A CLUSTER-BASED MULTIPLE DEEP NEURAL NETWORKS METHOD FOR LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

Pan Zhou[1], *Cong Liu*[2], *Qingfeng Liu*[2], *Lirong Dai*[1], *Hui Jiang*[3]

[1]National Engineering Laboratory of Speech and Language Information Processing,
University of Science and Technology of China, Hefei, P.R.China
[2]Anhui USTC iFLYTEK Corporation Limited, Hefei, P.R.China
[3]Department of Computer Science and Engineering, York University, Toronto, Canada

pan2005@mail.ustc.edu.cn, {congliu2,qfliu}@iflytek.com, lrdai@ustc.edu.cn, hj@cse.yorku.ca

ABSTRACT

Recently a pre-trained context-dependent hybrid deep neural network (DNN) and HMM method has achieved significant performance gain in many large-scale automatic speech recognition (ASR) tasks. However, the error back-propagation (BP) algorithm for training neural networks is sequential in nature and is hard to parallelize into multiple computing threads. Therefore, training a deep neural network is extremely time-consuming even with a modern GPU board. In this paper we have proposed a new acoustic modelling framework to use multiple DNNs instead of a single DNN to compute the posterior probabilities of tied HMM states. In our method, all tied states of context-dependent HMMs are first grouped into several disjoined clusters based on the training data associated with these HMM states. Then, several hierarchically structured DNNs are trained separately for these disjoined clusters of data using multiple GPUs. In decoding, the final posterior probability of each tied HMM state can be calculated based on output posteriors from multiple DNNs. We have evaluated the proposed method on a 64-hour Mandarin transcription task and 309-hour Switchboard Hub5 task. Experimental results have shown that the new method using cluster-based multiple DNNs can achieve over 5 times reduction in total training time with only negligible performance degradation (about 1-2% in average) when using 3 or 4 GPUs respectively.

Index Terms— LVCSR, DNN, state clustering, cluster-based multi-DNN, parallelization among GPUs

1. INTRODUCTION

The state of the art speech recognition systems rely on acoustic models of context-dependent phones, each of which is modelled by a 3-state hidden Markov model (HMM). Gaussian mixture models (GMM) are often used in each tied state to model probability density of acoustic observations. Since long time ago [1], feed-forward multi-layer perceptron (MLP) based artificial neural network (ANN) has been considered as an alternative to GMM for modelling posterior probabilities of HMM states given an acoustic pattern. In more recent years, the pre-trained context-dependent hybrid deep neural network (DNN) and HMM method has been attracting more and more research attention as a promising acoustic modelling approach for ASR [2, 3, 4] since it has achieved significant performance gain over GMMs in many large scale speech recognition tasks [5, 6, 7, 8, 9, 4]. However, for a long time, extremely long training time has been considered as the Achilles heel of artificial neural network. ANNs need to be trained by stochastic gradient descent (SGD) algorithm in the standard error back-propagation framework. It has been shown [10] that performance of ANNs heavily depends on the amount of available training data. For large MLPs, e.g. DNNs, thousands of hours of speech data is normally used, amounting to several hundreds million training samples. Recently, training of large ANNs has been remarkably accelerated since GPUs were introduced for general purpose computing by NVIDIA's high level programming language CUDA [11, 12, 13]. However, even with help of powerful GPUs, a typical training process for a large DNN in ASR may still last for weeks or even months [14].

It is known that the SGD algorithm is a serial training method in nature and it is not straightforward to parallelize it into independent computing threads to explore multiple CPUs or GPUs. Recently, some methods have been proposed to conduct parallel training of ANNs by taking advantage of a number of CPUs in a computing cluster. In [15, 16], it is proposed to achieve parallelization using multi-computers based on the server-client mode, where training data is split into many bunch-sized training patterns to several client computers to compute update matrices. Similarly, in [17], the training set is divided into N disjoint subsets in each epoch and a separate MLP is trained on each subset. Then, these sub-MLPs are combined by a merger network that is trained by another subset of data. In [18], it scales up training to a cluster with thousands of CPUs by implementing a parallel training scheme based on the so-called asynchronous SGD. However, these methods still suffer from communication overhead problem when collecting gradients and redistributing updated model parameters among computers, which may become the major bottle neck in large-scale training. More recently, in [19], a pipelined implementation of BP is proposed for parallel training of DNN using multiple GPUs, where computation related to different layers of DNN is distributed to several GPUs. It was reported that this method achieves about 3.3x speed-up in total training time using 4 GPUs when comparing with the standard mini-batch SGD using one GPU.

