SEQUENCE TRAINING OF MULTIPLE DEEP NEURAL NETWORKS FOR BETTER PERFORMANCE AND FASTER TRAINING SPEED

Pan Zhou[1], *Lirong Dai*[1], *Hui Jiang*[2]

[1]National Engineering Laboratory of Speech and Language Information Processing,
University of Science and Technology of China, Hefei, P. R. China
[2] Department of Electrical Engineering and Computer Science, York University, Toronto, Canada
Email: pan2005@mail.ustc.edu.cn, lrdai@ustc.edu.cn, hj@cse.yorku.ca

ABSTRACT

Recently, sequence level discriminative training methods have been proposed to fine-tune deep neural networks (DNN) after the frame-level cross entropy (CE) training to further improve recognition performance of DNNs. In our previous work, we have proposed a new cluster-based multiple DNNs structure and its parallel training algorithm based on the frame-level cross entropy criterion, which can significantly expedite CE training with multiple GPUs. In this paper, we extend to full sequence training for the multiple DNNs structure for better performance and meanwhile we also consider a partial parallel implementation of sequence training using multiple GPUs for faster training speed. In this work, it is shown that sequence training can be easily extended to multiple DNNs by slightly modifying error signals in output layer. Many implementation steps in sequence training of multiple DNNs can still be parallelized across multiple GPUs for better efficiency. Experiments on the Switchboard task have shown that both frame-level CE training and sequence training of multiple DNNs can lead to massive training speedup with little degradation in recognition performance. Comparing with the state-of-the-art DNN, 4-cluster multiple DNNs model with similar size can achieve more than 7 times faster in CE training and about 1.5 times faster in sequence training when using 4 GPUs.

Index Terms— speech recognition, deep neural network (DNN), multiple DNNs, sequence training, parallel training

1. INTRODUCTION

For the past 20 years, Gaussian mixture models (GMMs) have remained as the dominant model to compute state emission probabilities of hidden Markov models (HMM) in automatic speech recognition (ASR). Recently, neural networks (NN) have revived as strong alternative acoustic model for ASR, where NN is used to calculate scaled likelihoods directly for all HMM states under a hybrid mode. When neural networks are expanded to have more hidden layers (the so-called deep neural network) and more nodes per layer (for all input, hidden and output layers), it has been shown that neural networks yield a dramatic performance gain over the conventional GMMs in almost all speech recognition tasks. At the beginning, deep neural networks (DNNs) are typically learned from concatenated speech frames in training data as well as their forced-alignment labels to distinguish different tied HMM states based on the frame-level cross entropy (CE) training criterion. However, speech recognition is a sequence classification problem in nature. It is well known that GMM-HMM based speech recognizers typically obtain notable performance gain after adjusting parameters with sequence-level discriminative training criteria [1], such as maximum mutual information (MMI) [2], minimum phone error (MPE) [3], minimum Bayes risk (MBR) [4] or large margin estimation (LME) [5]. Although cross entropy learning of deep neural networks is already a discriminative criterion, as shown in [6, 7], it may yield further improvement (about 10-15% relative error reduction) if DNN parameters are refined based on a sequence level discriminative criterion that is more closely related to speech recognition.

On the other hand, no matter what training criterion is used, it is always a very slow and time-consuming process to learn DNNs, especially from a large training data set. For example, it normally takes a few weeks to train a typical six-hidden-layer DNN from thousands of hours of speech data. The underlying reason for this is that the basic learning algorithm in the standard error back-propagation (BP) framework, namely stochastic gradient descent (SGD), is relatively slow in convergence and it is difficult to parallelize SGD because it is inherently a serial learning method. During the recent years, researchers have been pursuing various methods for more efficient DNN training. The first possible way is to simplify model structure by exploring sparseness in DNN models. As reported in [8], it results in almost no performance loss by zeroing 80% of small weights in a large DNN model. This method is pretty good to reduce total DNN model size but it gives no gain in terms of training speed due to highly random memory accesses introduced by sparse matrices. Along this line, as in [9, 10], it is proposed to factorize each weight matrix in DNN into product of two lower rank matrices, which is reported to achieve about 30-50% speedup in DNN computation. A more recent work in [11] proposes to use a shrinking hidden layers structure to simplify the DNN model and it also shows up to 50% speedup in DNN computation. Alternatively, another more straightforward way to speed up DNN training is to parallelize it using multiple GPUs or CPUs if a single thread of learning algorithm itself can not be made even faster. As in [12, 13], the so-called asynchronous SGD is proposed to use multiple computing units to parallelize DNN training in server-client mode. Moreover, the pipelined BP in [14] is another way to use multiple GPUs for parallel training of DNNs. Finally, in our previous work [15], we have proposed to use a cluster based multiple deep neural network to parallelize DNN training across multiple GPUs without involving any communication traffics among them. This method has achieved more than three times acceleration in training speed by using 3 GPUs for frame-level CE training criterion with very small performance degradation.

