PARALLELIZING ADAM OPTIMIZER WITH BLOCKWISE MODEL-UPDATE FILTERING

Kai Chen[⋆] *Haisong Ding*[†*] *Qiang Huo*[⋆]

[⋆] Microsoft Research Asia, Beijing, China
[†]University of Science and Technology of China, Hefei, China
{kaic, t-haidin, qianghuo}@microsoft.com

ABSTRACT

Recently Adam has become a popular stochastic optimization method in deep learning area. To parallelize Adam in a distributed system, synchronous stochastic gradient (SSG) technique is widely used, which is inefficient due to heavy communication cost. In this paper, we attempt to parallelize Adam with blockwise model-update filtering (BMUF) instead. BMUF synchronizes model-update periodically and introduces a block momentum to improve performance. We propose a novel way to modify the estimated moment buffers of Adam and figure out a simple yet effective trick for hyper-parameter setting under BMUF framework. Experimental results on large scale English optical character recognition (OCR) task and large vocabulary continuous speech recognition (LVCSR) task show that BMUF-Adam achieves almost a linear speedup without recognition accuracy degradation and outperforms SSG-based method in terms of speedup, scalability and recognition accuracy.

Index Terms— Adam, BMUF, distributed optimization, communication efficient, scalable training.

1. INTRODUCTION

As an adaptive learning rate stochastic optimization method, Adam [1] has been widely used in deep learning area. When applied to large scale tasks, Adam is often combined with a synchronous stochastic gradient (SSG) (e.g., [2, 3, 4]) technique to speed up training process with multiple connected workers. For simplicity, we call this method Sync-Adam hereinafter. During training, it will first split a large mini-batch of data into several sub-mini-batches and distribute them to local workers, then those workers will calculate respective gradients in parallel, followed by an all-reduce operation to aggregate the results. With the aggregated gradients, an Adam step will be executed to update model parameters.

Sync-Adam's frequent all-reduce operations result in high communication overhead, which limits its speedup ratio (e.g., [4]). Essentially, Sync-Adam is a mini-batch based stochastic optimizer, while too large mini-batch will degrade model performance (e.g., [3, 5]). Sync-Adam cannot scale out to too many workers, therefore its scalability is limited. In this paper, we propose to parallelize Adam with blockwise model-update filtering (BMUF) [6]. BMUF is a general communication efficient distributed optimization framework, where each worker optimizes its local model for several steps to get a local model-update first, then local model-updates are aggregated and filtered by a historical model-update with a block momentum to update a global model. BMUF can reduce communication overhead greatly compared with SSG-based method. In [6], BMUF was demonstrated to work with a momentum-based stochastic gradient descent (SGD) local optimizer and achieve a linear speedup with little recognition accuracy degradation. It has been successfully applied to train large scale deep learning models (e.g., [7, 8, 9, 10, 11, 12, 13]). Since Adam relies on the estimation of the first and second moments of stochastic gradients to determine per-parameter step size, directly plugging Adam into BMUF framework as a local optimizer will introduce inconsistency between the estimated moments and model parameters due to the existence of block momentum. Based on the assumption of stationarity made by Adam and BMUF, we propose a strategy to achieve moment-parameter consistency and figure out a way to set hyper-parameters of Adam appropriately to make BMUF-Adam work. Because hyper-parameter for first moment estimation in Adam plays a key role, we also propose a simple yet effective trick to improve performance.

We compare the proposed BMUF-Adam method with Sync-Adam on large scale English OCR task to train connectionist temporal classification (CTC) based character models, which consist of multiple convolutional layers and bidirectional long short-term memory (BLSTM) layers. Experimental results show that BMUF-Adam achieves a linear speedup without recognition accuracy degradation up to 64 workers, and outperforms Sync-Adam in terms of speedup ratio, scalability and recognition accuracy. And we also verify the effectiveness of BMUF-Adam on a 6k-hour English large vocabulary continuous speech recognition (LVCSR) task to train deep BLSTM (DBLSTM) based acoustic models with cross entropy (CE) criterion up to 32 workers.

