Open access • Proceedings Article • DOI:10.1109/ICASSP.2014.6854672

Asynchronous stochastic optimization for sequence training of deep neural networks

Georg Heigold, Erik McDermott, Vincent Vanhoucke, Andrew W. Senior ...+1 more authors

Institutions: Google

Published on: 04 May 2014 - International Conference on Acoustics, Speech, and Signal Processing

Topics: Stochastic optimization, Asynchronous communication, Asynchrony (computer programming) and Frame (networking)

ASYNCHRONOUS STOCHASTIC OPTIMIZATION FOR SEQUENCE TRAINING OF DEEP NEURAL NETWORKS

Georg Heigold, Erik McDermott, Vincent Vanhoucke, Andrew Senior, Michiel Bacchiani

Google Inc., USA

ABSTRACT

This paper explores asynchronous stochastic optimization for sequence training of deep neural networks. Sequence training requires more computation than frame-level training using pre-computed frame data. This leads to several complications for stochastic optimization, arising from significant asynchrony in model updates under massive parallelization, and limited data shuffling due to utterance-chunked processing. We analyze the impact of these two issues on the efficiency and performance of sequence training. In particular, we suggest a framework to formalize the reasoning about the asynchrony and present experimental results on both small and large scale Voice Search tasks to validate the effectiveness and efficiency of asynchronous stochastic optimization.

Index Terms— speech recognition, acoustic modeling, neural networks, sequence training, asynchronous stochastic optimization

1. INTRODUCTION

Deep neural networks (DNNs) have become the dominant acoustic model for speech recognition [1, 2, 3, 4, 5]. Two frameworks to incorporate DNNs into the existing HMM-based decoder include the hybrid [6] and tandem [7] approaches. In this paper, we focus on the hybrid approach.

DNNs are bootstrapped using a frame-level training criterion such as cross-entropy (CE) [1, 2, 3, 4, 5], optimizing the frame error of isolated frames. This is mismatched with the objective of interest, word error, which is a sequence-based criterion. Directly minimizing the word error is a hard optimization problem and thus, several surrogates have been proposed, including maximum mutual information (MMI) [8], minimum phone error (MPE) [9] or state-level minimum Bayes risk (sMBR) [10]. Good gains have recently been reported for sequence training of DNNs [10, 11, 12, 13].

Stochastic gradient descent (SGD) [14] is the most commonly used optimization procedure for frame training of DNNs [1, 2, 3, 4, 5]. Unfortunately, optimization in SGD is an inherently sequential process that is difficult to distribute over many machines. This in turn poses practical challenges when scaling up to large datasets. One approach to speeding up sequential optimization of DNNs has been to use a GPU-based implementation [1, 2, 3, 4, 5].

It is not immediately clear how to implement sequence training using a stochastic optimization process [12, 13]. For stable SGD optimization, the observations need to be randomized (shuffled) [12, 14]. This is in contrast to the computation of a gradient that is utterance-derived, and hence sequential in nature and hard to shuffle. Batch optimization schemes are a more natural match for this type of objective [4, 10, 11, 15]. For a direct comparison of frame/sequence training and to avoid adding a sequence training-specific optimization algorithm or duplicating large parts of the GPU/CPU implementation, we would like to retain the parallelization (and as a result the scalability to large data sets) that we obtain from using our DistBelief software framework [16, 17, 18]. In particular, retaining good scalability through parallelism inevitably brings asynchrony. Fig. 1 depicts the extended version of the architecture adopted here.

Asynchronous stochastic optimization has been used before for sequence training, although in a rather limited and controlled way [12]. Overall, it is an emerging, poorly understood topic. In particular, the authors are not aware of an explicit study of the tradeoffs related to:

- Parameter asynchrony (gradient computations take time during which the model parameters change in the parallelized optimization), and
- Data shuffling (gradient computations are per utterance which limits data randomization).

This paper addresses these two issues and provides an empirical study (Section 4) of asynchronous stochastic optimization in DistBelief (Section 3) for sequence training of DNNs (Section 2). Furthermore, the proposed optimization algorithm is shown to use an auxiliary function, which allows for generalized Expectation-Maximization optimization under certain assumptions (Section 3.3). The conclusions can be found in Section 5.

