Distributed Learning of Multilingual DNN Feature Extractors using GPUs

Yajie Miao, Hao Zhang, Florian Metze

Language Technologies Institute, School of Computer Science, Carnegie Mellon University
Pittsburgh, PA, USA

{ymiao,haoz1,fmetze}@cs.cmu.edu

Abstract

Multilingual deep neural networks (DNNs) can act as deep feature extractors and have been applied successfully to cross-language acoustic modeling. Learning these feature extractors becomes an expensive task, because of the enlarged multilingual training data and the sequential nature of stochastic gradient descent (SGD). This paper investigates strategies to accelerate the learning process over multiple GPU cards. We propose the *DistModel* and *DistLang* frameworks which distribute feature extractor learning by models and languages respectively. The time-synchronous DistModel has the nice property of tolerating infrequent model averaging. With 3 GPUs, DistModel achieves 2.6× speed-up and causes no loss on word error rates. When using DistLang, we observe better acceleration but worse recognition performance. Further evaluations are conducted to scale DistModel to more languages and GPU cards.

Index Terms: Deep neural networks, distributed learning, automatic speech recognition

1. Introduction

DNNs are evolving into the state of the art for acoustic modeling on large vocabulary continuous speech recognition (LVCSR) [1, 2] tasks. In DNN-HMM hybrid systems, neural networks with multiple hidden layers are trained to estimate the posterior probabilities of HMM context-dependent states. Likelihoods of speech frames are derived from these posteriors and state priors to replace Gaussian mixture model (GMM) likelihoods [1]. Compared with GMMs, DNN acoustic models tend to have more parameters and higher complexity. With adequate training data, DNNs have shown superior performance than GMMs [1, 2, 3]. However, when transcribed speech becomes highly limited (e.g., only several hours), the large parameter space of DNNs may lead to degradation on unseen testing data. Thus, we are likely to observe drastically different recognition performance between rich-resource and low-resource languages [4, 5].

Speech recognition on a low-resource target language can be enhanced by taking advantage of external rich-resource languages. For example, [6, 7] realize cross-language knowledge transfer either through pre-training the target-language DNN with the source languages, or through initializing its parameters with a network fine-tuned on another language. An alternative proposal is motivated by the feature learning formulation about DNNs [8]: hidden layers are treated as a series of nonlinear transformations that convert the original input features to high-level representations; a softmax layer is finally added to perform classification with respect to HMM states. Following this idea, multilingual DNNs are trained collaboratively over the source languages, with their hidden layers shared across languages [9]. On the target language, these shared layers are taken as a feature extractor which is intrinsically language-independent.

Previous work [5, 9] has reported the effectiveness of aforementioned feature extractors in addressing low-resource acoustic modeling. Training such feature extractors relies on the sequential SGD algorithm. When multiple rich-resource source languages (e.g., English and Mandarin) are available, feature extractor learning can be prohibitive even with GPU-based implementations. In this paper, we focus on accelerating the learning process. Our goal is to scale feature extractor learning to large datasets and diverse languages. We propose two distribution schemes to parallelize the learning task over multiple GPUs. In the first scheme DistModel, each GPU trains an instance of the feature extractor over a portion of the whole training data. The parallel model instances are averaged periodically after a predefined number of mini-batches. The second scheme DistLang distributes learning by languages. Separate feature extractors are trained on individual languages without any communication between GPUs. On the target language, outputs from these language-specific extractors are fused into the final feature representation.

Parallelized training of DNN acoustic models has been investigated thoroughly under monolingual settings [10, 11, 12, 13]. In [14], the authors parallelize multilingual DNN training over CPUs based on the DistBelief framework [15, 16]. We concentrate on multi-GPU training, and are particularly interested to analyze how parallelism affects the quality of the learned feature extractors. Both DistModel and DistLang are evaluated on the multilingual BABEL corpus. The performance of feature extractors is measured by WERs on the target language. A salient observation from the experiments is that DistModel can tolerate fairly infrequent model averaging, e.g., 2000 mini-batches between two consecutive averaging operations. As a result, the synchronizing delay is alleviated greatly, despite the fact that DistModel is time-synchronous. With DistModel deployed over 3 GPUs, feature extractor learning becomes 2.6× faster than using a single GPU, while giving the identical WER as single-thread training. In comparison with DistModel, DistLang achieves better speed-up but worse WER. Larger-scale evaluations further reveal the scalability of DistModel to more languages and GPUs.

