STOCHASTIC DATA SWEEPING FOR FAST DNN TRAINING

Wei Deng *Yanmin Qian* *Yuchen Fan* *Tianfan Fu* *Kai Yu*

Institute of Intelligent Human-Machine Interaction
MOE-Microsoft Key Lab. for Intelligent Computing and Intelligent Systems
Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
{cosmic_phantom, yanminqian, fyc0624, erduo, kai.yu}@sjtu.edu.cn

ABSTRACT

Context-dependent deep neural network (CD-DNN) has been successfully used in large vocabulary continuous speech recognition (LVCSR). However the immense computational cost of the mini-batch based back-propagation (BP) training has become a major block to utilize massive speech data for DNN training. Previous works on BP training acceleration mainly focus on parallelization with multiple GPUs. In this paper, a novel *stochastic data sweeping* (SDS) framework is proposed from a different perspective to speed up DNN training with a single GPU. Part of the training data is randomly selected from the whole set and the quantity is gradually reduced at each training epoch. SDS utilizes less data in the entire process and consequently save tremendous training time. Since SDS works at data level, it is complementary to parallel training strategies and can be integrated to form a much faster training framework. Experiments showed that, combining SDS with asynchronous stochastic gradient descent (ASGD) can achieve almost 3.0 times speed-up on 2 GPUs at no loss of recognition accuracy.

Index Terms— Deep neural network, Speech recognition, Stochastic Data Sweeping, Asynchronous SGD, GPU

1. INTRODUCTION

Context-dependent deep-neural-network HMM (CD-DNN-HMM) has achieved dramatic performance improvements for large-vocabulary continuous speech recognition (LVCSR) [1, 2]. Here, the traditional Gaussian mixture model (GMM) is replaced by *deep neural network* (DNN) to model the state emission probabilities of HMMs. Training of DNN consists of two stages: initialization and fine tuning. Various layer-by-layer initialization approaches, such as unsupervised deep-belief-network (DBN)[3], can be used. After that, the back-propagation (BP) [4] based fine-tuning is applied. It has been observed that immense computational cost is required for DNN training, which becomes a major bottleneck of employing DNN for massive speech data. Even if GPU is used, the training process is still very expensive and the demands on further acceleration is high. In particular, since BP training is normally required every time new data is incorporated, accelerating BP attracts much interest of researches from both academia and industry.

BP training of DNN employs the *stochastic gradient descent* (SGD) approach, which involves a full model update after error propagation with a mini-batch of data (e.g. only a few hundred frames). One widely used framework to accelerate BP training is *multiple-GPU parallelization*. Pipelined back-propagation combining with model striping in the output layer [5] is proposed to reduce the training time by allocating different layers computation on different GPUs in parallel. Cluster-based DNN splits the model into independent sets for parallel training and then merges the result using an additional NN [6]. Asynchronous stochastic gradient descent (ASGD) uses multiple GPUs to compute gradients on different data using the latest model independently, and updates the model in the host server asynchronously[7]. All these approaches use the same architecture of one server with multiple GPU cards and save tremendous training time without loss of recognition accuracy.

Although *multiple-GPU parallelization* approaches can reduce training time dramatically, they require additional and expensive hardwares and hence are not feasible for researchers who have limited resources. In this work, a new framework of DNN training acceleration without facility enlargement is proposed. The framework relies on data sampling rather than parallelization. It randomly samples a subset of the whole data set for training at each epoch, where the sampling data amount is controlled by a data sweeping function. It is referred to as *stochastic data sweeping* (SDS). Due to the reduction in training data amount, the computation cost is saved naturally. Moreover this framework can be easily combined with multiple-GPU parallelization. The advantages of both frameworks are additive and much faster DNN training architecture can be constructed. In this paper, SDS with a single GPU will first be investigated and then com-

This work were supported by the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning and the China NSFC project No. 61222208.

