Asynchronous, Online, GMM-free Training of a Context Dependent Acoustic Model for Speech Recognition

Michiel Bacchiani, Andrew Senior, Georg Heigold

Google Inc.

{michiel,andrewsenior,heigold}@google.com

Abstract

We propose an algorithm that allows online training of a context dependent DNN model. It designs a state inventory based on DNN features and jointly optimizes the DNN parameters and alignment of the training data. The process allows flat starting a model from scratch and avoids any dependency on a GMM/HMM model to bootstrap the training process. A 15k state model trained with the proposed algorithm reduced the error rate on a mobile speech task with 24% compared to a system bootstrapped from a CI HMM/GMM and with 16% compared to a system bootstrapped from a CD HMM/GMM system.

Index Terms: Deep Neural Networks, online training

1. Introduction

The previously predominant approach to acoustic modeling for speech recognition was based on the use of Hidden Markov Models (HMMs) which modeled state emission probabilities using Gaussian Mixture Models (GMMs). Nowadays, Deep Neural Networks (DNNs) have become common place in the acoustic model [1]. The application of such DNNs can be grouped into two. In the bottleneck approach [2, 3, 4], the neural network is used to extract features from the audio signal. Those features then serve as the input to a HMM/GMM system. The other popular application type is the hybrid model where the DNN provides probability estimates that substitute the state emission probabilities, previously modeled by GMMs [5, 6, 7, 8].

For bottleneck approaches, the HMM/GMM remains an integral part of the system as recognition relies on it. The DNN only functions as a feature extractor. The benefit of this approach is that the algorithms such as discriminative training and speaker adaptive techniques developed specifically for HMM/GMM systems can be applied directly. In contrast, the hybrid systems need new formulations for such algorithms. Sequence training to implement discriminative training for DNNs recently received a lot of attention [9, 10, 11, 12]. Investigations into speaker adaptation for DNNs have also been studied recently [].

Although in contrast to the bottlneck approach, the hybrid system does not rely on the HMM/GMM as a core component, the commonly used systems retain a large dependency on the HMM/GMM. First, the DNN is trained to provide state probability estimates for a state inventory that is derived from an HMM/GMM system. If the DNN is trained based on cross entropy (CE), an additional HMM/GMM dependency is through the initial alignment associating input features with state labels. For sequence trained systems, the alignment is not a requirement, but the dependency on the state inventory definition remains and sequence training itself is generally bootstrapped from an initial DNN that is commonly obtained from CE training.

In recent work, we investigated the possibility of training a Context Independent (CI) DNN from scratch using a flat start procedure. In that online approach, the DNN is initialized randomly. Forced alignment of the input using the DNN then defines associations of input data with CI labels providing the DNN with training data. As DNN parameter estimates evolve with training, the forced alignments based on the DNN parameters change, which in turn provides altered training data for the DNN training. In the work described in [13] we show that such a procedure is viable for a CI system and leads to convergence. However, this study did not extend to doing this type of online training of a CD system due to issue related to the stability of learning such large state inventory model.

In related recent work, we investigated the feasibility of designing the Context Dependent (CD) state inventory based on the activations obtained from a CI DNN. The advantage of this approach is that the state inventory is matched to the DNN model as opposed to the state inventory from the GMM which is mismatched in terms of the features and model family used to design it. However, this system still relied on a CI HMM/GMM system to bootstrap the training procedure. Furthermore, the poorly matched alignment has a negative impact on the resulting system accuracy.

In the work here, we describe a system that completely avoids the use of a GMM/HMM system. It uses the flat start procedure to train an initial CI system. It uses the activation clustering approach to define a CD state inventory based on the DNN activations. And it jointly optimizes the segmentation and DNN network through online training. Furthermore, we implement this approach in our distributed asynchronous DNN training infrastructure to obtain an algorithm that allows scaling to large data sets. Both asynchrony and a large tied-state inventory make online training challening and a focus of this paper is to provide an algorithm that provides stable learning of such a system.

In section 2 we describe the online training setup of the system. In section 3 we describe experimental results and discuss various choices for the parameter updates. Finally, in section 4 we discuss effectiveness of the prosed algorithm.

2. Model Training

In this section, we describe the training system used to optimize the DNN and the tied-state inventory. A high level outline of the training procedure is that we first train a CI system from a random initialization by online training. That CI system is then used to define a CD tied-state inventory and online training of the CD system completes the system optimization. First, in section 2.1 we describe the DNN topology. Then in section 2.2

we describe the distributed model trainer. In section 2.3 we describe how context dependent modeling is included in the system.