In this paper, we further extend the multiple DNNs (mDNN)

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).

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

model in our previous work [15] to sequence training framework for better recognition performance. We first investigate how sequence discriminative training can be applied to mDNN modelling framework. As shown in this work, sequence training of mDNN can be viewed as a joint training process to further improve performance of mDNN. Next, we also consider to implement sequence training of mDNN partially in parallel by using multiple GPUs for faster training speed. Experiments on the 320-hour Switchboard task have revealed that even one epoch of MMI-based sequence training can improve CE-trained mDNN from 15.9% to 14.8% in word error rate (about 7% relative error reduction). Meanwhile, sequence training of mDNN can be expedited by 1.5 times by exploring a partial parallel implementation in 4 GPUs. When taking the initial CE learning into account, we have achieved over 5 times training speedup with mDNN when 4 GPUs are used.

2. REVIEW OF MULTIPLE DNNS

In large vocabulary ASR tasks, it is common to have tens of thousand of tied HMM states. This results in extremely large output weight matrix that largely slows down the back-propagation process in DNN training. In [15], we have proposed a multiple deep neural network (mDNN) structure as shown in Fig. 1. By using some unsupervised clustering methods [16], we first divide the whole training set into several disjointed subsets, which have no common state labels. In this way, each DNN in the middle column of Fig. 1 is trained from each subset of training data to model only HMM states belonging to this subset. In other words, each DNN is learned to compute posterior probability of each HMM state, s_j, given the current cluster, c_i, and input data, X, i.e., $\Pr(s_j|c_i, X)$. At the same time, a smaller top-level DNN, denoted as NN_0, is trained from all training data to compute posterior probability of each cluster given the input data, $\Pr(c_i|X)$. At the end, the final output posteriori probabilities of multiple DNNs can be calculated as follows:

$$y_{rt}(s) = Pr(s_j|X) = Pr(c_i|X) \cdot Pr(s_j|c_i, X) \quad (s_j \in c_i). \tag{1}$$

This product can be directly used for decoding in the same way as DNN. See [15] for more details on the mDNN model structure and how to perform data partition to train mDNN. The advantage of mDNN is that its training process can be done in a highly parallel manner. Moreover, each DNN has much smaller model size and each DNN is learned from less training data. As a result, the training speed of mDNN can be accelerated dramatically by using multiple training threads in several GPUs.

3. CROSS ENTROPY TRAINING OF MULTIPLE DNNS

Traditional DNN-based acoustic models estimate the posterior for each tied HMM state at its output layer. DNNs are trained to optimize a given objective function, such as cross entropy between the actual output distribution and the desired target distribution, using the standard error back-propagation algorithm [17] through SGD. The output distribution is calculated using softmax activation function as:

$$y_{rt}(s) = \Pr(s|X_{rt}) = \frac{\exp\{a_{rt}(s)\}}{\sum_{s'} \exp\{a_{rt}(s')\}}, \tag{2}$$

where $a_{rt}(s)$ is the activation at the output layer corresponding to state s at time t for utterance r.

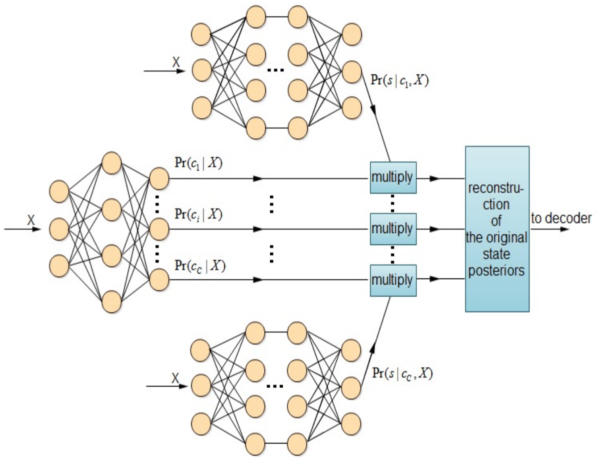

Fig. 1. Illustration of multiple DNNs for acoustic modelling.