2. DEEP NEURAL NETWORKS IN ASR

Let $X = x_1, \ldots, x_T$ denote a sequence of T feature vectors and W a word sequence. According to the HMM assumption, the probability for acoustic models is decomposed as follows:

$$p(X|W) = \sum_{s_1,\ldots,s_T \in W} \prod_{t=1}^{T} p(x_t|s_t)p(s_t|s_{t-1}),$$

where the marginalization is over the HMM states $s_1, \ldots, s_T$ representing the given word sequence W. In the hybrid modeling approach, the emission probability is represented as $p(x|s) = p(s|x)p(x)/p(s)$ (Bayes rule). The state posterior $p(s|x)$ is estimated with a static classifier such as a DNN [1, 2, 3, 4, 5]. The state prior $p(s)$ is the relative state frequency. The data likelihood $p(x)$ does not depend on state s and thus can be ignored for decoding/lattice generation and forced alignment [6]. The model parameters θ comprise the DNN weights and biases, estimated by maximizing the cross-entropy (CE) on all utterances u and frames t

$$F_{CE}(\theta) = \frac{1}{T} \sum_u \sum_{t=1}^{T_u} \sum_s l_{ut}(s) \log p_\theta(s|x_{ut}). \quad (1)$$

Here, $T = \sum_u T_u$ is the total number of frames. The targets are set to $l_{ut}(s) = \delta(s, s_{ut})$ for fixed state alignments $s_{u1}, \ldots, s_{uT_u}$, where δ denotes the Kronecker delta.

An example of a training criterion for sequence training is maximum mutual information (MMI) [8]:

$$F_{MMI}(\theta) = \frac{1}{T} \sum_u \log \frac{p_\theta(X_u|W_u)^\kappa p(W_u)}{\sum_W p_\theta(X_u|W)^\kappa p(W)}. \quad (2)$$

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

For sequence training, a weak language model (here, a unigram language model) is used and κ, the reciprocal of the language model weight, is attached to the acoustic model. The logarithm diverges if the argument goes to zero, i.e., if the correct word sequence has zero probability in decoding. To avoid numerical issues with such utterances, we use the frame rejection heuristic described in [13], i.e., discard frames with state occupancy close to zero, $\gamma_{ut}^{(den)}(s) < \epsilon$ (here, $\epsilon = 0.001$). No regularization (for example, ℓ_2-regularization around the initial network) or smoothing such as the H-criterion [12] is used in this paper as there is no empirical evidence for overfitting.

In contrast to CE, which optimizes a frame-level criterion, sequence training optimizes a cost function reflecting the quality of the overall recognition process given the whole utterance context. Though there are training criteria that are better correlated with test word error than MMI [11, 13], in this paper we shall focus only on MMI.

The gradient of the training criterion can be written as

$$\nabla F_{MMI}(\theta) = \frac{1}{T} \sum_u \sum_{t=1}^{T_u} \sum_s \kappa[\gamma_{\theta,ut}^{(num)}(s) - \gamma_{\theta,ut}^{(den)}(s)] \times \nabla \log p_\theta(s|x_{ut})$$

where $\gamma_{\theta,ut}^{(num/den)}(s)$ denotes the numerator/denominator state occupancy for utterance u and frame t [10, 12, 13]. Using chain rule terminology, we shall refer to $\kappa[\gamma_{\theta,ut}^{(num)}(s) - \gamma_{\theta,ut}^{(den)}(s)]$ and $\nabla \log p_\theta(s|x_{ut})$ as the outer and inner gradients, respectively. The outer gradient is similar to the targets $l_{ut}(s)$ in Eq. 1, with the important difference that the outer gradient depends on the model parameters θ while the targets are independent of θ. As a consequence, unlike the CE targets, the outer gradient cannot be pre-computed. Moreover, while the outer derivatives can be negative and sum up to zero, the CE targets are assumed to be non-negative and sum up to one. Given the sequence training outer derivatives encoded as targets, the inner gradient is computed by back-propagation in the same way as the gradient for $F_{CE}(\theta)$. For a single, synchronous parameter update (the case for SGD or batch optimization), the gradient is exact. For ASGD, however, the gradient will only be approximate as we cannot update the outer gradients over multiple parameter updates; see the more formal discussion in Section 3.3. In terms of the implementation (Section 3), the computation of the outer and inner gradients is mapped to the Speech and DistBelief modules, respectively (see Fig. 1).

3. ASYNCHRONOUS SGD & DISTBELIEF

This section briefly summarizes asynchronous SGD (ASGD) in the DistBelief framework [16, 17] as used for sequence training, with special focus on data shuffling (Section 3.1) and asynchrony issues (Section 3.2 and Section 3.3).