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

bined with a multi-GPU parallelization approach, the ASGD algorithm.

The rest of this paper is organized as follows. Section 2 briefly reviews DNN-HMMs and the SGD algorithm. In section 3, stochastic data sweeping is described in detail. Experiments as well as analysis are given in section 4. Section 5 concludes the paper and discusses future research directions.

2. DNN TRAINING WITH SGD

In the hybrid DNN-HMM framework, acoustic events are modelled by a DNN, whose outputs are softmax posterior probabilities $P(s|\mathbf{o})$ with respect to each HMM state s. The DNN training process optimizes the cross entropy function

$$\mathcal{L}(\theta) = -\sum_{s} d_s \log P(s|\mathbf{o}, \theta) \tag{1}$$

where θ is the set of parameters of DNN, d_s is 1 for the target state and 0 for non-target states. During evaluation, $\log P(s|\mathbf{o})$ is calculated and used together with $\log P(s)$ to form the acoustic score $\log p(\mathbf{o}|s)$.

Once a DNN is initialized, it is fine tuned using the well known gradient descent based error back-propagation (BP) algorithm [4]. *Stochastic gradient descent* (SGD) has proved to be useful in practice for BP training on large datasets. The model parameter θ is optimized using

$$\theta_{t+1} = \theta_t - \gamma \nabla \mathcal{L}(\theta_t) \tag{2}$$

where $\mathcal{L}(\theta)$ is the error function as defined in equation (1) and γ is the learning rate, t denotes the index of a small subset of data, often referred to as *mini batch*, selected randomly from the whole training set. From equation (2), full model parameter θ is updated for each mini-batch. This results in vast amounts of full DNN parameter updates when sweeping the whole training data set. The small data size (normal several hundred frames in speech recognition) in one mini-batch and the frequent full model updates in SGD are the main limitations in BP training. Even with GPU, for a LVCSR task with hundreds or thousands of hours speech data, it can take several weeks or even months to train a DNN model.

To speed up DNN training, great efforts have been put on parallelising SGD using multiple-GPUs [5, 6, 7]. They all aim at parallelised training with the whole training data set using enhanced hardware facilities. For example, asynchronous SGD (ASGD) users a server-client model, where multiple client GPUs are used to independently update DNN models using data within a mini-batch and the updated DNN are uploaded to server for synchronisation. Although it significantly improves training speed, it also requires more hardwares such as a server and multiple GPUs.

3. STOCHASTIC DATA SWEEPING

In contrast to SGD parallelization on the whole training data set, in this section, an alternative novel framework, *stochastic data sweeping* is proposed.

The basic idea is to only sweep a randomly selected subset of the training data in each BP training epoch. Although reducing the amount of training data at each epoch will obviously save the total training time, it may also significantly degrade the recognition performance. As shown in the experiments, using fixed amount of random selected subsets throughout the entire BP training process will not yield good trade-off between saving training time and keeping recognition accuracy. Therefore, dynamic subset selection using *data sweeping function* is investigated in this paper.

3.1. Data usage rate of stochastic data sweeping

To evaluate the total amount of training time saving with respect to the standard whole-data training process, *data usage rate* (DUR) is defined as the metric:

$$r_i = \frac{1}{K} \sum_{k=0}^{K} s_i(k) \tag{3}$$

where i is the type index, k is the epoch index, $s_i(k)$ denotes the data sweeping function of type i at the k^{th} training epoch and K is the total number of epochs for BP training. From the definitions of the data sweeping functions, DUR is always between 0 and 1. It reflects the overall computational cost saving after the entire BP training process.

3.2. Data sweeping functions

Dynamic subset selection is to randomly sample different amount of data from the whole data set at each training epoch. General idea is to use larger subsets for early epochs and smaller subsets for late epochs. Since data are sampled randomly at each iteration and large portions are used in early epochs, the hope is that all training data will be effectively swept and contribute to the training process. The exact amount of the random data selection at each epoch is determined using a *data sweeping function*.