2.1. DNN Model Topology

In all cases we use a DNN that consists of an input layer, a number of hidden layers and a softmax output layer. We fix the dimension of the hidden layers to be 2560 nodes using a ReLU non-linearity [8]. We also fix the input layer to take 26 frames (20 frames preceding the current input frame and 5 consecutive frames) of 40-dimensional log-filterbank parameters. The softmax layer will have one output per state and hence its dimension is defined by the clustering procedure that defines the tied-state inventory.

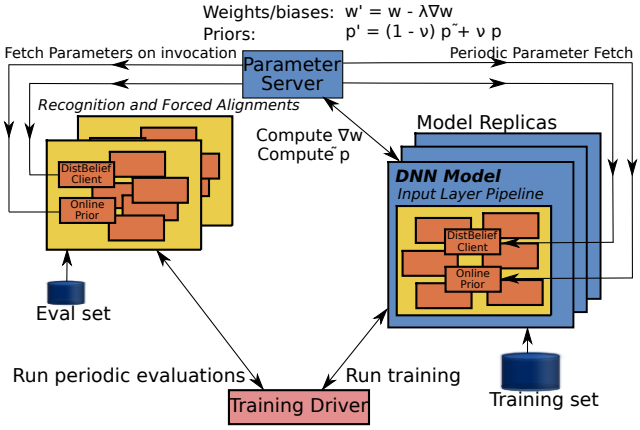

Figure 1: *Online DNN training system overview.*

2.2. Asynchronous Parallel DNN Trainer

The DNN parameters are trained with the online training system depicted in figure 1. This system is based on our *DistBelief* distributed parallelized neural network optimization system [14]. The training process is driven by the training driver. This driver controls a number of model replicas (in this study, we consistently use 100). Each such model replica retains a complete DNN model and has a special input layer which has the ability to read an utterance from the training data and force align it using the Input Layer Pipeline. This pipeline retains model components like the lexicon and context dependency transducer. The state likelihood computations that it requires for the forced alignment process are obtained from a *DistBelief client*. This client is itself a complete DNN model allowing inference computation. Given an input pattern, the client can compute DNN layer outputs. The DNN parameters used for that inference computation are periodically updated from a separate *parameter server* process. Another parametric model that is used in the forced alignment computation is a state prior which will be discussed in more detail in section 2.2.2. This prior distribution parameter is similarly obtained from an *online prior* which periodically fetches its values from the parameter server process.

Once the input layer in a model replica completes a forced alignment, the result associates input patterns (in the form of 26 frames of 40-dimensional log-filterbank parameters) with state labels. These pairs are passed to the DNN training process. Training uses Stochastic Gradient Descent (SGD) to update the DNN parameters. For each mini-batch (in this study 200 samples), the model replica computes the inference of the DNN given the current model parameters w, then back propagates the cross entropy loss observed at the output softmax layer providing the gradients ∇w of error with respect to the parameters. Right before starting this computation, the model replica will query the parameter server to get the most recent parameter estimates w and on completion of the computation, the model replica will send the computed gradients ∇w back to the parameter server.

2.2.1. Gradient Based DNN Training

The parameter server retains the latest DNN parameter estimates w and updates those parameters to w' based on the gradient information ∇w it obtains from the model replicas by gradient descent learning as $w' = w - \lambda \nabla w$. The learning is parameterized through the learning rate λ. In our study we used a fixed learning rate of 0.003. Note that due to the distributed nature, the learning process is inherently asynchronous. In the time it takes a model replica to compute a batch update, other model replicas will update the parameters. Hence the gradient computation, based on the parameters fetched right before the gradient computation are stale by the time the replica sends its gradient information back to the parameter server.

2.2.2. Online State Prior Estimation

The forced alignment pipelines in the input layers of the model replicas require an a priori estimate of the likelihood $p(x \mid s)$ of observing acoustic observation x given state s. However, the estimates of the DNN provide $p(s \mid x)$ instead. Hence, to arrive at the desired probability estimate, the alignment pipeline will in addition need an estimate of the state prior $p(s)$ so that the alignment process can use $p(s \mid x)p(s)^{-1}$ in its computation. To learn this prior, we use an interpolation model similar to what was used in out previous work in [13]. The required state prior probability estimate is learned similar to the DNN parameters themselves in the sense that the estimate is retained in the parameter server and that model replicas provide updates to the estimate by processing training data in parallel. But in contrast to the DNN SGD type learning, the prior is updated by linear interpolation. More concisely, if a model replica has estimated a new state prior estimate $\tilde{p}(s)$ and sends that prior distribution estimate to the parameter server, the new prior estimate $p'(s)$ is updated from its previous estimate $p(s)$ as $p'(s) = (1-\nu)p(s)+\nu\tilde{p}(s)$ with ν a parameterization of the prior learning. Another contrast of this prior learning process in comparison to the DNN parameter updates is that the prior interpolation is not performed for each mini-batch but replicas provide an update after a certain number of frames have been counted. That update interval is a second parameterization of the prior learning.

