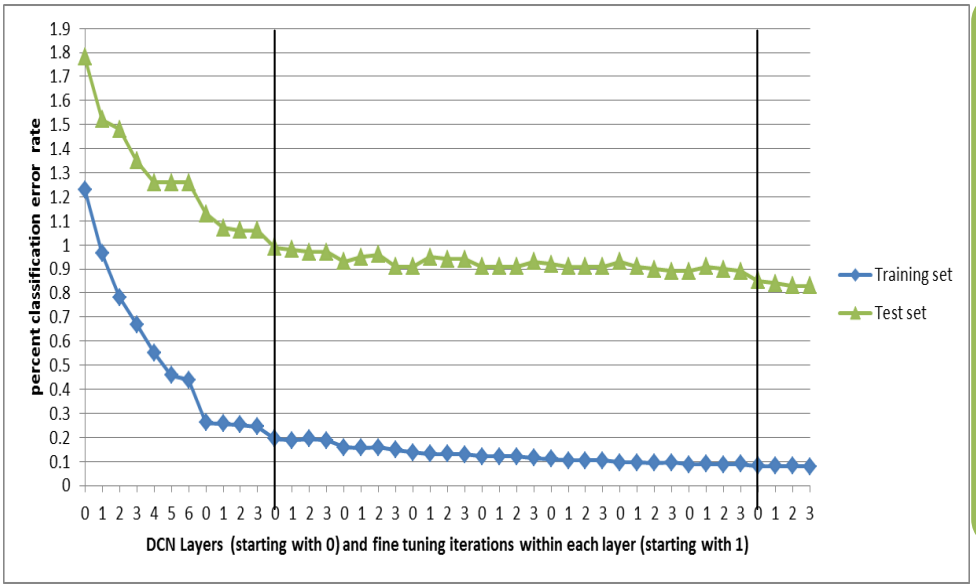

Figure 2:

MNIST test set frame-level classification error rate (%) as a function of DCN layers and fine tuning iteration number run within each layer. Horizontal axis "0" indicates the beginning of a new layer. A total of 10 layers/modules are used, giving the final 0.83% error rate. RBM was used to initialize the weight matrix.

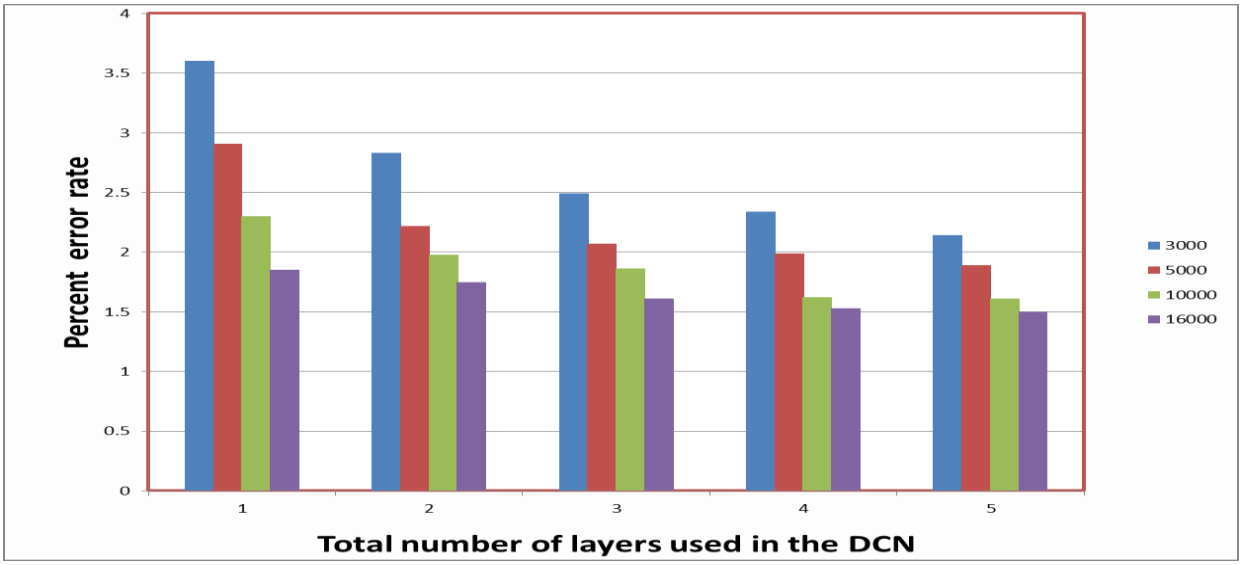

Figure 3: *MNIST test set classification error rate (%) when random numbers were used to initialize the weight matrix.*

DBN (Hinton's [1])	***DBN (Re-tuned by authors @ MSR)***	***DCN (Fine-tuning)***	***DCN (Copying RBMs up; no Fine-tuning)***	***Shallow (D)CN (Single layer w. heavy fine tuning)***
1.20%	*1.06%*	*0.83%*	*0.95%*	*1.10%*

Table 1: Error rate comparisons among DBNs and three variants of DCNs

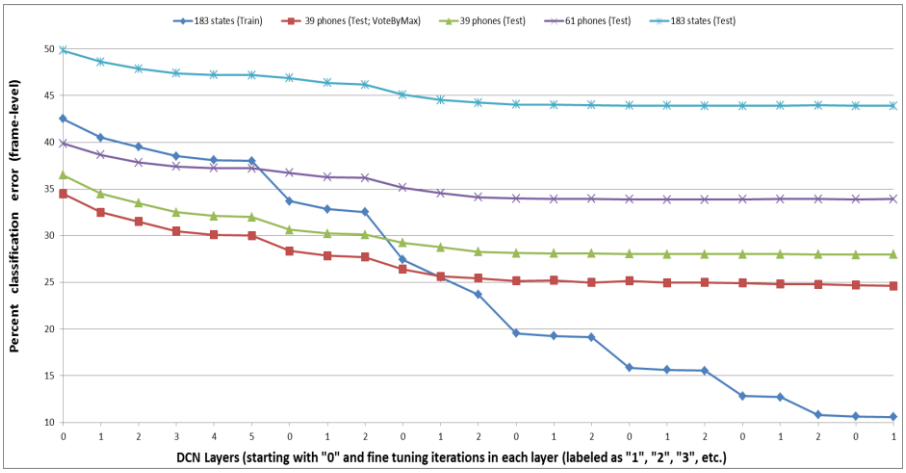

Figure 4: Frame-level classification error rates on TIMIT core test and training sets. Training set results with 183 phone states as the classes are in Blue; Test set results are in other colors with four kinds of performance measures: 1) 183 phone states as targets as for training; 2) 61 phones as targets when the confusion of three states within the same phone is discarded; 3) 39 phones as targets when the 61 phones are folded into the standard 39 phone set; and 4) 39 phones as targets when the decision is made by majority voting over frames with the boundaries of each phone.