The cross entropy (CE) objective function can be expressed as the following form:

$$\mathcal{F}_{CE} = -\sum_{r=1}^{R}\sum_{t=1}^{T_r} \log y_{rt}(s_{rt}), \tag{3}$$

where s_{rt} denotes the forced-aligned state label at time t for utterance r. In back-propagation, the most important quantity to calculate is the gradient of the CE objective function with respect to the activations at each layer, a.k.a. *error signal*. The gradients for all DNN weight parameters can be easily derived from error signals in the BP procedure. The error signal at the output layer to be back propagated to the previous layers is:

$$e_{rt}(s) = \frac{\partial \mathcal{F}_{CE}}{\partial a_{rt}(s)} = -\frac{\partial \log y_{rt}(s_{rt})}{a_{rt}(s)} = y_{rt}(s) - \delta_{rt}(s), \tag{4}$$

where $\delta_{rt}(s) = 1$ if the forced-alignment label s_{rt} is equal to s and $\delta_{rt}(s) = 0$ otherwise.

For multiple DNNs (mDNN), we use the same CE objective function and derive error signals at the output layer in a similar way. Note that $y_{rt}(s_{rt})$ in eq. (4) is calculated as in eq. (1) for mDNN. Thus, we can obtain error signals at the output layer for CE training of mDNN as follows:

$$\begin{aligned} e_{rt}(s) &= \frac{\partial \mathcal{F}_{CE}}{\partial y_{rt}(s_{rt})} \frac{\partial y_{rt}(s_{rt})}{\partial a_{rt}(s)} \\ &= \begin{cases} 0, & s \notin C_{rt} \\ y_{rt}(s_j) - \delta_{rt}(s_j), & s \in C_{rt} \\ y_{rt}(c) - \delta_{rt}(c), & s \in NN_0 \end{cases} \end{aligned} \tag{5}$$

where C_{rt} denotes the cluster that contains label s_{rt}, s_j is the state index of s in cluster C_{rt}, c is the cluster index. According to eq. (5), it is clear that each input data X_{rt} contributes zero error signal at the output layer of other DNNs that do not contain its corresponding state label. Meanwhile, for cluster C_{rt} containing its state label, it has the exactly same form as training with pattern X_{rt} in the regular DNN. This justifies that all DNNs can be trained totally independently on its own cluster data in mDNN without involving any communication traffic among them. Therefore, after clustering training data into different groups, mDNN can be trained independently with the standard BP using its own data and labels. This leads to maximum degree of parallelism.

4. SEQUENCE TRAINING OF MULTIPLE DNNS

Sequence training attempts to simulate the actual MAP decision rule in speech recognition by incorporating sequence level constraints from acoustic models, lexicon and language models. In this work, we study the sequence training of mDNN based on the maximum mutual information (MMI) criterion.

4.1. MMI Sequence Training for regular DNN

Assuming $O_r = \{o_{r1}, ..., o_{rT_r}\}$ denotes the observation sequence of utterance r, and W_r is its reference word sequence label, the MMI objective function criterion is represented as:

$$\mathcal{F}_{MMI} = \sum_r \log \frac{p(O_r|S_r)^k P(W_r)}{\sum_{W \in \mathcal{G}_r} p(O_r|S)^k P(W)}, \tag{6}$$

where $S_r = \{s_{r1}, ..., s_{rT_r}\}$ is the reference state sequence corresponding to W_r, k is the acoustic scaling factor, and in the denominator W is summed over all competing hypotheses in a word graph, $\mathcal{G}_r$. Differentiating the above objective function in eq. (6) w.r.t. log likelihood $\log p(o_{rt}|s)$ for each state s, we get:

$$\frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}|s)} = k(\gamma_{rt}^{num}(s) - \gamma_{rt}^{den}(s)), \tag{7}$$

where $\gamma_{rt}^{num}(s)$ and $\gamma_{rt}^{den}(s)$ stand for the posterior probabilities of being in state s at time t, computed for utterance r from the reference state sequence S_r and the word graph $\mathcal{G}_r$, respectively. Thus, the required error signal is calculated as follows:

$$\begin{aligned} e_{rt}(s) &= \frac{\partial \mathcal{F}_{MMI}}{\partial a_{rt}(s)} = \sum_{s'} \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}|s')} \frac{\partial \log p(o_{rt}|s')}{\partial a_{rt}(s)} \\ &= \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}|s)} - p(s|o_{rt}) \sum_{s'} \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}|s')} \end{aligned} \tag{8}$$

where s' sums over all states in the model. After some minor manipulation, it is straightforward to derive that the second term in eq. (8) equals to zero. Substituting eq. (7) in, we get the error signals at time t of utterance r for state s as $e_{rt}(s) = k(\gamma_{rt}^{num}(s) - \gamma_{rt}^{den}(s))$.