2.2.3. Asynchronous Online Learning

As discussed in the description of the trainer, the parallel nature of the optimization system make asynchrony inherent. But a number of training parameters control the level of asynchrony. Specifically, the alignment pipeline in the input layer will periodically fetch model parameters and prior parameters from the parameter server. The periodicity of this update defines in part the level of asynchrony of the training procedure. Staleness of parameters can lead to optimization with a poor auxillary function which is discussed in more detail in light of sequence training in our recent work in [12].

For the prior, the asynchrony issue needs to balance the update interval but also take into account the quality of the new prior estimate obtained from counting. If few examples have been seen due to a rapid update interval, the prior estimate $\tilde{p}(s)$ will be poor. Setting a longer period in between prior updates will lead to a more accurate local estimate, but the prior used in the alignment pipeline to compute training examples might be very stale leading to incorrect learning.

For both the DNN parameter learning and the prior update, the period between updates and the impact of those updates expressed by the learning rate λ and interpolation rate ν need to be balanced. A poor balance might lead to unstable learning as bad or stale parameter estimates cause the samples that guide training (as computed from the forced alignment in the input layer) to make the model estimates diverge. In our previous work in [13], we empirically found that a prior interpolation factor that performed well. In the work here, we investigate the options a bit more, varying the update interval and interpolation weight and observe the resulting learning behavior.

2.2.4. Training Metrics

Besides supervising the training process implemented by the model replicas and parameter server, the training driver also runs periodic evaluation pipelines. These evaluations are executed on an evaluation set, not part of the training set. Two such pipelines are used and both use the DNN and prior state distribution that is being trained. At the start of a new evaluation run, the pipelines will fetch the DNN and prior parameters from the parameter server and then keep the estimates fixed through the evaluation iteration.

One evaluation pipeline runs speech recognition and measures word error rates. The other runs forced alignment on the evaluation data and computes frame-based statistics. In particular, we track frame accuracy by measuring the agreement between the forced alignment frame state alignments and the classification result of the DNN. If under the forced alignment constraint a frame is aligned with a state s and if the DNN produces a likelihood for that state that exceeds the likelihood for any other state, the frame is counted as correct. Another key metric we observe from forced alignment is referred to as the *error cost*, the likelihood gap between the best scoring state and the correct state (where correctness is expressed based on the state labeling found by forced alignment).

2.3. Context Clustering

For the construction of a CD system, the work here uses the CI model obtained from the online training procedure described above. Using the algorithm detailed in [15] we construct a tied-state inventory from the activations of the last hidden layer of the DNN (the input to the softmax layer). In contrast to that previous work the DNN is used for alignment, not a CI HMM/GMM. Like our previous work, we limit the scope to modeling triphones and we implement the learned context dependency model through the transducer construction detailed in our previous work [15]. However, in contrast to our previous work, we initialize the newly defined context dependent model differently.

The transition from a CI to a CD system gives rise to a new softmax layer. We first train this layer from a random initialization keeping the hidden layer parameters fixed to the values that were obtained from CI training. Training the new softmax layer, we use the exact same forced alignments that were used to define the context dependent state inventory. In other words, we use CE training for the the softmax layer parameters alone. In a second stage, we still use CE training but train the complete model (softmax as well as hidden layer parameters). The DNN parameters obtained from that initialization process serves as the starting point for online training (as described above) of the CD system.

To initialize the prior distribution for the newly defined tied-state inventory, we use the tied state counts found in tree clustering of the context dependent state inventory. Let the total number of frames associated with CI state s be denoted as N_s and let the prior probability of that state be denoted as $p(s)$. Then consider tied-state q that represents a set of context dependent versions of state s and let N_q denote the frame count for tied-state q. We then initialized the prior probability of the tied state as $p(q) = \frac{N_q}{N_s} p(s)$. In other words, we partition the prior probability mass of the CI states among the tied CD states in proportion to the frame counts assigned to those tied states.