In this paper, three data sweeping functions, represented by the percentage of the entire training set selected at the n^{th} epoch, are investigated. These functions are shown in figure 1 and mathematically defined as below:

$$s_0(n) = 1 \qquad n \in [0, K] \tag{4}$$

$$s_1(n) = \alpha \qquad n \in [0, K], \;\; \alpha \in (0, 1] \tag{5}$$

$$s_2(n) = \begin{cases} 1 - \beta n & n \in [0, l], \;\; \beta \in (\frac{1}{K}, \frac{1}{l}] \\ c & n \in [l+1, K] \end{cases} \tag{6}$$

$$s_3(n) = \begin{cases} \cos(\lambda n) & n \in [0, l], \;\; \lambda \in (\frac{\pi}{2K}, \frac{\pi}{2l}] \\ c & n \in [l+1, K] \end{cases} \tag{7}$$

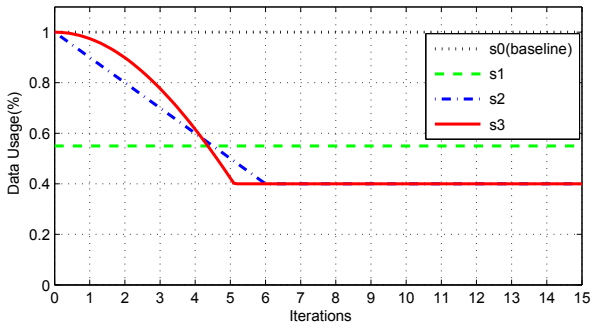


Fig. 1: Different data sweeping functions

where $s_i(n)$ denotes the data sweeping function at epoch n of type i, K is the total number of epochs, α, β, λ are deweighting factors, c is the smallest percentage of the training data that is used in SDS. Hence the parameter α, β, λ, and c, l control the sampled data quantity which consequently decide the overall training time. It is also possible to set the DUR first and then given c and l, determine the parameters α, β, λ. This back-calculation is useful for performing experiments and is used in this paper.

From the definition and figure 1, s_0 indicates the normal training method that uses the whole data set, and s_1 randomly selects samples at a fixed percentage throughout the entire BP training process. In contrast, s_2 and s_3 make the percentage variable according to individual epoch, and at early epochs the sampled data quantity is relative large and the amount gradually decreases in later epochs. It has been known that at early BP training stages, the learning rate is usually large and the model parameters change greatly. So the early epochs are relatively more crucial and more data is helpful for guiding DNN training towards the correct direction. While in the later epochs, the model is more stable and learns slowly, so more training data may be able to be reduced. s_2 and s_3 are designed to reflect this motivation.

4. EXPERIMENTS

Effect of data usage rate and learning rate during BP were first investigated under the stochastic data sweeping (SDS) framework on a 50-hour switchboard English dataset for fast experiments. SDS was then applied to two large LVCSR tasks, a 309-hour switchboard English task and a 250-hour Mandarin ASR task. Combination of SDS with asynchronous SGD (ASGD) was also tested on the two LVCSR tasks.

4.1. Effect of data usage rate and learning rate

In this experiment, 810 speakers out of 4869 speakers from the 309 hours data of the switchboard data[8] set were first randomly selected, which forms a small training set of about 50 hours for fast experimental investigation. PLP features normalised by per-speaker mean and variance, along with 1^{st} and 2^{nd} derivatives were used as the raw features. Cross-word triphone models with 3001 tied-states was used. State alignment for DNN training was generated using a GMM model. A trigram language model which was trained on the transcription of the 2000h Fisher corpus and interpolated with a background trigram model was used for decoding. The switchboard (`swb`) part of the Hub5'00 data set and the fisher (`csh`) part of the RT03S data set were used as the test set in this paper, which is the same as [9]. All DNNs have 7 hidden layers with 2048 nodes per layer. The RBM-pretrained DNN was fine tuned with cross-entropy objective function, along with a L2-norm weight-dacay term of coefficient 10^{-6}.