The basic architecture for DistBelief is as follows. We partition the training data into a number of subsets and run training with a replica of the model for each of these subsets. Models periodically update their model parameters by requesting fresh values from the parameter server, which maintains the current state of all model parameters, distributed across many machines (see Fig. 1). The models send parameter updates to the parameter server after each gradient computation. In addition, in our implementation, sequence training runs an independent decoder in the input layer of each model to generate on-the-fly lattices, and then computes the outer gradient. The decoders request fresh values from the parameter server to avoid stale outer gradients.

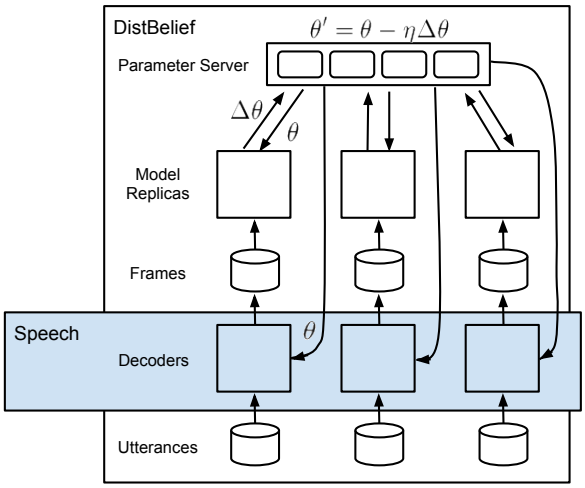

Fig. 1. Asynchronous SGD: Model replicas asynchronously fetch parameters θ and push gradients $\Delta\theta$ to the parameter server. The speech decoders independently and asynchronously fetch the parameters θ.

The next subsections address the shuffling and asynchrony issues in more detail. Shuffling and asynchrony are not independent issues, and in practice a reasonable tradeoff needs to be found. (In general, more shuffling implies more asynchrony).

3.1. On-the-Fly Shuffling

It is a well-known fact that the shuffling of training data for SGD has a significant impact on efficiency and performance. A loss in performance usually is observed when employing SGD with incompletely shuffled data, for example, a 7% relative loss in WER is reported in [12]. DistBelief and ASGD allow for better shuffling when processing utterances, since many utterances are being processed in parallel by different model replicas. Our approach to shuffling is illustrated in Fig. 2. The batch size N is constant. Each slot in the batch loads a separate utterance and processes the frames within an utterance one by one. When all frames of an utterance are consumed, another utterance is loaded. Shuffling primarily derives from two separate aspects of the setup: (1) using batches of frames from different, random utterances, as just described, and (2) running multiple model replicas independently and asynchronously on different subsets of utterances.

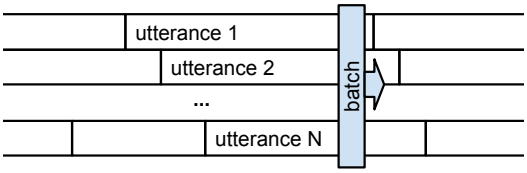

Fig. 2. On-the-fly shuffling for each model replica. A batch contains one frame from each of N utterances.

3.2. Asynchrony

The DistBelief approach sketched in Fig. 1 is fundamentally asynchronous in several aspects: the model replicas, parameter server shards and decoder all run independently of each other. Since decoding, forced alignment, and forward/backward pass to compute the outer and inner gradients take time, gradients are typically computed using stale and even inconsistent model parameters. For evaluation, additional model replicas are run to compute the statistics. The

degree of asynchrony depends on the number of model replicas and the batch size, and can be significant. For 50 replicas, the average staleness of the outer gradients is around one minute, corresponding to a few hundred DNN update steps.