The remainder of this paper is organized as follows. We revisit BMUF-based training framework and present our BMUF-Adam approach in Section 2, evaluate its effectiveness on large scale OCR and LVCSR tasks in Section 3, and draw conclusions in Section 4.

2. OUR APPROACH

2.1. BMUF-based framework

In BMUF-based training framework, we have N parallel workers. A worker can be a GPU card, a compute node, etc. Given a block of training data, it will be partitioned into N splits, and each split contains τ mini-batches. Starting from a common initial model parameter $\boldsymbol{\theta}_{t-\tau}^{(\text{init})}$, all workers update their local models with respective data splits for τ steps in parallel to get N local models $\{\boldsymbol{\theta}_{t,1}, \boldsymbol{\theta}_{t,2}, \cdots, \boldsymbol{\theta}_{t,N}\}$. This procedure is called intra-block parallel optimization (IBPO). Instead of treating the averaged local models $\overline{\boldsymbol{\theta}}_t$ as updated global model $\boldsymbol{\theta}_t$, BMUF introduces a block momentum to leverage historical update information to get a better $\boldsymbol{\theta}_t$ as follows:

$$\boldsymbol{\Delta}_n = \eta \boldsymbol{\Delta}_{n-1} + \zeta(\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_{t-\tau}^{(\text{init})}) \quad (1)$$

*Haisong Ding contributed to this work when he worked as an intern in Speech Group, Microsoft Research Asia. We would like to thank Changliang Liu and Bo Ren for their help to facilitate our LVCSR experiments.

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-\tau} + \boldsymbol{\Delta}_n \tag{2}$$

where $t = n\tau$, η is the block momentum, ζ is a block learning rate, and $\boldsymbol{\Delta}_n$ is the n-th global model-update. We set $\zeta = 1$ in this paper, which is a common practice of BMUF.

As discussed in [6], block momentum can compensate per mini-batch's inadequate contribution to final model-update caused by averaging operation to improve model performance. We define a variable ρ_n to represent the number of *equivalent mini-batches* required to get $\boldsymbol{\Delta}_n$. Since

$$\boldsymbol{\Delta}_n = \sum_{i=1}^{n} \eta^{n-i} (\overline{\boldsymbol{\theta}}_{i\tau} - \boldsymbol{\theta}^{(\text{init})}_{i\tau-\tau}) \ , \tag{3}$$

we have

$$\begin{aligned} \rho_1 &= \tau \\ \rho_n &= \eta\rho_{n-1} + \tau \\ \lim_{n\to+\infty} \rho_n &= \frac{\tau}{1-\eta} \ . \end{aligned} \tag{4}$$

If η is set to $1 - \frac{1}{N}$ according to [6], $\lim_{n\to+\infty} \rho_n = N\tau$, which is equal to the number of mini-batches in a data block. This deduction shows that $\lim_{n\to+\infty} \boldsymbol{\Delta}_n$ can simulate a model-update resulting from processing an $N\tau$-mini-batch data block in serial. This conclusion requires an assumption that $(\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^{(\text{init})}_{t-\tau})$ is stationary.

If classical block momentum (CBM) is used, the initial model for next IBPO will be the same as the global model:

$$\boldsymbol{\theta}^{(\text{init})}_t = \boldsymbol{\theta}_t = \boldsymbol{\theta}^{(\text{init})}_{t-\tau} + \boldsymbol{\Delta}_n \tag{5}$$

which is equivalent to

$$\boldsymbol{\theta}^{(\text{init})}_t = \overline{\boldsymbol{\theta}}_t + \eta\boldsymbol{\Delta}_{n-1} \ . \tag{6}$$

If Nesterov block momentum (NBM) is used, the initial model for next IBPO will be

$$\boldsymbol{\theta}^{(\text{init})}_t = \boldsymbol{\theta}_t + \eta\boldsymbol{\Delta}_n \ . \tag{7}$$