3. Experimental Results

We evaluate the effectiveness of our training procedure on a database of mobile speech recordings originating from a number of Android applications: voice search, translation and the voice-based input method. These recordings are anonymized; we only retain the speech recording but remove all information indicating which device recorded the data. The training set consists of a sampling of about 3 million utterances containing about 2000 hours of speech. We obtained manual transcriptions for these utterances. The evaluation pipelines process a test set that contains data sampled uniformly from all applications emphasizing the use frequency of each application. This test set contains about 23000 utterances or about 20 hours of speech. This test set and the evaluation system is identical to the one described in our previous work [15] and hence error rate numbers are comparable.

3.1. Flat Start Initialization

To build a 128 state CI model, we initialized a DNN with random parameters and initialized with a uniform prior distribution. Figure 2 shows the training evolution of a network with 7 hidden layers of 2560 nodes each. It shows the cross entropy loss, the word error rate from the recognition evaluation pipeline, the frame classification rate and error cost from the alignment evaluation pipeline over 10 million steps of SGD training using 200 frame mini batches. The word error rate steadily decreases with training down to 32.9%. This error rate is 3.4% lower than the 36.3% reported in our previous work for a DNN with the same topology trained on the alignments of a CI HMM/GMM [15].

In additional experiments with flat starting CI systems, where we varied the number of hidden layers from a one to eight hidden layers, we observed the same convergence behavior as shown in figure 2.

3.2. Context Dependent Models

In some initial experiments, we started with a CI system with three hidden layers of 2560 nodes, trained them using the flat start procedure and then constructed CD tied state inventories of 400, 800 and 2000 states. We then ran online training, *randomly* initializing these networks. We ran three online training experiments of this sort where we varied the update interval of the DNN parameters in the input alignment layer inference client. We set the fetch interval to 50, 2500 and 5000 mini-batches re-

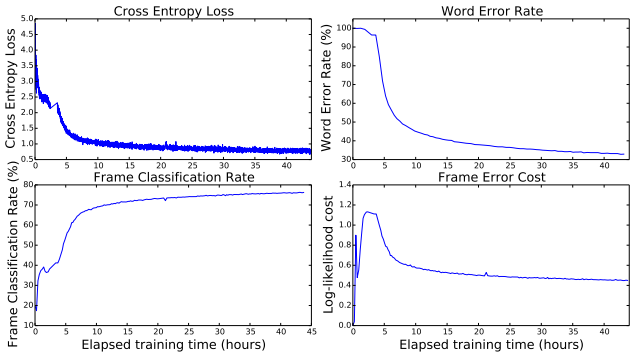

Figure 2: *Training metrics during the flat start procedure running 10M steps of 200 frame mini-batch SGD training.*

State Inventory	Batch Update Interval		
	50	2500	5000
CI	35.9	35.5	35.3
400	24.4	24.4	24.3
800	21.5	22.0	21.5
2000	16.7	16.9	17.2

Table 1: *WER performance of systems with three hidden layers, obtained from 10M step training using various state inventory sizes and parameter fetch update intervals.*

spectively. In each of these experiments, we updated the prior with an interpolation weight ν of 0.9 for every 10k frames seen. We observed that all training runs converged and obtained the error rate results after 10M steps of training as shown in table 1.

Starting online training from a random initialization for state inventories of 3000 states or larger, we observed divergence of the training procedure. In those cases, we observed the error cost go up 100 fold compared the the CI or small CD state inventory runs. This indicates that some states have a likelihood far exceeding any other state. Closer inspection revealed that this was caused by some states getting vanishingly small prior state probabilities. The division of the a priori state probability by such small state prior probabilities leads to instability of the estimates, which in turn throws off the alignment process in the input layer. Even when using the careful initialization algorithm using CE training of the newly formed softmax layer from context clustering, we observed divergence. However, with a careful choice of a prior update regimen we were able to get stable learning. The parameters that define the prior update are the update interval and the interpolation weight ν. Figure 3 shows training of a 10k state system setting the prior update interval to 100k, 300k or 3M frames respectively, but keeping the interpolation weight constant at 0.9 It also plots the performance of a 15k state model using a 10k frame update interval, but using a 0.999 interpolation weight. The large interval updates cause instability in early updates as the prior is poorly matched, but training does not diverge. As the update interval gets shorter, the prior estimate get poorer and learning is more chaotic. The 15k state system using frequent updates but with a large interpolation weight is better behaved. Note that training with an interpolation weight of 0.99 as opposed to 0.999 lead to divergence. The 15k system training converged to a 12.2% WER. The equivalent system described in [15] trained from a CI HMM/GMM alignment achieved a 16.0% WER, one that was bootstrapped from a CD HMM/GMM achieved 14.5%. The online trained system here provide a relative error rate reduction of 24% or 15% respectively.