3.3. Auxiliary Functions

The use of an auxiliary function allows more formal reasoning and justification of the use of ASGD optimization for sequence training. Auxiliary functions are approximations to the original training objective that are simpler to optimize. Here, "simpler" means that the auxiliary function looks like a frame-level training criterion (e.g., Eq. 1) that can be (partially) optimized with a stand-alone tool such as DistBelief. The optimization of the total training criterion is performed iteratively by updating the auxiliary function at the current parameter estimate θ' and optimizing the auxiliary function to obtain a new estimate θ. Auxiliary functions can make tangential contact with the training criterion at θ' or lie in the hypograph of the training criterion in addition [9, 19], see Fig. 3. The first type of auxiliary function is easy to construct, although little can be said about convergence. Constructive and efficient lower bounds are harder to find but lead to generalized Expectation-Maximization [19, Chapter 9.4], with stronger convergence properties, in particular convergence to a local optimum. Generally speaking, the tangential contact is only

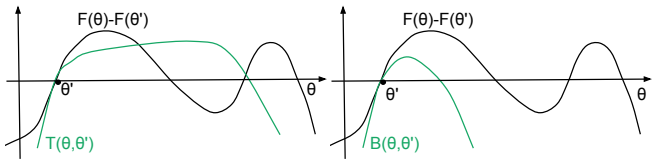

Fig. 3. Auxiliary function T making tangential contact with the training criterion F at θ' (left) vs. lower bound B for F at θ'.

locally valid and requires frequent updates of the outer gradient to guarantee stable convergence. In contrast, a lower bound is globally valid and expected to be less sensitive to frequent updates of the outer gradient.

We use the following auxiliary function for F_{MMI} (Eq. 2)

$$T_{MMI}(\theta', \theta) = \frac{1}{T} \sum_{u} \sum_{t=1}^{T_u} \sum_{s} l_{\theta',ut}(s) \log p_\theta(s|x_{ut}) + F_{MMI}(\theta')$$

with $l_{\theta',ut}(s) = \kappa[\gamma_{\theta,ut}^{(num)}(s) - \gamma_{\theta,ut}^{(den)}(s)]$ to enforce $\nabla T_{MMI}(\theta', \theta)|_{\theta=\theta'} = \nabla F_{MMI}(\theta)|_{\theta=\theta'}$.

Here, the key assumption is that the outer gradient changes more slowly than the inner gradient; see "Fetch interval", Section 4.2, for an empirical evaluation of this assumption. More details and refinements on this topic can be found in [20], including converting a tangential contact into a lower bound [20, Chapter 6.2.2] or an alternative tangential contact with better qualitative properties [20, pp.118].

4. EXPERIMENTS

An empirical study of ASGD for sequence training was conducted for a small and a large Voice Search task [21]. The small size of the Icelandic dataset allows for a thorough evaluation of the different issues and hyper-parameters while the large scale English dataset demonstrates the scalability of the algorithm.

4.1. Dataset & Setup

The training and test datasets consist of 90,000 utterances/360,000 word tokens/60 hours and 10,000 utterances/38,000 words/6 hours, respectively. They were recorded under the same conditions.

The DNN setup is based on that in [4, 18]. The input for the DNN is 26 contiguous frames (20 on the left and 5 on the right) of 40-dimensional log-filterbank features. The DNN consists of four hidden layers each of 1,024 nodes with linear rectifier activation (ReLUs), and an output layer with softmax activation representing the 2,400 context-dependent states from the baseline GMM model. The baseline DNN system is boosted by transfer learning from a DNN with the same topology for English [18]. The baseline DNNs are bootstrapped with CE training using alignments from a standard HMM-based system with discriminatively trained Gaussian mixture models (GMMs).

Sequence training is initialized by copying the ReLU weights of this DNN baseline. The lattices are generated during training using the current DNN estimate and a unigram language model based on the training vocabulary with 29,000 words. On-the-fly decoding/alignment improves the overall stability of training, as it helps to avoid issues with convergence, heuristics such as when to update the lattices and alignments [12, 13] or special treatment for silence [12]. The training criterion is MMI, as described in Section 2. Most frames are correctly classified and the numerator and denominator cancel (85% for this task). These frames are not further processed in DistBelief, which saves a considerable amount of computing time. To compensate for this, the batch size is reduced from 200 (CE) to 32 (MMI).

4.2. Experimental Results

Figures 4, 5, 6, and 7 show the training criterion F_{MMI} (Eq. 2) and the word error rate (WER) for decoding with a unigram language model, both computed on a held out dataset of roughly 30 minutes from the training data, not used for training. In contrast, test WERs in Table 1 are for regular decoding on the test dataset using a trigram language model.

Data shuffling. First, different shuffling schemes are evaluated. To make the benchmark harder, the network is trained from scratch. Note that we use a stronger baseline below for MMI training, initialized from an English network with matching topology. Test WERs for this benchmark are 15.2% (no shuffling), 14.5% (on-the-fly shuffling, see Section 3.1) vs. 14.4% (full shuffling). Unlike for the shuffling scheme for the GPGPU-based implementation in [12], we do not observe a WER difference between the two shuffling types. Also, convergence times for the two approaches are comparable. We observe similar results for other setups and languages.