Since

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-\tau} + [\eta\boldsymbol{\Delta}_{n-1} + (\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^{(\text{init})}_{t-\tau})] = \overline{\boldsymbol{\theta}}_t \ , \tag{8}$$

we have

$$\boldsymbol{\theta}^{(\text{init})}_t = \overline{\boldsymbol{\theta}}_t + \eta\boldsymbol{\Delta}_n \ . \tag{9}$$

According to [6], NBM performs better than CBM, therefore we will focus on NBM hereinafter, albeit the same method can be applied to CBM as well.

2.2. BMUF-Adam approach

Adam is an adaptive learning rate based stochastic optimizer, whose learning rates are determined by the first and second moment of stochastic gradient as follows:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \frac{E[\boldsymbol{g}_t]}{\sqrt{E[\boldsymbol{g}_t \odot \boldsymbol{g}_t]}} \tag{10}$$

where $\odot$ represents element-wise multiplication and $\boldsymbol{g}_t$ is the stochastic gradient of the t-th mini-batch. Based on an assumption that $E[\boldsymbol{g}_t]$ and $E[\boldsymbol{g}_t \odot \boldsymbol{g}_t]$ are stationary, Adam uses an exponential moving average to estimate these two moments as follows:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon} \tag{11}$$

where

$$\begin{aligned} \boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \qquad \hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t} \\ \boldsymbol{v}_t &= \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \odot \boldsymbol{g}_t, \qquad \hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1-\beta_2^t} \ . \end{aligned} \tag{12}$$

From the above update formulas, Adam depends on moment buffers $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$. When plug Adam into BMUF framework as a local optimizer, we need to handle these buffers carefully. We can simply set initial Adam buffers for next IBPO as the averaged buffers of local optimizers as follows:

$$\begin{aligned} \boldsymbol{m}^{(\text{init})}_t &= \overline{\boldsymbol{m}}_t = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{m}_{t,i} \\ \boldsymbol{v}^{(\text{init})}_t &= \overline{\boldsymbol{v}}_t = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{v}_{t,i} \end{aligned} \tag{13}$$

where $\boldsymbol{m}_{t,i}$ and $\boldsymbol{v}_{t,i}$ are moment buffers of the i-th worker. This strategy is used by [14] when they plug Adam into BMUF. Obviously $\overline{\boldsymbol{m}}_t$ and $\overline{\boldsymbol{v}}_t$ are compatible with $\overline{\boldsymbol{\theta}}_t$, and they become stale for $\boldsymbol{\theta}^{(\text{init})}_t$ due to the existence of $\eta\boldsymbol{\Delta}_n$ as Eq. (7) shows.

From the above analysis, $\boldsymbol{\Delta}_n$ can be treated as a model-update obtained by processing ρ_n mini-batches serially. So $\boldsymbol{m}^{(\text{init})}_t$ and $\boldsymbol{v}^{(\text{init})}_t$ compatible with $\boldsymbol{\theta}^{(\text{init})}_t$ can be treated as updated buffers after processing $\eta\rho_n$ mini-batches from $\overline{\boldsymbol{m}}_t$ and $\overline{\boldsymbol{v}}_t$.

According to Eq. (12), we have

$$\overline{\boldsymbol{m}}_t = \beta_1^\tau \boldsymbol{m}^{(\text{init})}_{t-\tau} + \sum_{i=1}^{\tau} \beta_1^{\tau-i}(1-\beta_1)\overline{\boldsymbol{g}}_{t-\tau+i} \ . \tag{14}$$

