Comparison and Combination of Multilayer Perceptrons and Deep Belief Networks in Hybrid Automatic Speech Recognition Systems

Van Hai Do*†, Xiong Xiao†, Eng Siong Chng*†
* School of Computer Engineering, Nanyang Technological University, Singapore
† Temasek Laboratories@NTU, Nanyang Technological University, Singapore
{dova0001, xiaoxiong, aseschng}@ntu.edu.sg

***Abstract*—To improve the speech recognition performance, many ways to augment or combine HMMs (Hidden Markov Models) with other models to build hybrid architectures have been proposed. The hybrid HMM/ANN (Hidden Markov Model / Artificial Neural Network) architecture is one of the most successful approaches. In this hybrid model, ANNs (which are often multilayer perceptron neural networks - MLPs) are used as an HMM-state posterior estimator. Recently, Deep Belief Networks (DBNs) were introduced as a newly powerful machine learning technique. Generally, DBNs are MLPs with many hidden layers, however, while weights of MLPs are often initialized randomly, DBNs use a greedy layer-by-layer pre-training algorithm to initialize the network weights. This pre-training initialization step has resulted in successful realizations of DBNs for various applications such as handwriting recognition, 3-D object recognition, dimensionality reduction and automatic speech recognition (ASR) tasks. To evaluate the effectiveness of the pre-initialization steps that characterize DBNs from MLPs for ASR tasks, we conduct a comparative evaluation between the two systems on phone recognition for the TIMIT database. The effectiveness, advantages and computational cost of each method will be investigated and analyzed. We also show that the information generated by DBNs and MLPs are complementary, where a consistent improvement is observed when the two systems are combined. In addition, we investigate the ability of the hybrid HMM/DBN system in the case only a limited amount of labeled training data is available.**

I. Introduction

Many researchers have been trying to augment or combine HMMs (Hidden Markov Model) with other models to build hybrid architectures to improve performance over the HMM/GMM (Hidden Markov Model / Gaussian Mixture Model) approach for acoustic modelling. The hybrid HMM/ANN (Hidden Markov Model / Artificial Neural Network) architecture proposed by Bourlard and Morgan [1] is one of the most successful approaches. In this hybrid model, ANNs (which are often multilayer perceptron neural networks - MLPs) are used to estimate HMM-state posterior probabilities instead of using GMMs in the traditional HMM/GMM approach. This hybrid model offers several advantages over the HMM/GMM approach such as: ANNs are discriminative as compared to GMMs. In addition, there is no required detailed assumption about input distribution. And they are very flexible in terms of merging multiple input streams. Normally, they use several frames surrounding the current frame as the input instead of using only one frame in the HMM/GMM systems.

Recently, Deep Belief Networks (DBNs) were introduced as a newly powerful machine learning technique [2]. Generally, DBNs are MLPs with many hidden layers, however, while weights of MLPs are initialized randomly, DBNs use a greedy layer-by-layer pre-training algorithm to initialize the network weights. Note that this process is totally unsupervised. DBNs have been applied effectively for many applications, including handwriting recognition [2], 3-D object recognition [3], dimensionality reduction [4]. They show significant advantages over the conventional MLPs. In [5], [6], [7], DBNs were used for speech recognition. However, there is no clear comparison between DBNs and MLPs in ASR tasks. In this paper, we conduct a comparative evaluation of hybrid HMM/DBN and HMM/MLP systems on the TIMIT speech database to show the effectiveness, advantages and computational cost of each method. We also show that the information generated by DBNs and MLPs are complementary, in that a consistent improvement is observed when the two systems are combined. In addition, we investigate the ability of the hybrid HMM/DBN system in the case where only a limited amount of labeled training data is available.

The rest of this paper is organized as follows. Section II briefly introduces Deep Belief Networks. The system architecture of hybrid HMM/DBN and HMM/MLP systems is described in Section III. The experimental setup and results are reported in Section IV and Section V, respectively, and Section VI concludes the paper.

II. Deep Belief Networks

Why do we need deep architectures with many layers? Complexity theory of circuits strongly suggests that deep architectures are much more efficient in terms of required computational elements than shallow architectures, including hidden Markov models, neural networks with only one hidden layer, conditional random fields, kernel regression, support vector machines, and many others [8]. Unfortunately, such deep networks are very hard to train. To overcome this problem, in 2006, Hinton et al. [2] introduced a moderately fast, unsupervised learning algorithm for deep generative models called Deep Belief Networks (DBNs). The greedy layer-by-layer training is the key feature of this algorithm in order to

efficiently learn a deep, hierarchical probabilistic model. A DBN is created as a stack of its main building blocks which are bipartite undirected graphical models called restricted Boltzmann machines (RBMs).