2. DNN Feature Extractors

Figure 1(a) depicts the learning of DNN-based feature extractors over a group of source languages [4, 5, 17]. The hidden layers of the multilingual DNNs are shared across all the languages. Each language has its own output layer to classify context-dependent states. Fine-tuning of the DNNs is carried out using the standard mini-batch based SGD. The difference is that each epoch traverses data from all the source languages, instead of one single language. The SGD estimator loops over languages iteratively, each time picking one mini-batch from a language. Also, it switches to the softmax layers and class labels corresponding to the language from which the current mini-batch comes. Parameters of the shared layers are updated with gradients accumulated from all the languages.

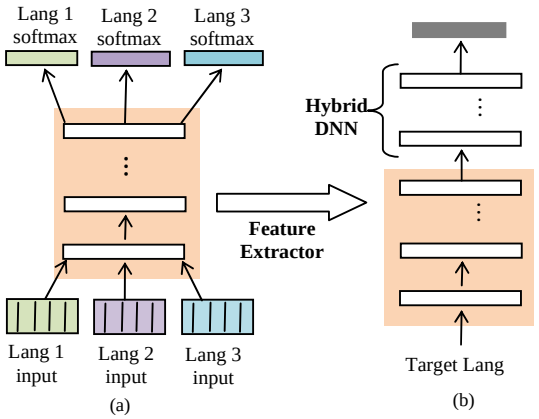

Figure 1. (a) *Learning of feature extractors over source languages*; (b) *applying this extractor to the target language for cross-language hybrid system building*.

When the multilingual DNNs are trained, the shared hidden layers serve as a language-universal feature extractor [9]. On the target language, as shown in Figure 1 (b), a hybrid system is built using features generated from this extractor, instead of the raw acoustic features (e.g., MFCCs). The DNN in this hybrid system is trained to estimate posterior probabilities of the target-language HMM states. This cross-language acoustic modeling approach enables knowledge transfer across languages and thus improves the recognition performance on the target language, especially when the target language has limited transcribed speech [5, 18]. This differs from [9] in that on the target language, we are building a fully-connected DNN model, while [9] re-estimates a single softmax layer over the extracted features.

3. Distributed Feature Extractor Learning

Ideally, feature extractors introduced in Section 2 are trained over rich-resource languages with adequate data. However, training based on SGD is sequential and hard to be parallelized. This makes feature extractor learning an expensive task, even with the powerful GPU cards. We aim at distributing computation over multiple GPUs, and two parallelization schemes are presented to accomplish this.

3.1. DistModel: Distribution by Models

Model averaging [10, 19, 20] has been exploited on distributed learning problems, for both convex and non-convex models. We port this idea to distributed training of multilingual DNN feature extractors on GPUs. The implementation is straightforward. Training data of each language is partitioned evenly across all the GPU threads. Suppose that sources include 3 languages each of which contains 100 hours of training data. When distributing training to 4 GPUs, we assign to each GPU 75 hours of data, i.e., 25 hours from each source language. Different GPUs have no overlapping on their data. Then, each GPU trains the feature extractor as described in Section 2. After a specified number of mini-batches, feature extractor instances from the individual GPUs are averaged into a unified model. We refer to the number of mini-batches between two consecutive averaging operations as *averaging interval*. Note that both the language independent (shared hidden layers) and the language specific (softmax layer) parameters are averaged. The averaged parameters are sent back to each GPU as the new starting model for the subsequent training.

DistModel is inherently time synchronous in that the parallel threads have to wait for each other to perform model averaging. This tends to cause delay, especially when the frequency of model averaging is high or some computing nodes run slowly. Compared with the more popular asynchronous methods [11, 13, 14, 15, 16, 21], time synchronous methods generally achieve worse acceleration. However, we discover that on this particular feature learning task, DistModel is robust to large averaging interval up to 2000 mini-batches. This is partly because multitask learning performed by each GPU prevents local optima on the multilingual DNN models. The averaged feature extractor still provides unbiased feature representations, even after SGD has processed many mini-batches of training examples on each GPU. Because of this, delay resulting from model averaging ends up to be a tiny fraction of the entire training time. We are confident to think that DistModel has comparable efficiency with asynchronous implementations. This is also demonstrated in Section 5 by comparing DistModel with previously reported results [12, 13].