In this paper, we propose to use a new cluster-based multiple DNNs to replace a single large DNN in the hybrid DNN-HMM model for speech recognition. In this method, we first automatically cluster large training set into several subsets that have disjoint class labels. Different from other previous data division methods, we use an unsupervised clustering method to group training data so that all subsets are disjoint in terms of labels of tied HMM states. In

This work is partially funded by the National Nature Science Foundation of China (Grant No. 61273264) and the National 973 program of China (Grant No. 2012CB326405).

this way, we can independently train a smaller DNN on each subset of training data to distinguish different labels within this cluster and another even smaller DNN to distinguish different clusters. In this way, it is obvious that training of multiple DNNs can be done in parallel by using multiple GPUs without causing any significant data traffic among GPUs. As a result, this method can effectively take advantage of multiple GPUs to significantly reduce the total training time in many large-scale ASR tasks. We have evaluated the proposed method on a 64-hour Mandarin transcription task and 309-hour Switchboard Hub5 task. Experimental results have shown that the new method using cluster-based multiple DNNs can achieve over 5 times reduction in total training time with only negligible performance degradation (about 1-2% in average) when using 3 or 4 GPUs respectively.

The remainder of this paper is organized as follows. Section 2 reviews the standard hybrid DNN-HMM model. In section 3, we describe the proposed cluster-based multiple DNNs method in detail. The experimental results are reported in section 4. The paper is concluded with our findings and future work in Section 5.

2. DNN-HMM TRAINING STEPS

The structure of DNN is a conventional multi-layer perceptron with many hidden layers (usually more than 3 hidden layers). The hybrid DNN-HMM model attempts to directly model posterior probabilities of all tied triphone HMM states [7]. In this section, we briefly review the main training steps in DNN-HMM.

2.1. DNN-HMM

A $(L+1)$-layer DNN is used to model the posterior probability $P_{s|o}(s|o)$ of an HMM tied-state s given an observation vector o. The first L layers, $\ell = 0...L-1$, use sigmoid activation function to compute input to the next layer, while the top layer L uses softmax operation to compute posteriors for all classes as:

$$P^{\ell}_{h_j|v}(h^{\ell}_j|v^{\ell}) = 1/(1+e^{-z^{\ell}_j(v^{\ell})}) = \sigma(z^{\ell}_j(v^{\ell})), \quad 0 \le \ell < L, \quad (1)$$

$$P^{L}_{s|v}(s|v^{L}) = \frac{e^{z^{L}_{s}(v^{L})}}{\sum_{s'} e^{z^{L}_{s'}(v^{L})}} = \text{softmax}_s(z^{L}(v^{L})), \quad (2)$$

$$z^{\ell}(v^{\ell}) = (W^{\ell})^T v^{\ell} + a^{\ell}, \quad (3)$$

where W^{ℓ} and a^{ℓ} represent weight matrix and bias vector for hidden layer ℓ, and h^{ℓ}_j and $z^{\ell}_j(v^{\ell})$ are the j-th component of h^{ℓ} and $z^{\ell}(v^{\ell})$, respectively.

DNN is often trained with stochastic gradient descent in a well-formed error back-propagation procedure:

$$(W^{\ell}, a^{\ell}) \leftarrow (W^{\ell}, a^{\ell}) + \eta \frac{\partial \mathcal{L}}{\partial (W^{\ell}, a^{\ell})}, \quad 0 \le \ell \le L, \quad (4)$$

where $\mathcal{L}$ is an objective function and η is the learning rate. In ASR, objective is usually set to maximize the total log posterior probability over all training patterns $o(t)$ with class labels $s(t)$, i.e.

$$\mathcal{L} = \sum_{t=1}^{T} \log P_{s|o}(s(t)|o(t)), \quad (5)$$

The detailed form for the gradient of the objective function $\mathcal{L}$ with respect to W^{ℓ} and a^{ℓ} can be found in [6, 7]. Lastly, when using DNN-HMM for decoding, state posteriors $P_{s|o}(s|o)$ generated from the DNN are converted to scaled likelihoods by dividing their priors [6, 7] and then are sent to a Viterbi decoder to search for the best path.

2.2. Initialization with pre-training

It is well-known that BP can easily get stuck into any poor local optima in deep networks because of its random initialization. As a result, it is very important to alleviate this by using a layer-wise pre-training algorithm.