An RBM is a particular type of Markov random field that has a two-layer architecture, in which the visible stochastic units $\boldsymbol{v}$ (typically Bernoulli or Gaussian) are connected to the hidden stochastic units $\boldsymbol{h}$ (typically Bernoulli). Normally, all visible units are connected to all hidden units and there is no visible to visible or hidden to hidden unit connection. The weights of the connections and the biases of the individual units form a joint probability distribution $P(\boldsymbol{v}, \boldsymbol{h}|\theta)$ over the visible units $\boldsymbol{v}$ and hidden units $\boldsymbol{h}$ given the model parameters θ. This distribution is computed based on an energy function $E(\boldsymbol{v}, \boldsymbol{h}|\theta)$ [9]:

$$P(\boldsymbol{v}, \boldsymbol{h}|\theta) = \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}|\theta))}{Z(\theta)} \tag{1}$$

where $Z(\theta)$ is known as the normalizing constant.

$$Z(\theta) = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}|\theta)) \tag{2}$$

The marginal probability $P(\boldsymbol{v}|\theta)$ is computed as

$$P(\boldsymbol{v}|\theta) = \frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}|\theta))}{Z(\theta)} \tag{3}$$

With different types of visible and hidden units, different energy functions are defined. In the case for the visible and the hidden units which are Bernoulli, the energy function is defined [9].

$$E(\boldsymbol{v}, \boldsymbol{h}|\theta) = -\sum_{i=1}^{V} \sum_{j=1}^{H} w_{ij} v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j \tag{4}$$

where model parameters $\theta = \{\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{a}\}$, w_{ij} is the weight between visible unit i and hidden unit j, b_i and a_j are biases for visible unit i and hidden unit j, respectively. V, H are the numbers of visible units and hidden units, respectively.

Since there is no visible-visible connection, all of the visible units become independent given the hidden units, and vice versa. The conditional distributions can be effectively derived [9] as

$$P(h_j = 1|\boldsymbol{v}, \theta) = \sigma\left(\sum_{i=1}^{V} w_{ij} v_i + a_j\right) \tag{5}$$

$$P(v_i = 1|\boldsymbol{h}, \theta) = \sigma\left(\sum_{j=1}^{H} w_{ij} h_j + b_i\right) \tag{6}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoidal function.

Similarly, in the case for the visible and hidden units which are Gaussian and Bernoulli, respectively, the energy function is defined as

$$E(\boldsymbol{v}, \boldsymbol{h}|\theta) = -\sum_{i=1}^{V} \sum_{j=1}^{H} w_{ij} v_i h_j + \frac{1}{2} \sum_{i=1}^{V} (v_i - b_i)^2 - \sum_{j=1}^{H} a_j h_j \tag{7}$$

and the conditional probabilities

$$P(h_j = 1|\boldsymbol{v}, \theta) = \sigma\left(\sum_{i=1}^{V} w_{ij} v_i + a_j\right) \tag{8}$$

$$P(v_i|\boldsymbol{h}, \theta) = \mathcal{N}\left(\sum_{j=1}^{H} w_{ij} h_j + b_i, 1\right) \tag{9}$$

where v_i is a real number which follows a Gaussian distribution with the mean $\sum_{j=1}^{H} w_{ij} h_j + b_i$ and unit variance.

Equations 5 and 8 allow us to use the weights of an RBM after pre-training to initialize an MLP with sigmoidal activation functions. Since the inference for RBM hidden units can be equated with the forward phase in an MLP.

The goal of learning in an RBM is to maximize the marginal probability of the data $P(\boldsymbol{v}|\theta)$. It can be done very effectively by a procedure called "Contrastive Divergence" [10].