3.2. DistLang: Distribution by Languages

Another way for distributed learning is to train the feature extractors independently on individual source languages. Each GPU uses the complete data from one language and trains the normal DNN model. Figure 2 depicts the idea of DistLang by showing N source languages which translate to N separate feature extractors after DNN training. On the target language, each speech frame is fed into these extractors. The N separate feature representations are fused to form the input into the target-language hybrid DNN. We apply two methods for this feature fusion.

Assume that the feature dimension emitted from each extractor is D. Our first method *FeatConcat* borrows the idea of MLP feature merging proposed in [22, 23, 24]. We concatenate outputs from the language-specific feature extractors into a single vector which has the dimension of $N \times D$. In the second method *FeatMix*, we fuse the feature representations via a linear weighted combination. More formally, given the target-language input feature $\mathbf{o}_t$, the feature representation from the n-th feature extractor is denoted as $f_n(\mathbf{o}_t)$. The combined feature vector can be computed as

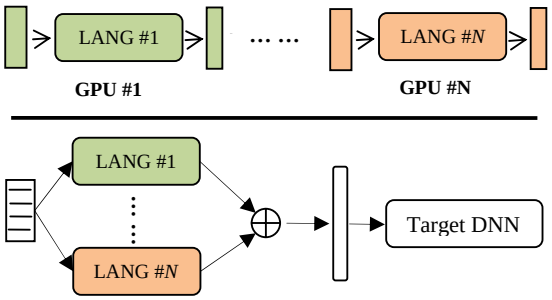

Figure 2. *DistLang for distributed feature extractor learning. Top: each GPU trains a feature extractor on each language. Bottom: on the target language, outputs from individual extractors are fused into a unified feature representation.*

$$\mathbf{c}_t = \mathbf{b} + \sum_{n=1}^{N} \mathbf{a}_n \otimes f_n(\mathbf{o}_t) \quad (1)$$

where the *D*-dimensional vector $\mathbf{a}_n$ contains the mixture weights for the *n*-th extractor, **b** is a *D*-dimensional bias vector, and $\otimes$ represents element-wise product. We do not define the values of $\mathbf{a}_n$ and **b** in advance. Instead, they can be learned during training of the target-language DNN through back-propagation (BP). If each hidden layer of the target-language DNN has *M* units, then the sizes of the input layer weight matrix are *NDM* and (*N*+1+*M*)*D* for FeatConcat and FeatMix respectively. Therefore, this FeatMix method helps to shrink the parameter space of the target-language DNN model. We expect FeatMix to be particularly advantageous when we have many source languages (i.e., a large *N*) and the target language has highly limited training data.

A notable difference between DistModel and DistLang is that the parallel GPU jobs in DistLang are completely independent without any mutual communication. DistLang incurs no delay cost from thread synchronization. Also, when a new source language becomes available, we only need to train the language-specific feature extractor on this new language. However, in order for inclusion of a new source language, DistModel has to retrain the entire feature extractor from scratch. One caveat of DistLang lies in the limitation that the number of GPUs is hardcoded by the number of source languages. In comparison, DistModel is more flexible and can make use of an arbitrary number of GPUs.

4. Experiments

4.1. Experimental Setup

Our experiments are conducted on the multilingual BABEL corpus collected under the BABEL research program [5, 25, 26, 27, 28]. The corpus has covered a wide range of languages including Tagalog, Pashto, Bengali, etc. The full language pack (FullLP) of each language contains around 80 hours of telephone conversational speech for training and 10 hours for decoding. On each language, there is also a low-resource 10HrLP condition under which only 10 hours of transcribed speech are allowed to be used for system building. We take the 10HrLP condition of Tagalog (IARPA-babel106-v0.2f) as the target language. The source languages include the FullLP sets of Cantonese (IARPA-babel101-v0.4c), Turkish (IARPA-babel105b-v0.4) and Pashto (IARPA-babel104b-v0.4aY). We measure the quality of multilingual DNN feature extractors based on WERs of hybrid systems on the target language. For fast turnarounds, we select 2 hours of speech from the 10-hour decoding data as the testing set. Decoding runs use a trigram language model built from the 10-hour training transcripts.