The pre-training procedure treats each consecutive pair of layers in DNN as a restricted Boltzmann machine (RBM). RBM is a two-layer undirected neural network that associates each configuration of visible and hidden state vector with an energy function. One step contrastive divergence (CD) algorithm [11] can efficiently learn the connections between two layers of RBM in an unsupervised way. Then multiple layers can be trained in a layer-wise manner. In that method, we train the first RBM and fix it, then use activations of the output layer of this RBM as the input to the next layer and continue until all layers have been estimated. After that, we will initialize all weights of DNN using the learned RBM connections and a randomly initialized softmax output layer is created at the top of the last RBM. At the end, the above-mentioned BP algorithm is used to fine tune all parameters in DNN until convergence. Details on RBM based pre-training can be found in [11, 5, 6].

3. CLUSTER-BASED MULTI-DNNS AND ITS PARALLEL TRAINING METHOD

The hybrid DNN-HMM architecture models the temporal nature of speech with HMM and uses DNN to replace GMMs to compute posterior probabilities of tied HMM states, which can be converted to scaled likelihoods for Viterbi decoding. In large vocabulary ASR tasks, it is common to have tens of thousands of tied HMM states, which makes the output layer of DNN extremely big and may significantly slow down the BP process in DNN training [14, 19]. Here we investigate to use multiple DNNs for acoustic modelling so that training of multiple DNNs can be easily parallelized among different computing units.

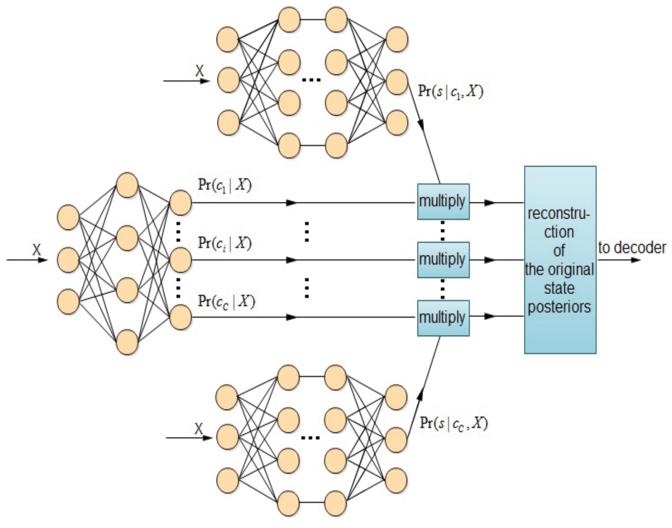

Fig. 1. Illustration of using multiple DNNs for acoustic modelling.

Fig. 1 illustrates how to use multiple DNNs for acoustic modelling. Suppose we totally have N tied HMM states in the GMM-HMM baseline system, all data in the training set is forced-aligned against word level transcriptions to generate state alignments. Based

Algorithm 1 Cluster-based Multiple DNNs Method

```
Training:
 1: Train a GMM-HMM baseline system with $N$ tied states, denoted as gmm-hmm.
 2: Use gmm-hmm to generate state level alignments of all training data.
 3: Split training data belonging to $N$ tied-states into several (e.g. $C$) disjoint clusters.
 4: Generate a mapping from each tied state to the cluster label to which it belongs, denoting this mapping as state2clststateid.
 5: Use the entire training set to train a small NN, i.e., $dnn_0$.
 6: Use all clustered subsets of training data to train multiple smaller DNNs, denoted as $dnn_i, 1 \le i \le C$.
Decoding:
 7: Feed each input frame to all the above DNNs to compute $Pr(c_i|X)$ and $Pr(s_j|c_i, X), 1 \le i \le C, s_j \in c_i$.
 8: Use Eq. (7) and mapping state2clststateid to compute posteriori probabilities $Pr(s_j|X)$ of all tied HMM states.
 9: Normalize $Pr(s_j|X)$ and send to Viterbi decoder for decoding.
```

on the state alignments, all speech frames are assigned to one of the state labels. Some unsupervised clustering techniques, such as k-means or GMM based clustering, can be used to split training data into several clusters without containing common state labels. Taking GMM clustering as example, any C states are randomly selected to estimate C initial GMMs (one for each cluster). The remaining states are classified to one cluster based on the estimated GMMs and then all GMMs are re-estimated based on data assigned to them. This process is repeated until all clusters converge.