SGD vs. ASGD. Fig. 4 compares the convergence of MMI for SGD (1 model replica) and ASGD (10 and 50 model replicas). The parameters are fetched at the beginning of each utterance. The learning rates (0.002 for SGD and 0.001 for ASGD) are set to the largest value that leads to stable convergence; a larger value would lead to unstable convergence and eventually result in divergence. The learning rate decays by a factor of 10 after the convergence curve flattens for a while. Convergence is sped up by using more model replicas, and is also stable for a fairly large number of model replicas.

Keeping in mind that the MMI model updates include full decoding passes, and that only 15% of the utterance frames are actually passed to DistBelief due to canceling numerator/denominator accumulation weights, the optimization efficiency looks reasonable: an MMI step takes approximately twice as long as a CE step.

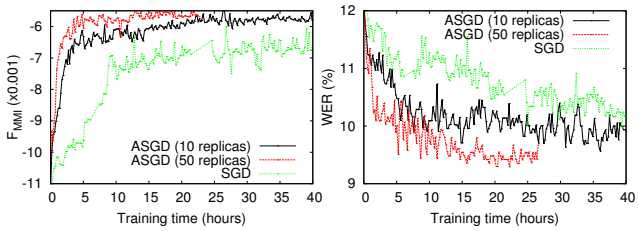

Fig. 4. SGD vs. ASGD: Evolution of training criterion F_{MMI} and WER (unigram language model) on held out data.

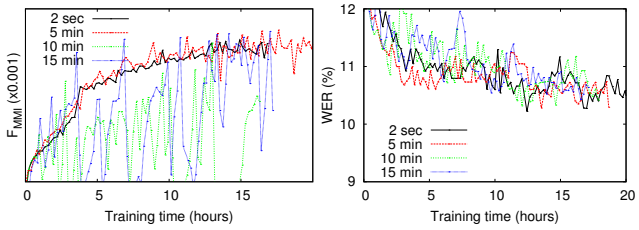

Fig. 5. Fetch interval for the decoding parameters: Evolution of training criterion F_{MMI} and WER (unigram language model) on held out data.

Fetch interval. Section 3.3 shows that ASGD for sequence training uses a tangential contact, which needs to be updated frequently. To give some measure of the stability of ASGD, we empirically determine how often the parameters for the decoders need to be fetched (referred to as the fetch interval) in order to preserve convergence. In contrast to the other experiments, the parameter fetch here is done synchronously across all replicas every fetch interval. Fig. 5 sweeps the fetch interval for updating the last layer using 10 replicas and a learning rate of 0.001. This setup allows for a fetch interval of up to 15 minutes without losing performance (convergence speed and value of objective), although at the expense of more noisy convergence. We found that several factors can have great impact on the maximum fetch interval (for which overall performance doesn't change significantly): (1) updating all layers and not only the last layer halves the maximum fetch interval, because small changes in the parameters are amplified when propagated through the hidden layers, and worse, ReLUs are non-differentiable implying that small changes can cause big changes in the outputs; and (2) not updating the silence model [12] doubles the maximum fetch interval.

Frame vs. sequence training. According to Bayes theory, CE training minimizes the frame error. In contrast, there is empirical evidence that MMI is better correlated with WER than CE [8]. We confirm this hypothesis in Fig. 6, which clearly demonstrates that CE and MMI are opposing objectives: improving the MMI criterion, starting from the CE trained DNN which is close to the optimal frame error, significantly decreases word error and significantly increases frame error.

A valid objection about the MMI gains is that the gain may not be due to an improved training criterion but rather from (frequently) re-aligning the utterances [12, 13], using numerator lattices, performing more training updates, re-starting the training, etc. To exclude these possibilities, we run the same setup as for sequence training without the denominator. Both re-aligning frequently, or only at the beginning, results in small improvements in the training criterion, as expected, but no stable and significant changes in WER.

Updating the last layer vs. all layers. DNNs can be thought of as a feature extraction (the hidden layers) and a classifier (the last softmax layer). This raises the question whether it suffices to only update the classifier to tweak word error, or if all layers need to be updated to take full advantage of sequence training. The results in Fig. 7 suggest that for this specific setup, updating all layers tends to be slightly better than updating only the last layer. However, this small difference does not transfer to the test word error (10.9% WER in either case).