Since Adam assumes that $E[\boldsymbol{g}_t]$ is stationary, we have

$$E[\overline{\boldsymbol{m}}_t] = \beta_1^\tau E[\boldsymbol{m}^{(\text{init})}_{t-\tau}] + (1-\beta_1^\tau)E[\boldsymbol{g}^{(n)}] \tag{15}$$

where $E[\boldsymbol{g}^{(n)}]$ is the stochastic gradient expectation of the n-th data block. Based on these deductions, we can estimate $\boldsymbol{m}^{(\text{init})}_t$ as

$$\boldsymbol{m}^{(\text{init})}_t = \beta_1^{\tau+\eta\rho_n} \boldsymbol{m}^{(\text{init})}_{t-\tau} + (1-\beta_1^{\tau+\eta\rho_n})E[\boldsymbol{g}^{(n)}] \tag{16}$$

where

$$E[\boldsymbol{g}^{(n)}] = \frac{\overline{\boldsymbol{m}}_t - \beta_1^\tau \boldsymbol{m}^{(\text{init})}_{t-\tau}}{1-\beta_1^\tau}. \tag{17}$$

So we have

$$\boldsymbol{m}^{(\text{init})}_t = \frac{\beta_1^\tau(\beta_1^{\eta\rho_n}-1)}{1-\beta_1^\tau}\boldsymbol{m}^{(\text{init})}_{t-\tau} + \frac{1-\beta_1^{\tau+\eta\rho_n}}{1-\beta_1^\tau}\overline{\boldsymbol{m}}_t \ . \tag{18}$$

Since the respective number of mini-batches of $\boldsymbol{m}^{(\text{init})}_t$ has been changed, its bias correction term should be changed accordingly. The deduction of $\boldsymbol{v}^{(\text{init})}_t$ is analogous to $\boldsymbol{m}^{(\text{init})}_t$. The complete BMUF-Adam approach is summarized in Algorithm 1.

Averaging strategy assigns a weight of β_1^τ to $\boldsymbol{m}^{(\text{init})}_{t-\tau}$, as is shown in Eq. (14). Obviously it assigns a fixed weight, which is only determined by data split size τ, to $\boldsymbol{m}^{(\text{init})}_{t-\tau}$ no matter the number of workers used. As a result, when τ is small and N is large, $\boldsymbol{m}^{(\text{init})}_{t-\tau}$ becomes too stale for $\boldsymbol{\theta}^{(\text{init})}_t$, but still accounts for large proportion of $\boldsymbol{m}^{(\text{init})}_t$. This will hurt model performance and we will show experimental evidence in Section 3.1.2. While in our proposed strategy, the assigned weight to $\boldsymbol{m}^{(\text{init})}_{t-\tau}$ is $\beta_1^{\tau+\eta\rho_n}$. If η is set to $1 - \frac{1}{N}$,

Algorithm 1 BMUF-Adam Algorithm

Input:
number of workers N, sync period τ, block momentum η, and initial parameter $\overline{\boldsymbol{\theta}}_0$

Input:
stochastic objective function $f(\boldsymbol{\theta})$, step size α, exponential decay rates for the moment estimates $[\beta_1, \beta_2]$, small scalar ϵ, and worker id i

Init: $\boldsymbol{\Delta}_0 \leftarrow \mathbf{0}$, $\rho_0 \leftarrow 0$, $n \leftarrow 0$, $\boldsymbol{m}_0^{(\text{init})} \leftarrow \mathbf{0}$, $\boldsymbol{v}_0^{(\text{init})} \leftarrow \mathbf{0}$

Init: $\boldsymbol{\theta}_{0,i} \leftarrow \overline{\boldsymbol{\theta}}_0$, $\boldsymbol{m}_{0,i} \leftarrow \mathbf{0}$, $\boldsymbol{v}_{0,i} \leftarrow \mathbf{0}$, $t \leftarrow 0$, $k \leftarrow 0$