After data clustering, we start to train several hierarchically structured DNNs. First of all, a small shallow NN (with 3 hidden layers), denoted as dnn_0, is trained to distinguish different clusters in the training data, i.e., for computing posteriori probability of each cluster c_i given X, $Pr(c_i|X)$. Since this is a very small NN, containing 3 hidden layers and a small number of output nodes, training of dnn_0 is very fast even though it needs to access the entire training set along with cluster labels. Meanwhile, all clustered subsets of training data are used to train multiple DNNs to classify different states within each cluster, i.e., for computing posteriori probabilities of all tied states within each cluster, $Pr(s_j|c_i, X)$. Since each of these DNNs is trained only on a subset of training data, all DNNs including dnn_0 can be separately trained in parallel using different GPUs without transferring any data or gradients among GPUs during the entire training process. Moreover, it is important to note that each of these DNNs is much faster to train than the joint large DNN in the normal DNN-HMM method because each DNN only involves a faction of training data belonging to that cluster and each DNN has much less output classes, leading to a smaller DNN in size. The main steps involved are summarized in Algorithm 1.

As for decoding, each observation sample, X, is fed into all of the estimated DNNs to compute $Pr(c_i|X)$ and $Pr(s_j|c_i, X)$, as illustrated in Fig. 1. Generally speaking, we can calculate the posteriori probability of any tied state, s_j, as follows:

$$Pr(s_j|X) = \sum_{c_i} Pr(c_i|X) \cdot Pr(s_j|c_i, X). \qquad (6)$$

Moreover, since all c_i are disjoint in terms of state labels, the above computation can be simplified as follows:

$$Pr(s_j|X) = Pr(c_i|X) \cdot Pr(s_j|c_i, X) \quad (\text{with } \ s_j \in c_i). \qquad (7)$$

This posterior probability is used for decoding in the same way as in the normal hybrid DNN-HMM model in [7].

4. EXPERIMENTS

In this section, we will evaluate the proposed cluster-based multiple DNNs method on two large vocabulary ASR tasks, namely 64-hr Mandarin Chinese transcription task and the 309-hr Switchboard task. The proposed method is compared with the conventional single DNN method in terms of both recognition performance and training efficiency. The open source QuickNet [20] is adopted to perform DNN fine tuning on 4 NVIDIA Geforce GTX590 GPU cards.

4.1. Mandarin transcription task

The training set of the Mandarin transcription task contains 76,843 utterances (about 64 hours) from 1,500 speakers and an independent test set contains 3,720 utterances (about 3 hours) from other 50 speakers. The standard GMM-based model is estimated based on maximum likelihood (ML) criterion, which includes 3969 tied HMM states and 30 Gaussian mixtures per state. This GMM-HMM model is also discriminatively trained based on MPE criterion. The GMM-HMM is used to do forced-alignment to generate state level segmentation for DNN training. In Table 1, we give the baseline recognition performance of GMM-HMM (both ML and MPE) for comparison purpose.

Table 1. *Baseline recognition performance (CER in Mandarin and WER in Switchboard) of various GMM/HMM models*

	Mandarin	Switchboard	
		Hub98	Hub01
ML-GMM	18.2%	46.6%	37.2%
MPE-GMM	16.7%	43.4%	32.8%

We have trained the baseline DNN-HMM systems with a variable number of hidden layers (from 3 to 6 layers and 1024 hidden units per layer) using 11 consecutive frames of MFCC. The output layer of DNN has 3969 units corresponding to all tied HMM states. DNN is first pre-trained using RBM-based algorithm layer by layer, and followed by a supervised fine tuning step using mini-batch SGD to minimize the cross entropy objective function.

For the parallelized multi-DNN-HMM system, we first cluster all 3969 HMM states into 4 sub-classes using the above GMM clustering method. This partition of data is denoted as *PAR-A* in Table 2. It is noted that this partition is not balanced in data because silence (3 states) and short pause (1 state) account for nearly 20% data. As a result, we use GMM method to cluster all other states (excluding these 4 state data) into 4 clusters. And these silence and short pause data are uniformly distributed to the 4 clusters. This leads to a more balanced data partition, denoted as *PAR-B*. In this case, we calculate $P(sil|X)$ based on Eq. (6) since sil exists in all 4 clusters. When training multi-DNN, we fix $dnn0$ to only 3 hidden layers and configure all subsequent DNNs with various hidden layers (3-6). We keep the size of each hidden layer the same as that of the baseline to reuse the pre-trained weights obtained for the baseline DNN and the last layer is randomly initialized. Each of these multi-DNN is smaller than DNN in the baseline in number of weights since it has much less output labels in the final layer. At the end, the error back propagation is performed to fine tune all weights using a number of different GPUs. In our experiments, each DNN is trained with 10 epoches. The GPU finishes last determines the overall training time of the proposed parallel method.