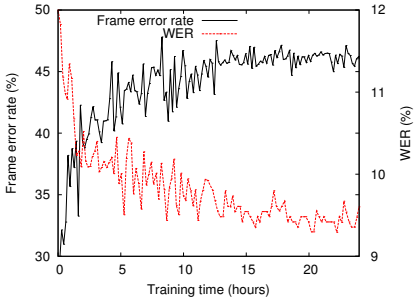

Fig. 6. Reducing F_{MMI}: Evolution of frame error (context-dependent states) and word error rate (unigram language model) on held out data.

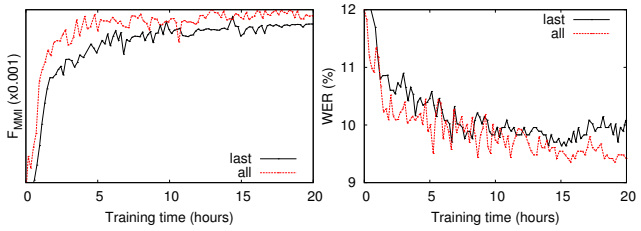

Fig. 7. Updating the last layer vs. updating all layers: Evolution of training criterion F_{MMI} and WER (unigram language model) on held out data.

Large scale task. We confirm the results on the large scale English Voice Search task: a DNN with eight hidden layers and 14,000 outputs, trained on 3,000 hours of data. The results are shown in Table 1. Most of the gain is obtained within a few days. More remarkable, running the training for much longer (say, two more weeks) does not show any overfitting but rather keeps improving marginally.

5. SUMMARY & CONCLUSIONS

We investigated asynchronous stochastic gradient descent for sequence training of deep neural networks. For the two Voice Search tasks, asynchronous sequence training as proposed works, and gives up to 15% reduction in word error rate over a frame-level trained deep neural network baseline. In particular, asynchronous stochastic optimization leads to stable and, possibly due to improved data shuffling, better convergence, and is effective and efficient at optimizing the sequence training criterion. Moreover, we empirically found that the optimization shows some robustness against stale model parameters. Although MMI is better correlated with the test word error rate than the CE frame-level criterion, small differences on the held out dataset for the training criterion using the weak language model often do not carry over to the test word error rate using the trigram language model. This shows that MMI is imperfect in terms of correlation with test word error. Future work will focus on expanding to other training criteria such as MPE and sMBR.

Table 1. Summary of test WERs for Voice Search tasks.

(training data)	Icelandic (60h)	English (3,000h)
CE	13.0	11.7
MMI	10.9	10.7

6. REFERENCES

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

[2] G.E. Dahl, D. Yu, and L. Deng, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

[3] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing, Special Issue on Deep Learning for Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

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

[5] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[6] H. Bourlard and N. Morgan, *Connectionist speech recognition*, Kluwer Academic Publishers, 1994.

[7] H. Hermansky, D.P.W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Istanbul, Turkey, June 2000.

[8] Y. Normandin, *Hidden Markov Models, Maximum Mutual Information, and the Speech Recognition Problem*, Ph.D. thesis, McGill University, Montreal, Canada, 1991.

[9] D. Povey, *Discriminative Training for Large Vocabulary Speech Recognition*, Ph.D. thesis, Cambridge, England, 2004.

[10] B. Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, Apr. 2009, pp. 3761–3764.

[11] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," in *INTERSPEECH*, 2012.

[12] H. Su, G. Li, D. Yu, and F. 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, pp. 6664–6668.

[13] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *INTERSPEECH*, 2013.

[14] L. Bottou, "Stochastic gradient learning in neural networks," in *Neuro-Nîmes*, 1991.

[15] Y. Kubo, T. Hori, and A. Nakamura, "Large vocabulary continuous speech recognition based on WFST structured classifiers and deep bottleneck features," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[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 *International Conference on Machine Learning*, 2012, pp. 81–88.

[17] 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 *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[18] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, Apr. 2013, vol. 1.

[19] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[20] G. Heigold, *A Log-Linear Discriminative Modeling Framework for Speech Recognition*, Ph.D. thesis, RWTH Aachen University, Aachen, Germany, June 2010.

[21] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, ""Your word is my command": Google search by voice: A case study," in *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*, chapter 4, pp. 61–90. Springer, 2010.