while $\overline{\boldsymbol{\theta}}_t$ not converged **do**
  $t \leftarrow t+1$ % number of local steps
  $k \leftarrow k+1$ % number of Adam steps w.r.t. moment buffers
  $\boldsymbol{g}_{t,i} \leftarrow \frac{\partial f_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{t-1,i}}$
  $\boldsymbol{m}_{t,i} \leftarrow \beta_1 \boldsymbol{m}_{t-1,i} + (1-\beta_1)\boldsymbol{g}_{t,i}$
  $\boldsymbol{v}_{t,i} \leftarrow \beta_2 \boldsymbol{v}_{t-1,i} + (1-\beta_2)\boldsymbol{g}_{t,i} \odot \boldsymbol{g}_{t,i}$
  $\hat{\boldsymbol{m}}_{t,i} \leftarrow \boldsymbol{m}_{t,i}/(1-\beta_1^k)$, $\hat{\boldsymbol{v}}_{t,i} \leftarrow \boldsymbol{v}_{t,i}/(1-\beta_2^k)$
  $\boldsymbol{\theta}_{t,i} \leftarrow \boldsymbol{\theta}_{t-1,i} - \alpha \hat{\boldsymbol{m}}_{t,i}/(\epsilon + \sqrt{\hat{\boldsymbol{v}}_{t,i}})$
  if t divides τ **then**
    $n \leftarrow n+1$ % number of BMUF steps
    Get $\overline{\boldsymbol{\theta}}_t$, $\overline{\boldsymbol{m}}_t$, $\overline{\boldsymbol{v}}_t$ by all-reduce
    $\boldsymbol{\Delta}_n \leftarrow \eta \boldsymbol{\Delta}_{n-1} + [\overline{\boldsymbol{\theta}}_t - (\overline{\boldsymbol{\theta}}_{t-\tau} + \eta \boldsymbol{\Delta}_{n-1})]$
    (which equals to $\boldsymbol{\Delta}_n \leftarrow \overline{\boldsymbol{\theta}}_t - \overline{\boldsymbol{\theta}}_{t-\tau}$)
    $\boldsymbol{\theta}_{t,i} \leftarrow \overline{\boldsymbol{\theta}}_t + \eta \boldsymbol{\Delta}_n$, $\rho_n \leftarrow \eta \rho_{n-1} + \tau$
    $\boldsymbol{m}_t^{(\text{init})} = \frac{\beta_1^\tau(\beta_1^{\eta\rho_n}-1)}{1-\beta_1^\tau}\boldsymbol{m}_{t-\tau}^{(\text{init})} + \frac{1-\beta_1^{\tau+\eta\rho_n}}{1-\beta_1^\tau}\overline{\boldsymbol{m}}_t$
    $\boldsymbol{v}_t^{(\text{init})} = \frac{\beta_2^\tau(\beta_2^{\eta\rho_n}-1)}{1-\beta_2^\tau}\boldsymbol{v}_{t-\tau}^{(\text{init})} + \frac{1-\beta_2^{\tau+\eta\rho_n}}{1-\beta_2^\tau}\overline{\boldsymbol{v}}_t$
    $\boldsymbol{m}_{t,i} \leftarrow \boldsymbol{m}_t^{(\text{init})}$, $\boldsymbol{v}_{t,i} \leftarrow \boldsymbol{v}_t^{(\text{init})}$, $k \leftarrow k + \eta\rho_n$
  end if
end while
return $\overline{\boldsymbol{\theta}}_t$

$\lim_{n\to+\infty}(\tau+\eta\rho_n) = N\tau$. So $\boldsymbol{m}_{t-\tau}^{(\text{init})}$'s weight decays exponentially as N increases, and consequently, its influence on $\boldsymbol{m}_t^{(\text{init})}$ will be alleviated.

A common practice when plugging momentum SGD into BMUF is setting momentum to a small value, because large momentum value leads to significant per mini-batch contribution loss caused by momentum buffer truncation before IBPO, especially when τ is small, according to the analysis of [6]. Since β_1 plays a similar role as momentum in SGD, and $\boldsymbol{m}_t^{(\text{init})}$ decays quickly, we set β_1 as 0.5 in BMUF-Adam, which is often set to 0.9 in Adam.