4.2. MMI Sequence Training for multiple DNNs

In this section, we derive MMI error signals for multiple deep neural network (mDNN). Let's randomly select a state s', the partial derivatives of its likelihood with respect to $a_{rt}(s)$ in mDNN is computed as:

$$\begin{aligned} \frac{\partial \log p(o_{rt}|s')}{\partial a_{rt}(s)} &= \frac{\partial (\log p(s'|o_{rt}) - \log p(s') + \log p(o_{rt}))}{\partial a_{rt}(s)} \\ = \frac{\partial \log p(s'|o_{rt})}{\partial a_{rt}(s)} &= \frac{\partial \log p(c'|o_{rt})}{\partial a_{rt}(s)} + \frac{\partial \log p(s'|c', o_{rt})}{\partial a_{rt}(s)} \\ &= \begin{cases} 0, & s' \notin c_s \\ 1 - p(s|c_s, o_{rt}), & s' = s \\ -p(s|c_s, o_{rt}), & s' \neq s, s' \in c_s. \end{cases} \end{aligned} \tag{9}$$

Therefore, for each parallel DNN in mDNN, the error signal is calculated as:

$$\begin{aligned} e_{rt}(s) &= \frac{\partial \mathcal{F}_{MMI}}{\partial a_{rt}(s)} = \sum_{s'} \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}|s')} \frac{\partial \log p(o_{rt}|s')}{\partial a_{rt}(s)} \\ &= \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}|s)} - p(s|c_s, o_{rt}) \sum_{s' \in c_s} \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_r t|s')} \end{aligned} \tag{10}$$

where $\frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}|s)}$ is calculated from eq. (7). In this case, the error signal at the output layer contains two terms. The second term is not equal to zero in mDNN since it is only summed over a subset of state labels. These error signals can be propagated in the same way as the regular BP to derive error signals for all layers.

Next, we consider to compute error signals for the top-level NN, namely NN_0. It is easy to show that the partial derivatives in eq. (9) for NN_0 take the following form:

$$\begin{aligned} \frac{\partial \log p(o_{rt}|s')}{\partial a_{rt}(c)} &= \frac{\partial \log p(c'|o_{rt})}{\partial a_{rt}(s)} + \frac{\partial \log p(s'|c', o_{rt})}{\partial a_{rt}(s)} \\ &= \begin{cases} 1 - p(c|o_{rt}), & s' \in c \\ -p(c|o_{rt}), & s' \notin c \end{cases} \end{aligned} \tag{11}$$

Therefore, we can calculate error signals at the output layer of NN_0 in the following form:

$$\begin{aligned} \frac{\partial \mathcal{F}_{MMI}}{\partial a_{rt}(c)} &= \sum_{s' \in c} \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}(s'))} - p(c|o_{rt}) \sum_{s'} \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}(s'))} \\ &= \sum_{s' \in c} \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}(s'))}. \end{aligned} \tag{12}$$

where $\frac{\partial \mathcal{F}_{MMI}}{\partial \log p(o_{rt}|s)}$ is still calculated from eq. (7). In the same way, these error signals are back-propagated to derive error signals in all layers of NN_0.

4.3. Implementation using multiple GPUs

In this paper, we use SGD to optimize the above MMI objective function as in [7] and also adopt F-smoothing in [7] to interpolate sequence level criterion with frame level criterion to ensure convergence of the SGD training process.

In our implementation, sequence training of DNNs is composed of three main steps: i) DNN forward pass: posterior probabilities of all HMM states are computed for all feature frames in each utterances as eq. (1); ii) Word graph processing: perform forward-backward algorithm in each word graph to compute statistics, $\gamma_{rt}^{den}(s)$ in eq. (7), for all HMM states; iii) DNN backpropagation: run BP to compute error signals in all DNN layers and update all DNN weights based on the corresponding error signals. In [7], these three steps are efficiently implemented in one GPU for regular DNN. For mDNN, it is straightforward to see that steps i) and iii) can be distributed to multiple GPUs to compute for all parallel DNNs independently. However, processing of word graphs in step ii) can not be efficiently parallelized across multiple GPUs and it must run in one GPU. As opposed to frame level training, the implementation of sequence training for mDNN can only be partially parallelized.