Table 2. *Results of two data partitions (with 4 clusters each) of Mandarin training data based on GMM clustering method.*

		c_1	c_2	c_3	c_4
PAR-A	# of states	1001	1558	954	456
	data %	17.3%	29.3%	43.0%	10.4%
PAR-B	# of states	1218	825	1167	771
	data %	25.5%	28.3%	26.0%	20.2%

Table 3. *Comparison between baseline single DNN and cluster-based multi-DNN in Mandarin transcription task in terms of recognition performance (CER in %) and training efficiency (time denotes average number of minutes for one epoch of DNN training).*

hidden layers		3	4	5	6
baseline	CER (%)	15.02	13.79	13.53	13.24
DNN	time (min)	45.2	48.7	52.4	56.3
multi-DNN	CER (%)	14.55	14.08	14.15	13.59
PAR-A	time (min)	12.9	15.0	17.1	19.1
(3 GPUs)	speed-up	3.5 x	3.2 x	3.1 x	2.9 x
multi-DNN	CER (%)	14.27	14.07	13.97	13.70
PAR-B	time (min)	11.1	11.5	12.9	14.5
(4 GPUs)	speed-up	4.3 x	4.5 x	4.2 x	4.0 x

Table 3 shows the detailed comparison of multi-DNN-HMM and baseline DNN-HMM. For *PAR-A*, as training data is not well balanced among different clusters, we train two DNNs related to c_1 and c_4 in turn on the same GPU and two more GPUs are used to train c_2 and c_3 in parallel. The speed-up is about 3 times faster than the baseline single DNN, at parallelization efficiency of 1.0. Given more balanced data clusters in *PAR-B*, we use 4 GPUs to train all 4 clusters in parallel. The training speed-up has surpassed 4 times, leading to over 1.0 parallelization efficiency. Results also show that both multi-DNN-HMM systems yield comparable recognition performance as the baseline DNN system, with only 1% degradation in average.

4.2. Switchboard task

We have also evaluated the proposed multi-DNN-HMM model in 309-hour Switchboard-I task. In this task, we use 39-dimension feature vector consisting of 13-d PLP features and their first and second derivatives. Cepstral mean and variance normalization (CMVN) is performed per conversation side. We first train standard cross-word triphone GMM/HMM models that use 3-state left-to-right topology with maximum likelihood criterion and then MPE is used to train the model discriminatively. All HMM states are tied to 8991 physical states, each of which contains 40 Gaussian components. A standard trigram language model, trained with all training transcripts, is used in decoding. Evaluation is performed on two standard test sets, namely Hub98 and Hub01. The recognition performances of two baseline GMM-HMM systems are reported in Table 1. The baseline GMM-HMM models are used to generate state level alignments for the following DNN training.

We then train a baseline DNN-HMM system with various number of hidden layers (from 3 to 6 layers) and each hidden layer contains 2048 hidden nodes. The training procedure of DNN-HMM is similar to that mentioned in the Mandarin task. As for multi-DNN-HMM systems, all 8991 states are first grouped into 4 clusters using the GMM clustering method. This partition is denoted as *PAR-C*, as shown in Table 4. In this task, both silence and short pause states account for over 30% of all training data. We use the same method as above to generate a more balanced partition. We first cluster the remaining states into 4 clusters and then uniformly distribute all sil and sp data into these 4 clusters. This results in a more balanced partition, denoted as *PAR-D* in Table 4.