III. System architecture

In this research, we examine two hybrid ASR systems: HMM/MLP and HMM/DBN as illustrated in figure 1, where MLPs and DBNs are used as HMM-state posterior estimators. Given a feature vector $\boldsymbol{x_t}$, MLPs and DBNs estimate the HMM-state posterior $P(s_i|\boldsymbol{x_t})$ of state s_i. The posterior is then converted to the likelihood probability using the Bayes' formula.

$$P(\boldsymbol{x_t}|\, s_i) = \frac{P(\, s_i|\, \boldsymbol{x_t})}{P(s_i)} P(\boldsymbol{x_t}) \tag{10}$$

In practice, we use $\frac{P(s_i|\boldsymbol{x_t})}{P(s_i)}$ as the scaled likelihood since the scaling factor $P(\boldsymbol{x_t})$ is a constant for all states and does not affect the classification decision. This approach has been applied successfully in hybrid HMM/ANN systems [1].

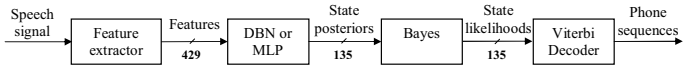

Fig. 1. DBNs or MLPs are used as HMM-state estimators in hybrid ASR systems.

Originally, DBNs were built by stacking RBMs with binary units. However, features used in speech recognition (e.g., MFCCs, PLPs) are continuous. Hinton et al. [2] indicated that we can handle continuous-valued inputs by scaling them to the [0, 1] interval, so that each continuous valued input can be considered as the probability for a binary random variable to take the value 1. This approach worked well for handwritten image recognition [2] where pixel gray levels were used as the input. However, when applying to speech recognition, we

realized that the recognition accuracy was much worse than the state-of-the-art ASR systems on the TIMIT corpus. It can be explained that, this coding approach may be inappropriate for complicated kinds of input variables which have a wide dynamic range such as speech signal. In this research, we use Gaussian visible units for the first RBM, to represent the continuous speech features, and Bernoulli hidden units, whereas all upper RBMs are Bernoulli-Bernoulli. This approach has been applied successfully in speech recognition [5], [6], [7].

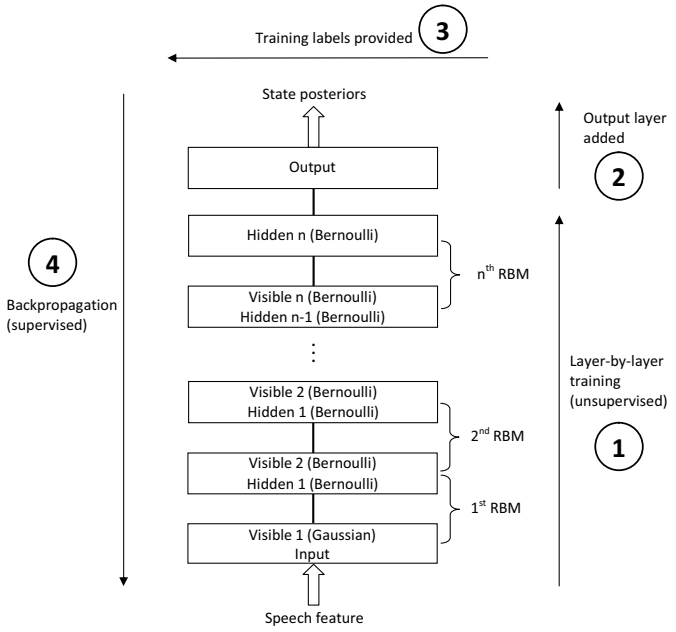

Fig. 2. Four steps in training a DBN for speech recognition.

Figure 2 illustrates four basic steps in training a DBN for speech recognition. After unsupervised training the first RBM, the activation probabilities of its hidden units are used as the visible data for the second RBM, and so on. When applying the DBN for speech recognition, the final layer of variables is added to represent the desired outputs (HMM-states). Next, a discriminative learning procedure, such as backpropagation, is used to fine-tune all of the network weights using labeled training data. In our experiments, the activation function at the output layer is *softmax* to approximate posterior probabilities of states appearing at the current frame. In the decoding process, the DBN is treated as an MLP with the same architecture. The input features are processed layer by layer from the input layer to the final layer.

IV. Experimental setup

Database: The TIMIT database[1] is used in our experiments. The SA part of the TIMIT database is not used since it contains identical utterances for all speakers in the corpus, hence it can bias the result. The training set consists of 3696 utterances from 462 speakers. A small part extracted from the training set (50 speakers) is used as the development set. The complete test data set contains 1344 utterances from 168 speakers.

Phone set: The original 64 phonetic labels on TIMIT are mapped into the 45 phones as described in [11]. Phones "cl", "vcl", and "epi" are merged into the phone "sil". For phone recognition evaluation, after decoding, the phone accuracies are computed by down mapping the recognition output from 45 phones to 39 phones [11].