5. EXPERIMENTS

In this paper, we use the standard 320-hr Switchboard task to evaluate recognition performance and training efficiency of the proposed MMI-based sequence training for the multiple DNNs. We use the NIST 2000 Hub5 evaluation set, denoted as *Hub5e00*, to evaluate recognition performance in word error rate (WER). We use average training time (in hours) per epoch (measured in GTX690 and CUDA4.0) to compare various methods in efficiency.

5.1. Baseline systems

In Switchboard, we use PLP features (static, first and second derivatives) that are pre-processed with cepstral mean and variance normalization (CMVN) per conversation side. The baseline GMM-HMM (with 8,991 tied states and 40 Gaussians per state) is first trained based on maximum likelihood estimation (MLE) and then discriminatively trained using the MPE criterion. A trigram language model (LM) is trained using 3M words of the training transcripts and 11M words of the Fisher English Part 1 transcripts.

Table 1. *Baseline recognition performance in WER and training time per epoch in Switchboard.*

model	method	Hub5e00	Time (hr)
GMM-HMM	MLE	28.7%	-
	MPE	24.7%	-
DNN-HMM	CE	16.2%	15.0
	ReFA CE	**15.9**%	15.0
	MMI seq. training	**14.2**%	30.5

As in [18], the baseline DNN is composed of six hidden layers of 2048 hidden nodes per layer, which is pre-trained by RBM using 11 concatenated successive frames of PLP. Afterward DNN is fine-tuned by 10 epoches of frame level cross entropy (CE) training, which is followed by 10 more epoches of ReFA CE training. In ReFA CE training, DNN is further trained based on new state labels generated by CE trained DNNs. At last, the re-alignment DNN is used as the initial model for one more epoch of sequence training. In CE training, we use mini-batch of 1024 frames, and an exponentially decaying schedule for learning rates that starts from an initial learning rate of 0.002 and halves the rate each epoch from the fifth epoch. Word graphs used in sequence training are generated by decoding the training data using an unigram LM and the CE trained DNN models. Performance of these baseline HMM systems is summarized in Table 1, showing that the CE-trained hybrid DNN-HMMs can give 34.4% relative error reduction over the discriminatively trained GMM-HMMs on *Hub5e00* test set and one iteration of MMI sequence training can yield 14.2% in WER, accounting for additional 12.3% relative error reduction.

5.2. Frame-Level CE Training of Multiple DNNs

To build mDNN, we first cluster the whole training data (with 8991 HMM state labels) into 4 disjointed clusters, as shown in Table 2. This partition differs from *PAR-C* in [15] because a slightly different clustering method is used here, which leads to more balanced data partition. This partition is used to construct a 4-cluster mDNN system. In this work, we use smaller hidden layers in mDNN than those in [15]. Here each parallel DNN consists of 6 hidden layers of 1200 hidden nodes per layer and NN_0 has three hidden layers of 1200 nodes per layer. With this configuration, 4-cluster mDNN contains roughly 45.1 million weights, which is comparable with that of baseline DNN (about 40.3 million weights). For 4-cluster mDNN, all DNNs are trained independently using 4 GPUs. The results in Table 3 show that frame-level CE training of mDNN can be done extremely efficiently with multiple GPUs, yielding over 7 times speedup with only 4 GPUs. In terms of WER, 4-cluster mDNN yields 17.3% after 10 epoches of CE training, which is slightly worse than that of single DNN (16.2%). However, we have found that 4-cluster mDNN yields the same performance (15.9% in WER) as the baseline DNN after 10 more epochs of CE using new state labels re-aligned with mDNNs.

Table 2. *4-cluster data partition on Switchboard training data.*

	c_1	c_2	c_3	c_4
# of states	2553	2588	1544	2306
data (%)	19.17	18.16	46.23	16.44

Moreover, we also use the same clustering method to partition Switchboard training data into 10 clusters, which is used to build a 10-cluster mDNN containing about 91.4 million weights in total. Results in Table 3 show that 10-cluster mDNN may yield massive training speedup, up to more than 16 times faster than the baseline DNN if 10 GPUs are available for parallel training. In terms of recognition performance, 10-cluster mDNN is slightly worse than the baseline DNN. We believe 10 clusters may be too many for Switchboard since some clusters only contain less than 20 hours of training data. But it may be quite promising if we apply 10-cluster mDNN to other larger tasks with much more training data available.