Table 4. *Results of two data partitions (with 4 clusters each) of Switchboard training data based on GMM clustering method.*

		c_1	c_2	c_3	c_4
PAR-C	# of states	2450	3661	20	2860
	data %	45.0	25.3	9.3	20.4
PAR-D	# of states	1768	3350	2312	1573
	data %	26.3	31.1	22.6	20.0

Table 5. *Comparison between baseline single DNN and cluster-based multi-DNN in Switchboard task in terms of recognition performance (WER in %) and training efficiency (time denotes average number of hours for one epoch of DNN training).*

hidden layers		3	4	5	6
baseline	Hub98 WER(%)	33.4	32.4	31.2	30.9
DNN	Hub01 WER(%)	25.6	24.4	23.7	23.6
	time (hr)	14.7	16.1	17.7	19.3
multi-DNN	Hub98 WER(%)	33.8	32.8	32.5	32.0
PAR-C	Hub01 WER(%)	25.1	24.5	24.2	23.8
	time (hr)	2.8	3.3	4.0	4.5
(3 GPUs)	speed-up	5.3 x	4.9 x	4.4 x	4.3 x
multi-DNN	Hub98 WER(%)	34.0	33.1	32.9	32.6
PAR-D	Hub01 WER(%)	25.4	24.7	24.4	24.1
	time (hr)	2.7	3.1	3.6	4.0
(4 GPUs)	speed-up	5.4 x	5.1 x	4.9 x	4.8 x

Results in Table 5 show the proposed multi-DNN method can achieve about 4-5 speed-up in training time in *PAR-C* when 3 G-PUs are used, which indicates about 1.4 parallelization efficiency. In a more balanced data partition *PAR-D*, the multi-DNN method can achieve over 5 times of speed-up in training when 4 GPUs are used. Moreover, we can see that the multi-DNN method can yield very similar recognition performance as the baseline DNN-HMM in both test sets (Hub98 and Hub01), with only 1-2% WER degradation in average.

5. FINAL REMARKS

In this paper, we have proposed a new cluster-based multiple DNNs method for acoustic modelling in LVCSR. The new modelling method can yield comparable recognition performance as the regular DNN method but the multiple DNNs can be efficiently trained in parallel using multiple GPU devices, which leads to a very significant speed-up in training (about 4-6 times faster with 3-4 GPUs). As our next steps, we will investigate performance of the multiple DNNs method in data partitions with larger number of clusters (over 4 clusters) for even better training speedup when more GPUs are available. We will also study whether it is possible to use a smaller DNN configuration for each cluster to reduce the total model size.

6. REFERENCES

[1] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, 1993.

[2] Li Deng, "An overview of deep-structured learning for information processing," in *Proc. of Asian-Pacific Signal and Information Processing-Annual Summit and Conference (APSIPA-ASC)*, 2011.

[3] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing," *IEEE Signal Processing Magazine*, vol. 28, pp. 145–154, 2011.

[4] G. E. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, 2012.

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

[6] G. E. 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*, pp. 30–42, January 2012.

[7] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Interspeech*, 2011, pp. 437–440.

[8] G.E. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent dbn-hmms," in *ICASSP*, 2011.

[9] J. Pan, C. Liu, Z. Wang, Y. Hu, and H. Jiang, "Investigation of deep neural networks (DNN) for large vocabulary continuous speech recognition: Why DNN surpasses GMMs in acoustic modelling," in *The 8th International Symposium on Chinese Spoken Language Processing (ISCSLP-2012)*, 2012.

[10] Q. Zhu, A. Stolcke, B.Y. Chen, and N. Morgan, "Using MLP features in SRI's conversational speech recognition system," in *Interspeech*, 2005, pp. 2141–2144.

[11] G. E. Hinton, S. Osindero, and Y.W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[12] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *NIPS'06*, 2006, pp. 153–160.

[13] S. Scanzio, S. Cumani, R. Gemello, F. Mana, and P. Laface, "Parallel implementation of artificial neural network training," in *ICASSP*, 2010.

[14] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Interspeech*, 2012.

[15] Stanislav Kontár, "Parallel training of neural networks for speech recognition," in *12th International Conference on Soft Computing MENDEL*, 2006.

[16] K. Veselý, L. Burget, and F. sek Grézl, "Parallel training of neural networks for speech recognition," in *Interspeech*, 2010.

[17] J. Park, F. Diehl, M.J.F. Gales, M. Tomalin, and P.C. Woodland, "Efficient generation and use of MLP features for arabic speech recognition," in *Interspeech*, 2009.

[18] Q.V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, and A.Y. Ng, "Building high-level features using large scale unsupervised learning," in *ICML*, 2012.

[19] X. Chen, A. Eversole, G. Li, D. Yu, and F. Seide, "Pipelined back-propagation for context-dependent deep neural networks," in *Interspeech*, 2012.

[20] P. Farber, "Quicknet on multispert: Fast parallel neural network training," Tech. Rep., ICSI, 1997.