Features: The features used in all experiments are 12^{th}-order Mel frequency cepstral coefficients (MFCCs) and energy, along with their first and second temporal derivatives. The frame length is 25ms and the frame shift is 10ms. The features are normalized to zero mean and unit variance over the entire training set. All experiments use a context window to concatenate the 11 frames which surround the current frame. Hence we have 39x11=429 dimensional feature vectors. We use 135 state labels (i.e., 3 states for each phone of 45 phones). These labels are generated by a conventional HMM/GMM system.

Language model: No language model is used in all experiments.

Software: The Matlab scripts for DBN and MLP training are retrieved from Hinton's website[2]. The Viterbi algorithm used to produce the recognized phone strings is implemented by HVite, and HResults in the HTK speech toolkit[3] is used to evaluate the phone recognition results.

Parameter selection:

- *Weight initialization:* Both MLP and RBM weights are initialized randomly with a normal distribution with mean 0 and standard deviation 0.1.

- *Learning paremeters:* In the pre-training process, all RBMs are trained using stochastic gradient decent. In the first RBM (Gaussian-Bernoulli), the learning rate is 0.01 when the number of hidden units is 256. It reduces to 0.005 for the cases 512, 768, 1024, and 1536 hidden units. When the number of hidden units reaches 2048, the learning rate is selected as 0.002. For other RBMs (Bernoulli-Bernoulli), the learning rate is fixed at 0.1. For the first five epochs, the momentum is initialized to 0.5, and set to be 0.9 for subsequent epochs. All RBMs are trained for 50 epochs. A weight decay of 0.0002 is used to penalize large weights. A detailed explanation of learning parameter selection for DBNs can be found in [12].

- *Decoding paremeters:* In the decoding process, for each experiment, the phone insertion penalty is tuned based on the development data.

- *Transition probabilities:* In all experiments, we simply set all transition probabilities which include the self loop probability and the probability from the current state to the next state to be equal to 0.5. Further improvements can be obtained if they are borrowed from an existing ASR model and tuned carefully.

Computer: All experiments are conducted in computers which have following specifications:

[1] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1

[2] http://www.cs.toronto.edu/∼hinton/MatlabForSciencePaper.html

[3] HTK speech toolkit, http://htk.eng.cam.ac.uk

- OS: Windows 7 professional 64 bits
- CPU: 2 x (4-core-Intel Xeon 2.67GHz)
- RAM: 8GB

V. Experimental results

In all experiments, DBNs and MLPs have the same architecture (number of layers, units per layer, activation functions) as well as learning parameters. DBNs are only different from MLPs in the unsupervised pre-training process, since the weights of MLPs are initialized randomly. To simplify the comparison, we use the same size for every hidden layer in each network.

A. Different network sizes

1) Size of layers: In this experiment, we compare the performance of DBNs and MLPs in phone recognition when the number of hidden units increases from 256 to 2048. The number of hidden layers is kept fixed at 3. The results for the test set of DBNs and MLPs are shown in figure 3.

Surprisingly, in small networks with the layer size less than 1024, conventional MLPs outperform DBNs. It can be explained that DBNs are constructed by stacking up several RBMs, where the first RBM is Gaussian-Bernoulli. Note that 429 continuous valued inputs are represented by 429 Gaussian visible units, while hidden units are binary (Bernoulli). When the number of hidden units is small (e.g., 256, 512, 768), the binary hidden layer cannot infer the complicated data distribution at the input layer. Hence the network parameters can obtain extreme values. In the experiments, we realized that after the unsupervised pre-training process of the first RBM, the biases of the hidden layer take very large values, even when smaller learning rates were applied. This initialization is not good for the backpropagation algorithm in the fine-tuning process. In this case, the small random initialization in conventional MLPs is preferred.

However, when the hidden layer size is large enough ($\geq$ 1024 units/layer), the hidden layer can infer the data distribution at the visible layer, and we observe that DBNs consistently perform better than MLPs.

A possible suggestion for using DBNs for continuous valued inputs is that we can use a large number of hidden units for the first hidden layer, and higher level hidden layers can be assigned with smaller numbers of hidden units.