5.3. MMI Sequence Training of Multiple DNNs

For sequence training of 4-cluster mDNN, as shown in Table 3, after only one epoch of sequence training, WER is reduced from 15.9% down to 14.5%, about 8.8% relative error reduction, which is only slightly worse than performance of baseline DNN after sequence training (14.2%). On the other hand, if running sequence training of 4-cluster mDNN in 4 GPUs, the training time per epoch (measured based on simulation) is about 20.4 hours, equivalent to about 1.5 times faster than the baseline. If we consider the total training time from scratch, including 10 epochs of CE, 10 epochs of ReFA CE and 1 epoch of sequence training, the overall training speedup is about 5.2 times faster than the baseline DNN.

Table 3. Performance comparison of mDNN vs. DNN using various training methods in terms of WER (%) and training time per epoch and training speedup over DNN with 1 GPU. (* measured based on simulation)

model		CE	CE (ReFA)	MMI seq. tr.
DNN	WER	16.2%	15.9%	14.2%
	time (hr)	15.0	15.0	30.5
4-cluster mDNN (4 GPUs)	WER	17.3 %	**15.9**%	**14.5**%
	time (hr)	2.1	2.1	20.4 *
	speedup	**7.1 x**	**7.1 x**	1.5 x (**5.2 x**)
10-cluster mDNN (10 GPUs)	WER	17.4%	16.7%	15.5%
	time (hr)	0.9	0.9	23.7 *
	speedup	**16.3 x**	**16.3 x**	1.3 x (**8.0 x**)

6. FINAL REMARKS

In this paper, we have studied the MMI based sequence training for multiple DNNs in LVCSR for better performance and faster training speed. Experiments on Switchboard have shown that the proposed mDNN modelling structure may lead to significant training speed up by using multiple GPUs. Meanwhile, after frame-level cross entropy training and sequence training, mDNN models may yield comparable recognition performance as the baseline DNN. The proposed mDNN structure is quite promising for even larger ASR tasks where enormous amount of training data is available.

7. REFERENCES

[1] Hui Jiang, "Discriminative training for automatic speech recognition: A survey," *Computer and Speech, Language*, vol. 24, no. 4, pp. 589–608, 2010.

[2] V. Valtchev, J. J. Odell, P. C. Woodland, and S. Young, "MMIE training of large vocabulary recognition systems," *Speech Communication*, vol. 22, no. 4, pp. 303–314, 1997.

[3] Daniel Povey, "Discriminative training for large vocabulary speech recognition," *Cambridge, UK: Cambridge University*, vol. 79, 2004.

[4] Matthew Gibson and Thomas Hain, "Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition.," in *INTERSPEECH*, 2006.

[5] Xinwei Li and Hui Jiang, "Solving large margin HMM estimation via semidefinite programming," in *Proc. of International Conference on Spoken Language Processing (ICSLP)*, 2006.

[6] Brian Kingsbury, Tara Sainath, and Hagen Soltau, "Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization.," in *INTERSPEECH*, 2012.

[7] Hang Su, Gang Li, Dong Yu, and Frank Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[8] Dong Yu, Frank Seide, Gang Li, and Li Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 4409–4412.

[9] Tara Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6655–6659.

[10] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. of Interspeech*, 2013.

[11] Shiliang Zhang, Yebo Bao, Pan Zhou, Hui Jiang, and Lirong Dai, "Improving deep neural networks for LVCSR using dropout and shrinking structure," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[12] 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.

[13] Shanshan Zhang, Ce Zhang, Zhao You, Rong Zheng, and Bo Xu, "Asynchronous stochastic gradient descent for dnn training," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6660–6663.

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

[15] Pan Zhou, Cong Liu, Qingfeng Liu, Lirong Dai, and Hui Jiang, "A cluster-based multiple deep neural networks method for large vocabulary continuous speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6650–6654.

[16] Pan Zhou, Lirong Dai, Hui Jiang, Yu Hu, and Qingfeng Liu, "A state-clustering based multiple deep neural networks modelling approach for speech recognition," *submitted to IEEE Trans. on Audio, Speech and Language Processing*, November 2013.

[17] D. E. Rumelhart, Geoffrey E Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[18] Jia Pan, Cong Liu, Zhiguo Wang, Yu Hu, and Hui Jiang, "Investigation of deep neural networks (DNN) for large vocabulary continuous speech recognition: Why DNN surpasses GMMs in acoustic modeling," in *Chinese Spoken Language Processing (ISCSLP), 2012 8th International Symposium on*, 2012, pp. 301–305.