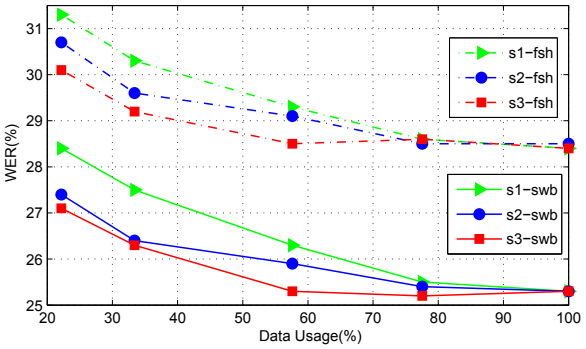


Fig. 2: The relationship between performance and data usage rate of 3 data sweeping functions

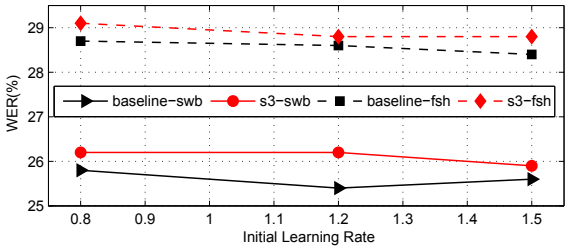
As indicated in the previous section, it is possible to tune the parameters of the data sweeping function to achieve certain level of overall data usage rate (DUR). The word error rates (WER) of the three data sweeping functions with different DUR are shown in figure 2. From the figure, keeping fixed data sweeping rate (i.e. s_1) yielded notable performance degradation when DUR is less than 77%, which is not effective. In contrast, using non-linear data sweeping functions in SDS resulted in better trade-off between WER and DUR. $s3$ with DUR of 55% showed the best trade-off and will used for later experiments.

In the experiments, a learning rate annealing and early stopping strategies as in [10] was used[1]. The learning rate γ is dynamically tuned for different epochs during BP training using the below formula

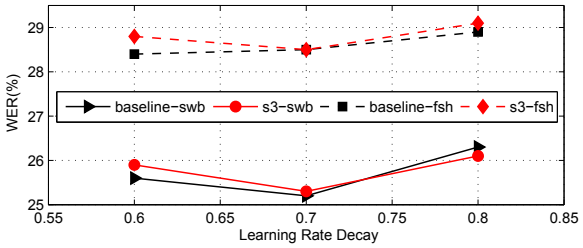
$$\gamma(n) = \begin{cases} T & n \in [0, l] \\ \tau^{n-l} T & n \in [l+1, K] \end{cases} \quad (8)$$

where T is the initial learning rate, K is the total number of epochs, l is the epoch index when the CV accuracy increase is smaller than a pre-defined threshold, τ is the decaying rate. It is then interesting to investigate effect of learning rate change on stochastic data sweeping.

[1]Note that in this paper, for fair comparison, the maximum number of iteration is limited to 16.



(a) Performance of stochastic data sweeping (s_3) with different initial learning rates



(b) Performance of stochastic data sweeping (s_3) with fixed initial learning rate of 1.5 and different decaying factors.

Fig. 3: Performance of stochastic data sweeping (s_3) with different initial learning rates and decaying factors.

The performance comparison between the SDS and the baseline DNN using different initial learning rates T and decaying factors τ are shown in figure 3. It can be observed that the degradation from baseline is relatively smaller for large initial learning rate and decaying factor. This may be because SDS essentially uses different data at each epoch and the amount is smaller than the standard full training, so relatively larger learning rate and decaying factor may help to tune the model to cover more varieties. Considering the overall performance, learning rate of 1.5 and decaying factor of 0.7 are used for later experiments.