The combination of information from different ASR systems generally improves speech recognition accuracy. The reason for this advantage is explained by the fact that different systems often provide different errors. In this research, we examine the combination of DBNs and MLPs at the probability level. The state posterior probability of state s_i given the input vector $\boldsymbol{x_t}$ generated by DBNs: $P_{\mathrm{DBN}}(s_i|\boldsymbol{x_t})$ and MLPs: $P_{\mathrm{MLP}}(s_i|\boldsymbol{x_t})$ are combined by the simple unweighted sum rule and the unweighted product rule [14] as:

$$P_{\mathrm{sum}}(s_i|\boldsymbol{x_t}) = 0.5P_{\mathrm{DBN}}(s_i|\boldsymbol{x_t}) + 0.5P_{\mathrm{MLP}}(s_i|\boldsymbol{x_t}) \quad (11)$$

$$P_{\mathrm{product}}(s_i|\boldsymbol{x_t}) = \frac{P_{\mathrm{DBN}}(s_i|\boldsymbol{x_t})P_{\mathrm{MLP}}(s_i|\boldsymbol{x_t})}{\sum_{i=1}^{N} P_{\mathrm{DBN}}(s_i|\boldsymbol{x_t})P_{\mathrm{MLP}}(s_i|\boldsymbol{x_t})} \quad (12)$$

where N is number of classes (states).

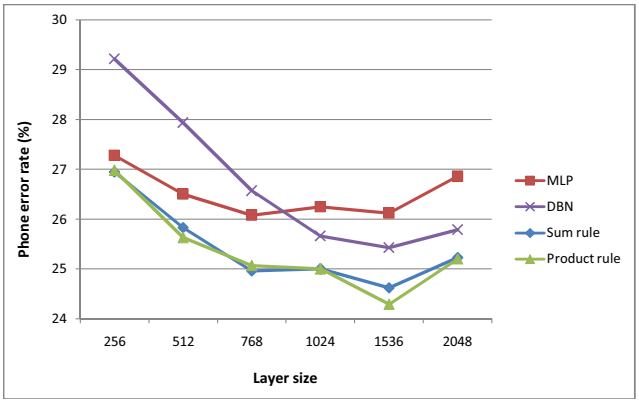

Fig. 3. Phone error rates on the test set of the DBN, the MLP and the combined systems with different hidden layer sizes.

The recognition results of the combined systems are shown in figure 3. The combined systems outperform both the DBN and MLP systems significantly in all cases. No big difference between the two combination schemes is observed. However, the product rule performs slightly better than the sum rule.

2) Depth of networks: This experiment is conducted to examine the performance of DBNs and MLPs when difference numbers of hidden layers are used. Figure 4 explores the effect of varying the number of hidden layers for DBNs and MLPs. In this case, the number of units in each hidden layer is kept fixed at 1024. Through the figure, it can be seen that adding the second hidden layer gives better performance in both DBNs and MLPs. However, with deeper architectures, MLPs do not achieve further improvement, and they even start performing worse when hidden layer 5 is added. Whereas the performance of DBNs is improved when more hidden layers are added, although the improvement is not large.

The posterior probabilities generated by the two individual systems are combined by the sum rule and the product rule. Significant improvements over both the DBN and MLP systems are observed in figure 4 .

B. Speed of convergence

Figure 5 illustrates the speed of convergence of DBNs and MLPs. In this experiment, a 3-hidden-layer architecture with 1536 hidden units in each layer is used. It is clearly seen that DBNs can converge after only a few epochs, while MLPs need nearly 100 epochs to converge. The fast convergent ability of DBNs is thanks to the unsupervised pre-training process. After pre-training, DBNs are close to the global optimal point. Hence DBNs can converge after a few fine-tuning iterations (backpropagation) [2].

Note that in the fine-tuning process of DBNs, a significant reduction in phone error rate is observed in epoch 7. The

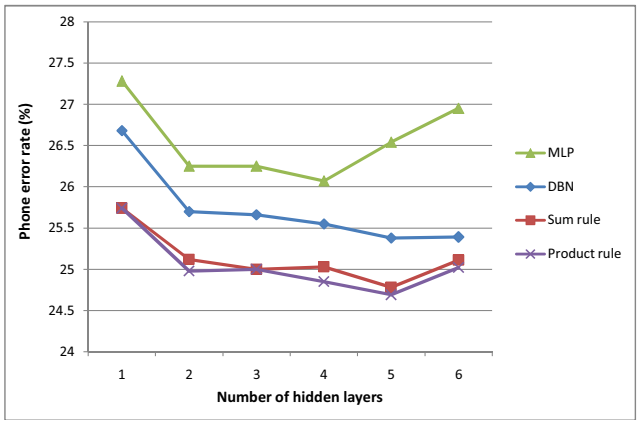

Fig. 4. Phone error rates on the test set of the DBN, the MLP and the combined systems with different numbers of hidden layers.