On each language, we build the GMM-HMM system with the same recipe. An initial maximum likelihood model is first trained using PLP+delta+acceleration features with mean normalization. Then 9 frames of PLPs are spliced together and projected down to 40 dimensions by linear discriminant analysis (LDA). A maximum likelihood linear transform (MLLT) is applied on the LDA features and generates the LDA+MLLT model. Finally, to deal with speaker variability, speaker adaptive training (SAT) is performed using feature-space maximum likelihood linear regression (fMLLR) [29]. On each language, class labels for speech frames are generated by its SAT GMM-HMM through forced alignment.

4.2. Results of Baseline Feature Extractors

Both monolingual and multilingual DNN training follows the similar configuration as [5] and is performed using the Kaldi+PDNN framework [30]. DNN inputs are 11 consecutive frames of 30-dimensional log-scale filterbanks with per-speaker mean and variance normalization. Fine-tuning starts from an initial learning rate (e.g., 0.08) which keeps unchanged for 15 epochs. Then the learning rate is halved at each epoch until the frame accuracy on a held-out validation set stops to improve. It's worth pointing out that we are experimenting with a highly low-resource condition. Also, the data collection covers a variety of environments, speaking styles and dialects. All these factors render speech recognition on Tagalog 10HrLP a very challenging task [5, 25, 26]. With 10HrLP, the monolingual DNN hybrid system has the WER of 65.8% on the 2-hour testing set. With the baseline feature extractor trained on a single GPU, we are able to reduce the WER down to 59.6%. That is, cross-language acoustic modeling described in Section 2 brings 9.4% relative improvement. This feature extractor has hidden layers with 1024 units and thus generates 1024-dimensional high-level feature representations.

4.3. Results of DistModel

Now that we have 3 source languages, 3 GPUs are employed for fair comparison between DistModel and DistLang. On the target language, we use the identical DNN topology, i.e., 4 hidden layers each with 1024 units, for hybrid system building over various feature extractors. A key variable in the DistModel scheme is the averaging interval (Section 3.1). Figure 3(a) shows the target-language WERs when DistModel adopts various averaging interval values. In Figure 3(b), acceleration coming from parallelization is quantified by the speed-up, that is, the ratio of the training time taken using a single GPU to the time using 3 GPUs. As expected, with larger averaging interval, we obtain monotonically better speed-up because of less model averagings. The change of WER displays more fluctuation, especially for averaging interval

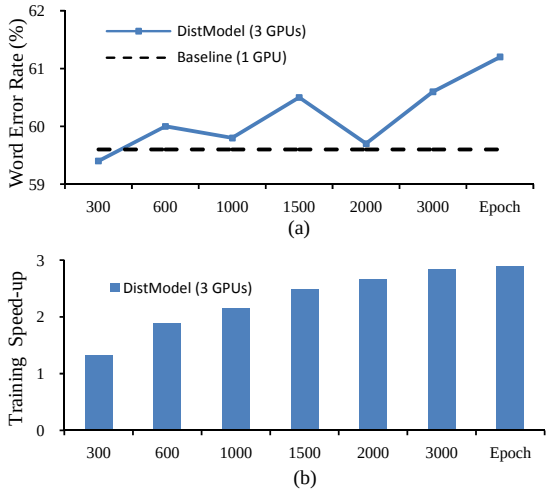

Figure 3. *Impact of averaging interval on (a) recognition performance and (b) training speed-up. WER(%) is reported on the Tagalog 2-hour testing set. "Epoch" means that averaging happens when each epoch ends.*

below 2000. When averaging interval equals 2000, feature extractor learning with 3 GPUs is 2.6× faster than using a single GPU, with the WER of 59.7% on the target language. This corresponds to 0.1% absolute degradation which can be considered negligible given the baseline 59.6%. Continuing to increase averaging interval gives further speed-up but significantly worse WERs. Thus, setting averaging interval to 2000 seems to be a good balance between training efficiency and recognition performance. A contrast experiment is to train the feature extractor with a single GPU and only one third of the data. In this case, we can get perfectly 3× speed-up. However, the WER on the target language goes up to 62.2%.

To investigate DistModel more closely, we apply DistModel to the Tagalog FullLP set for the normal monolingual DNN training. The resulting DNN model is used directly for hybrid system decoding, rather than for feature extraction. Table 1 shows that in this scenario, DistModel achieves similar speed-up as for multilingual DNN training. However, WER degradation caused by parallelization becomes more obvious compared with Figure 3. This demonstrates that DistModel is particularly suitable for multilingual feature extractor learning.