4.2. Combination with ASGD

To further demonstrate the effectiveness of stochastic data sweeping (SDS), it was applied to the full 309 hours English Switchboard data set and a 250 hours Mandarin LVCSR task. The Mandarin data were collected through mobile microphone under the real environment in a spontaneous style. Two test sets, referred to as `man1` and `man2`, were used. The `man1` has 2 hours and `man2` has 1 hour data. All the training process on Mandarin is the same as the switchboard English systems.

As indicated before, SDS is complementary to multi-GPU parallelization. Therefore, SDS was combined with asynchronous SGD (ASGD) in this experiment to obtain faster DNN BP training. The ASGD architecture was constructed as in [7]. The experiments were performed on a server with two GPU cards. Results of both the single GPU SDS training and the combination of SDS and ASGD are shown in table 1. Instead of DUR, the total BP training time are given here as more realistic and comparable results. The NVIDIA GeForce GTX 770 cards are used in the experiments here.

System	English (WER %)			Mandarin (CER %)		
	`swb`	`fsh`	`Time`	`man1`	`man2`	`Time`
Full Training	19.7	22.0	158h	15.0	7.9	108h
SDS	19.9	22.1	87h	15.1	7.8	60h
SDS + ASGD	19.7	21.9	52h	15.0	7.9	37h

Table 1: Performance of SDS and combination with ASGD on switch board English and Mandarin LVCSR tasks

It can be observed that, with single GPU, SDS only utilizes about 55% of the whole dataset in the entire training process while nearly with no loss in the recognition performance on both English and Mandarin tasks. It means that SDS can achieve about 2.0 times speed-up than the normal full training and greatly shorten the training cycle. Furthermore, combined with ASGD, the final BP training framework achieved almost 3.0 times speed-up on a 2-GPU server than the normal single-GPU baseline without any loss in accuracy. This is much faster than the pipeline BP (1.9 times speedup) [5] and the ASGD (1.6 times speedup) [7] using the similar facility.

5. CONCLUSIONS

Speeding up the BP training of DNN is of great interest to the speech community. Previous works all focused on the parallelization framework, especially multi-GPU parallelization. In this paper, an alternative novel framework is proposed to speed up BP training, referred to as *stochastic data sweeping* (SDS). In this framework, training data is selected stochastically from the whole set and the quantity is reduced dramatically at each training epoch. SDS cuts off the training time tremendously due to the fact that less training data is used. Furthermore the SDS approach can be easily integrated into multi-GPU parallelization approaches to form a much faster BP training framework. Experiments show that the final combined framework achievs almost 3.0 times speed-up on 2 GPUs than the normal single one, at no loss of recognition accuracy. This is much faster than the recently published pipeline BP (1.9 times speedup) [5] and the ASGD (1.6 times speedup) [7] using similar hardware setup.

6. REFERENCES

[1] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. InterSpeech*, 2011, pp. 437–440.

[2] George E Dahland, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks

for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[3] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 1, pp. 213, 2002.

[5] Xie Chen, Adam Eversole, Gang Li, Dong Yu, and Frank Seide, "Pipelined back-propagation for context-dependent deep neural networks.," in *Proc. InterSpeech*, 2012.

[6] 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 *Proc. ICASSP*. IEEE, 2013, pp. 6650–6654.

[7] Shanshan Zhang, Ce Zhang, Zhao You, Rong Zheng, and Bo Xu, "Asynchronous stochastic gradient descent for dnn training," in *Proc. ICASSP*. IEEE, 2013, pp. 6660–6663.

[8] John J Godfrey and Edward Holliman, "Switchboard-1 release 2," *Linguistic Data Consortium, Philadelphia*, 1997.

[9] Dong Yu, Frank Seide, Gang Li, and Li Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Proc. ICASSP*, 2012.

[10] Abdel rahman Mohamed, George E. Dahl, and Geoffrey E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.