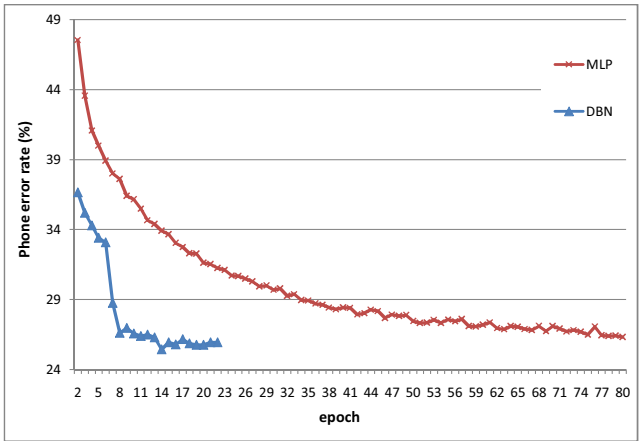

Fig. 5. Comparison of convergence speed between DBNs and MLPs in the error backpropagation phase.

reason is that the fine-tuning process of DBNs are divided into two phases. The first phase consists of the first six epochs and the rest of the epochs belong to the second phase. In the first phase, only the weights which connect the two top layers are trained, while the other weights are frozen. In the second phase, all network weights are trained jointly.

Each epoch in the backpropagation process takes around 3.5 hours. Hence the training process in MLPs takes around 12 days for 80 epochs while the fine-tuning process of DBNs takes only $1/4$ of that time (for 20 epochs). However, the total training time of DBNs involves both the pre-training and the fine-tuning processes, where the pre-training is also time consuming. In this experiment, DBNs with 3 hidden layers are used, that means 3 RBMs need to be trained. The first RBM consists of 429 visible units (i.e., dimensions of speech feature vectors) and 1536 hidden units, while in the two remaining RBMs, both the visible and hidden layers contain 1536 units. Each RBM is trained with 50 epochs, where each epoch takes 25 minutes and 45 minutes for the first RBM and the two remaining RBMs, respectively. Hence, the pre-training process takes around 4 days. Overall, while MLPs take 12 days for training, DBNs need only 7 days (4 days for pre-training and 3 days for fine-tuning). Note that the training process can be accelerated significantly if a highly parallel machine is used such as graphics processing units (GPUs) instead of a normal computer [5], [6], [7].

C. Limited labeled training data

In this experiment, we compare DBNs and MLPs for the case when limited labeled training data are available. The training corpus is divided into two parts. The first part is the labeled training data and the remaining part is treated as the unlabeled training data. Both DBNs and MLPs can only use the labeled part for fine-tuning the weights with the back-propagation algorithm. However, as opposed to MLPs, DBNs benefit from the unsupervised pre-training process which can use both the unlabeled and the labeled data. This advantage is very important since labeled data are very expensive and limited whereas unlabeled data are almost unlimited and much easier to collect.

Table I lists the phone error rate of DBNs, MLPs and the combined systems with different percentages of labeled training data. The relative improvement of DBNs and the combined systems over MLPs is illustrated in figure 6. It can be seen that, the smaller the amount of labeled training data is used, the better DBNs perform over MLPs. Whereas the relative improvement of the combined systems over the MLP system remains stable with different percentages of labeled training data. Although this improvement is not as large as expected, it points out that we can improve the speech recognition accuracy by using unlabeled speech data. Also note that this method is different from other approaches such as incremental training [13] where the high confidence utterances are recognized and used with labeled utterances to adapt or re-train the model. This procedure is normally repeated for several iterations and it is very time consuming. In the DBN method, the unlabeled training data are used only one time to pre-train DBNs before using the labeled data to fine-tune the network weights. Hence, it can perform faster than other methods. As a result, a larger amount of unlabeled training dataset can be used effectively.

TABLE I
PHONE ERROR RATE OF THE MLP, THE DBN AND COMBINED SYSTEMS IN PHONE RECOGNITION WITH DIFFERENT AMOUNT OF LABELED TRAINING DATA.