4.4. Results of DistLang

When switching to DistLang, we train the DNN feature extractor on each of the 3 source languages. Since these monolingual extractors are trained independently, this approach has approximately 3× speed-up. These extractors can adopt the same hidden layer structure as the multilingual extractors. In this case, feature representations from FeatConcat have the dimension of 1024×3, while FeatMix still generates 1024-dimensional features. An alternative setting is to insert a bottleneck-like layer, which has 341 units, in each monolingual extractor. Then, feature dimensionality of FeatConcat remains consistent with the multilingual extractors. Table 2 compares FeatConcat and FeatMix in terms of WER on the target language. Both FeatConcat and FeatMix perform worse than DistModel. Under the two dimension settings, the FeatConcat method outperforms FeatMix, even though the improvement is minor. We think the reason is that the linear combination in FeatMix limits the power of the transform applied to input features. As discussed in Section 3.2, we expect more gains to be achieved by FeatMix when the target language has even less training data (e.g., only 1 hour).

5. Larger-Scale Evaluations

In this section, we extend DistModel to larger-scale evaluations by adding Tagalog (IARPA-babel106-v0.2f) and Vietnamese (IARPA-babel101-v0.4c) into the source languages. This finally gives us 460 hours of speech data for feature extractor training. Our target language now is the 10HrLP condition of Bengali (IARPA-babel103b-v0.4b). Also, 2 hours of speech from the Bengali decoding data are selected as the testing set. DistModel adopts the optimal configuration found in Section 4.3. Table 3 shows how DistModel performs when we scale it up to 5 GPUs. We observe consistent acceleration, although the speed-up fails to improve linearly with the number of GPUs. Meanwhile, pooling more GPUs into distributed learning causes WER degradation, which is likely to be mitigated by further optimization (e.g., learning rate, feature dimension) on DistModel.

Table 1. *Performance of DistModel on Tagalog FullLP DNN training. We increase averaging interval from 300 to 3000. Parallelization uses 3 GPUs, and WER(%) is reported on the Tagalog 2-hour testing set.*

Method	WER(%)	Training speed-up
Single GPU (baseline)	49.3	---
DistModel – 300	50.1	1.5
DistModel – 600	50.5	1.9
DistModel – 1000	50.5	2.2
DistModel – 2000	50.3	2.5
DistModel – 3000	50.8	2.7

Table 2. *Comparison of FeatConcat and FeatMix within DistLang. WER(%) is reported on the 2-hour testing set.*

Method	Feature dimension	WER(%)
DistLang - FeatConcat	1024	61.4
DistLang – FeatMix	1024	61.6
DistLang - FeatConcat	341	60.3
DistLang – FeatMix	341	60.7

Table 3. *Performance of DistModel with 5 source languages. Distributed training uses 3, 4 and 5 GPUs respectively. WER(%) is reported on the Bengali 2-hour testing set.*

Method	WER(%)	Training speed-up
Monolingual DNN	72.5	---
Single GPU (baseline)	65.7	---
DistModel with 3 GPUs	66.2	2.4
DistModel with 4 GPUs	66.7	3.1
DistModel with 5 GPUs	66.8	3.4

6. Acknowledgments

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD / ARL) contract number W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

7. Conclusions and Future Work

In this paper, we present two effective schemes, DistModel and DistLang, to train multilingual DNN feature extractors in a distributed manner. These two strategies distribute computation by models and languages respectively. In our experiments with the BABEL corpus, DistModel is robust to infrequent model averaging and shows nice training speed-up. In comparison, the DistLang scheme is characterized by better acceleration but worse WER on the target language. In our future work, we will further examine FeatConcat and FeatMix by adding more languages as the sources and reducing the target-language training data. Also, we are interested to extend DistModel to deep convolutional networks (DCNs) [17, 31, 32], and study the efficient training of multilingual feature extractors with convolution layers.

8. References

[1] 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*, vol. 20(1), pp. 30-42, 2012.

[2] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, pp. 24–29, 2011.

[3] Y. Miao, H. Zhang, and F. Metze, "Towards speaker adaptive training of deep neural network acoustic models," to appear in *Proc. Interspeech*, 2014.