3. EXPERIMENTS

3.1. Large scale English OCR task

3.1.1. Experimental setup

We first evaluate the proposed approach on large scale English OCR task. Our training set contains about 1.9M English text line images, where 247k are handwritten text images cropped from whiteboard and handwritten note images, and 1.6M are printed text images cropped from Open Image dataset [15], street view images and synthetic text line images. A validation set of 204k text line images is used to guide training process. To evaluate trained model performances, we use two test sets, namely a "HWR-Test" set with 3,953 handwritten text line images and a "PRN-Test" set with 42,704 printed text line images cropped from Open Image dataset. All images are normalized to a 60-pixel height.

Table 1. Comparison of different Adam buffer estimation strategies and β_1 settings in BMUF with 16 and 32 workers, sync period $\tau = 8$.

Estimation Strategy	# workers	β_1	HWR-Test		PRN-Test	
			WER	CER	WER	CER
Baseline system			17.3	5.3	5.8	1.9
Average	16	0.9	17.9	5.4	5.8	1.9
	32	0.9	39.1	14.8	11.2	3.8
Proposed	16	0.9	17.2	5.3	5.9	1.9
		0.5	16.9	5.1	5.8	1.9
	32	0.9	18.7	5.8	6.3	2.0
		0.5	17.3	5.2	5.7	1.8

The character model we used on this task is called "DarkNet-DBLSTM", which is of the same topology as the one used in [13]. It first uses a tiny DarkNet [16], which has been successfully applied to object detection tasks, to scan input text line images and extract a feature vector sequence, which is then fed into a DBLSTM for sequence modeling. The DBLSTM contains 2 BLSTM layers, and each layer consists of 128 forward and 128 backward memory cells. This model is trained by CTC criterion, and its output dimension is 96, which represents 95 keyboard characters and a "blank" class for CTC. A hybrid 2-gram language model (LM) with a lexicon of 131k entries of words and sub-words is used in decoding [17]. The decoding setting is the same as in [13]. To evaluate the recognition performance of trained models, we use word error rate (WER in%) and character error rate (CER in %) calculated by Sclite tool [18].

Our implementation of BMUF-Adam is based on PyTorch [19] and Horovod [20], while Sync-Adam is implemented by leveraging Horovod's "DistributedOptimizer" directly. We first train a DarkNet-DBLSTM for 80 epochs with 1 Nvidia Tesla V100 GPU card using Adam as baseline system. The learning rate and its schedule is carefully tuned to achieve the lowest validation CTC loss. Sync-Adam and BMUF-Adam are evaluated with 16, 32, 64 GPU cards on a GPU cluster. Each computing node of this cluster is equipped with 4 Tesla V100 GPU cards and the computing nodes are connected by Ethernet. Learning rates of Sync-Adam are carefully tuned especially when more GPUs are used, and their schedules are the same as that of baseline system. Experiments of BMUF-Adam are warmed up with 10% training samples by Adam for 1 epoch. Learning rates and respective schedules of all BMUF-Adam experiments are exactly the same as that of baseline system. In this paper, we treat a GPU card as a worker of BMUF and block momentums for 16, 32, 64 GPU cards are set to 0.94, 0.97, 0.986 respectively. We set $\beta_1 = 0.9$ for baseline and Sync-Adam, and $\beta_2 = 0.999$ for all experiments.

3.1.2. Experimental results

We first compare the proposed method (Eq. (16)) with the method of averaging local worker's moment buffers (Eq. (13)) when plugging Adam into BMUF. We set sync period $\tau = 8$, $\beta_1 = 0.9$, and the number of workers as 16 and 32. The results are shown in Table 1. CTC best path decoding is used without LM. When #workers=16, averaging strategy performs similar with the proposed method and baseline system. As #workers becomes 32, performances of both strategies degrade, while averaging strategy performs much worse. According to our analysis in Section 2.2, from Eq. (13), the weight of stale moment term $\boldsymbol{m}_{t-\tau}^{(\text{init})}$ is only related to τ in averaging strategy, while $\eta\boldsymbol{\Delta}_n$'s equivalent mini-batches of 32-worker training is

Table 2. Performance comparison of BMUF-Adam and Sync-Adam in terms of WER (in %), CER (in %) and speedup ratio with different #workers and sync periods.