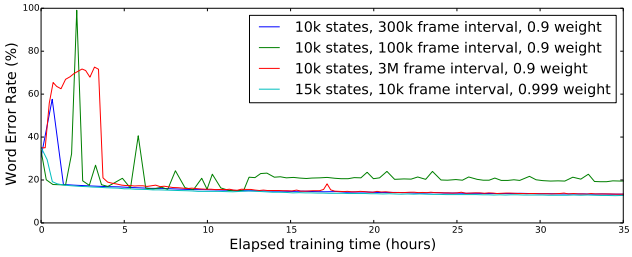

Figure 3: *WER performance of 10k and 15k systems trained with different prior parameter update regimen.*

4. Conclusions

This works shows the feasibility of online training of a CD DNN with a large state inventory within an asynchronous parallelized optimization framework. First the proposed algorithm provides a mechanism to flat start a CI system from a random initialization. We show that the proposed algorithm has stable convergence. Experimental results showed resilience to network depth (networks with one up to eight hidden layers showed stable convergence) as well as to a large range of DNN parameter update regimens (fetching parameters every 50 to 5000 batch computations made little impact on learning). The jointly optimized alignment and DNN parameters lead to a more accurate system than one trained from a CI HMM/GMM system, as evident from the 32.9% WER of the CI system trained here vs. 36.3% obtained in [15]. That performance gain persists when comparing the CD systems of 15k states. The joint optimization described here achieves a system reaching 12.2% WER comparing favorably with the 16.0% system (CE training from CI HMM/GMM alignments) or 14.5% system (CE training from CD HMM/GMM alignments) obtained in previous work.

The extension of the algorithm to a CD system with 15k state shows that the online training of a large state inventory system has stable convergence as long as the prior update regimen is well controlled. This is akin to choosing an appropriate learning date for the DNN SGD training. Experiments showed stable learning with a interpolation weight close to 1.0. Another option, setting the prior update interval large enough to allow a good prior estimate before interpolation also seems to lead to stable learning. Although in such schemes, early parameter updates seem less well behaved than the alternative of updating the prior frequently but with a large interpolation weight.

Not only is the joint optimization of the DNN parameters and alignment beneficial to the performance of the final system it provides the additional benefit that the system training described here has no dependency on an HMM/GMM at all. The optimization, matched to the model, leads to accuracy gains and the complexity of the implementation is limited as an HMM/GMM implementation no longer needs to be maintained.

The work here successfully extends our previous work on GMM-free DNN optimization reported in [13]. It shows that the proposed algorithm is feasible within asynchronous parallel optimization making it more scalable and in addition shows it allows online training of a large CD state inventory system.

5. References

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A.Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] T. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-Encoder Bottleneck Features Using Deep Belief Networks," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2012.

[3] Z. Yan, Q. Huo and J. Xu, "A Scalable Approach to Using DNN-Derived Features in GMM-HMM Based Acoustic," in *Proc. of Interspeech*, 2013, pp. 104–108.

[4] P. Bell, H. Yamamoto, P. Swietojanski, Y. Wu, F. McInnes and C. Hori, "A lecture transcription system combining neural network acoustic and language models," in *Proc. of Interspeech*, 2013, pp. 3087–3091.

[5] T. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed, "Making Deep Belief Networks Effective for Large Vocabulary Continuous Speech Recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

[6] F. Seide, G. Li, and D. Yu, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks," in *Proc. of Interspeech*, 2011.

[7] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application Of Pretrained Deep Neural Networks To Large Vocabulary Speech Recognition," in *Proc. of Interspeech*, 2011.

[8] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton, "On Rectified Linear Units for Speech Processing," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2013.

[9] 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 *Proc. of Interspeech*, 2012.

[10] K. Vesely, A. Ghoshal, L.Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. of Interspeech*, 2013.

[11] H. Su, D. Yu and F. Seide, "Error Back Propagation for Sequence Training for Context-Dependent Deep Networks for Conversational Speech Transcrtiption," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2013, pp. 6664–6668.

[12] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior and M. Bacchiani, "Asynchronous Stochastic Optimization for Sequence Training of Deep Neural Networks," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2014.

[13] A. Senior, G. Heigold, M. Bacchiani and H. Liao, "GMM-free DNN training," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2014.

[14] 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. Neural Information Processing Systems*, 2012.

[15] M. Bacchiani and D. Rybach, "Context Dependent State Tying for Speech Recognition using Deep Neural Network Acoustic Models," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2014.