	Individual system(%)		Combined system(%)	
% of labeled training data	MLP	DBN	Sum rule	Product rule
5%	43.42	40.56	41.10	41.09
10%	36.97	35.20	34.89	34.85
20%	33.27	31.89	31.39	31.45
50%	28.78	29.03	27.29	27.14
100%	26.25	25.66	25.00	25.00

VI. CONCLUSION AND DISCUSSION

In this paper, a comparative evaluation of DBNs and MLPs on the TIMIT speech database has been conducted. It has been shown that when the network size is large enough, DBNs

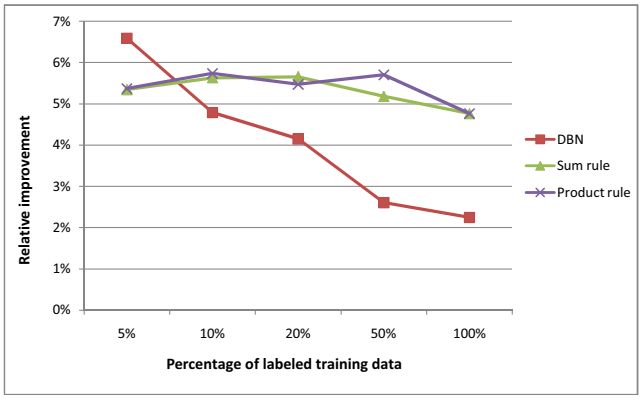

Fig. 6. The relative improvement of DBNs and the combined systems over MLPs in different percentages of labeled training data.

outperformed MLPs consistently. In addition, DBNs could be trained much faster than MLPs with the same architecture. Our experiments also pointed out that using unlabeled training data could improve the performance of the hybrid HMM/DBN system. However, experimental results also showed that the improvement of DBNs over conventional MLPs was not as large as the improvement in image classification shown in [4]. Hence there are still several questions that need to be solved by further research. For instance, what type of RBMs can better model speech features? Currently, DBNs are still used to estimate the state posteriors in the hybrid HMM/DBN model. The better combination methods should be investigated in the future.

We also showed that a consistent improvement is observed when the state posterior probabilities generated by DBNs and MLPs were combined. Although the two systems had an identical structure, the pre-training process in DBNs made the information generated by DBNs are complementary with MLPs. In this study, only the two simplest combination schemes (unweighted sum and unweighted product rules) were investigated. Further improvement can be obtained if more complicated combination schemes are used (e.g., weighted combinations, nonlinear combinations). In addition, we only combined the two systems at the probability level. In the future, combination of DBNs and MLPs at the feature level (e.g., Tandem [15]) or the hypothesis level (e.g., ROVER [16]) should be investigated.

REFERENCES

[1] H. Bourlard and N. Morgan, "Continuous speech recognition by connectionist statistical methods," *Neural Networks, IEEE Transactions on*, vol. 4, pp. 893-909, Nov. 1993.

[2] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets", *Neural computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

[3] V. Nair and G. Hinton, "3-d object recognition with deep belief nets.", *Advances in Neural Information Processing Systems*, no. 22, 2009.

[4] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, p. 504, 2006.

[5] A. Mohamed, G. Dahl, and G. Hinton, "Deep Belief Networks for phone recognition," in *Proc. of NIPS 2009 Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[6] A. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[7] G. Dahl, A. MarcAurelio Ranzato, and G. Hinton, "Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine," *Advances in Neural Information Processing Systems*, vol. 24, 2010.

[8] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems* 19: Proceedings of the 2006 Conference, p. 153, The MIT Press, 2007.

[9] R. Salakhutdinov, *Learning deep generative models.* PhD thesis, University of Toronto, 2009.

[10] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771-1800, 2002.

[11] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1641-1648, 1989.

[12] G. E. Hinton, "A practical guide to training restricted boltzmann machines," Tech. Rep., Univ. Toronto, Aug. 2010.

[13] B. Varadarajan, D. Yu, L. Deng, A. Acero, "Using collective information in semi-supervised learning for speech recognition," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on* , pp. 4633-4636, 19-24 April 2009.

[14] K. Kirchhoff, "Combining articulatory and acoustic information for speech recognition in noisy and reverberant environments," in *ICSLP*, pp. 891-894, 1998.

[15] H. Hermansky, D. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 1635-1638, 2000.

[16] J. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)," in *Automatic Speech Recognition and Understanding, 1997. Proceedings., IEEE Workshop on*, pp. 347-354, Dec. 1997.