[4] Y. Miao, and F. Metze, "Improving low-resource CD-DNN-HMM using dropout and multilingual DNN training," in *Proc. Interspeech*, pp. 2237-2241, 2013.

[5] Y. Miao, F. Metze, and S. Rawat, "Deep maxout networks for low-resource speech recognition," in *Proc. ASRU*, 2013.

[6] P. Swietojanski, A. Ghoshal, and S. Renals, "Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR," in *Proc. SLT*, pp. 246-251, 2012.

[7] N. T. Vu, W. Breiter, F. Metze, and T. Schultz, "An investigation on initialization schemes for multilayer perceptron training using multilingual data and their effect on ASR performance," in *Proc. Interspeech*, 2012.

[8] D. Yu, M. L. Seltzer, J. Li, J. Huang, and F. Seide, "Feature learning in deep neural networks – studies on speech recognition tasks," in *International Conference on Learning Representations 2013*.

[9] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *Proc. ICASSP*, pp. 7304-7308, 2013.

[10] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Proc. ICASSP*, 2014.

[11] T. N. Sainath, B. Kingsbury, H. Soltau, and B. Ramabhadran, "Optimization techniques to improve training speed of deep neural networks for large speech tasks," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 11, pp. 2267-2276, 2013.

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

[13] S. Zhang, C. Zhang, Z. You, R. Zheng, and B. Xu, "Asynchronous stochastic gradient descent for DNN training," in *Proc. ICASSP*, pp. 6660-6663, 2013.

[14] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *Proc. ICASSP*, pp. 8619-8623, 2013.

[15] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," in *Proc. NIPS*, 2012.

[16] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng, "Building high-level features using large scale unsupervised learning," in *Proc. International Conference on Machine Learning*, pp. 81–88, 2012.

[17] Y. Miao, and F. Metze, "Improving language-universal feature extraction with deep maxout and convolutional neural networks," to appear in *Proc. Interspeech*, 2014.

[18] Y. Miao, F. Metze, and A. Waibel, "Subspace mixture model for low-resource speech recognition in cross-lingual settings," in *Proc. ICASSP*, pp. 7339-7342, 2013.

[19] G. Mann, R. Mcdonald, M. Mohri, N. Silberman, and D. D. Walker, "Efficient large-scale distributed training of conditional maximum entropy models," in *Proc. NIPS*, 2009.

[20] A. Asuncion, P. Smyth, and M. Welling, "Asynchronous distributed learning of topic models," in *Proc. NIPS*, 2012.

[21] M. A. Zinkevich, A. Smola, M. Weimer, L. Li, "Parallelized stochastic gradient descent," in *Proc. NIPS*, 2010.

[22] J. Park, F. Diehl, M.J.F. Gales, M. Tomalin, and P.C. Woodland, "The efficient incorporation of MLP features into automatic speech recognition systems," *Computer Speech and Language*, volume 25, issue 3, pp. 519–534, 2011.

[23] F. Valente, M. M. Doss, C. Plahl, S. Ravuri, and W. Wang, "Transcribing Mandarin broadcast speech using multi-layer perceptron acoustic features," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 8, pp. 2439-2450, 2011.

[24] Y. Qian, and J. Liu, "Cross-lingual and ensemble MLPs strategies for low-resource speech recognition," in *Proc. Interspeech*, 2012.

[25] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Proc. ICASSP*, pp. 3377-3381, 2013.

[26] J. Gehring, W. Lee, K. Kilgour, I. Lane, Y. Miao, and A. Waibel, "Modular Combination of Deep Neural Networks for Acoustic Modeling," in *Proc. Interspeech*, pp. 94-98, 2013.

[27] J. Chiu, and A. Rudnicky, "Using conversational word bursts in spoken term detection," in *Proc. Interspeech*, 2013.

[28] J. Chiu, Y. Wang, J. Trmal, D. Povey, G. Chen, and A. Rudnicky, "Combination of FST and CN search in spoken term detection," to appear in *Proc. Interspeech*, 2014.

[29] M. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[30] Y. Miao, "Kaldi+PDNN: building DNN-based ASR systems with Kaldi and PDNN," arXiv:1401.6984, 2014.

[31] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Proc. ICASSP*, pp. 4277-4280, 2012.

[32] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. ICASSP*, pp. 8614-8618, 2013.