Method	# workers	τ	Epoch time(s)	Speedup	w/o LM				w/ LM			
					HWR-Test		PRN-Test		HWR-Test		PRN-Test	
					WER	CER	WER	CER	WER	CER	WER	CER
Baseline			4951.53	1.00	17.3	5.3	5.8	1.9	12.6	3.9	5.7	1.8
Sync-Adam	16	1	346.18	13.26	17.8	5.4	5.9	2.0	12.7	4.0	5.7	1.8
	32		180.65	25.41	18.3	5.5	5.9	2.0	13.0	3.9	5.8	1.9
	64		94.27	48.71	18.8	5.7	6.0	2.0	13.2	4.1	5.9	2.0
BMUF-Adam	16	8	320.34	14.33	16.9	5.1	5.8	1.9	12.3	3.6	5.7	1.8
		16	310.78	14.77	17.0	5.2	5.9	2.0	12.4	3.8	5.7	1.9
		32	303.54	15.13	17.2	5.2	5.8	1.9	12.5	3.9	5.7	1.9
	32	8	164.36	27.94	17.3	5.2	5.7	1.8	12.5	3.8	5.7	1.8
		16	157.68	29.12	17.5	5.3	5.9	1.9	12.9	4.0	5.7	1.8
		32	153.64	29.88	17.6	5.3	6.0	2.0	12.6	3.9	5.8	1.9
	64	8	84.83	54.13	17.3	5.3	5.9	1.9	12.4	3.8	5.7	1.9

Table 3. Testing WERs (in %) of models trained by BMUF-Adam with different #workers and sync periods, and their relative WERRs (in %) and speedup ratios against a baseline model trained by Adam.

# workers	τ	Cortana		Xbox		Speed-up
		WER	Relative WERR	WER	Relative WERR	
baseline		9.33	0.00	13.70	0.00	1.00
8	8	9.35	-0.21	13.70	0.00	5.25
	16	9.30	0.32	13.63	0.51	5.75
	32	9.28	0.54	13.65	0.36	5.98
16	8	9.30	0.32	13.63	0.51	8.54
	16	9.31	0.21	13.71	-0.073	10.31
	32	9.35	-0.21	13.74	-0.29	10.68
32	8	9.37	-0.43	13.70	0.00	18.69
	16	9.41	-0.86	13.77	-0.51	19.93
	32	9.44	-1.18	13.91	-1.53	23.00

almost $2\times$ of 16-worker training. This makes inconsistency between moment buffers and model parameter more serious and hurts model performance. In the proposed approach, #workers also account for stale moment weight, which alleviates the inconsistency. When #workers=32, the performance gap between the proposed approach and baseline system is caused by inadequate per mini-batch contribution due to large β_1, which can be solved by setting β_1 to a small value. As Table 1 shows, $\beta_1 = 0.5$ bridges the performance gaps. We will set β_1 as 0.5 in all the following experiments.

Then we compare BMUF-Adam with Sync-Adam using different sync periods (8, 16, 32) and worker numbers (16, 32, 64), as shown in Table 2. Sync-Adam suffers from recognition accuracy degradation as #workers increases, while BMUF-Adam achieves little performance degradation up to 64 workers. We also measure time costs to train models for 1 epoch. BMUF-Adam achieves better speedups than Sync-Adam due to much less communication overhead. Obviously, BMUF-Adam outperforms Sync-Adam in terms of recognition accuracy, scalability and speedup ratio.

3.2. Large scale LVCSR task

We also verify the effectiveness of BMUF-Adam on an LVCSR task. Training data has about 6k hours of anonymized and transcribed Microsoft production data, including Cortana [22], Xbox [23] and Conversation data, recorded in both close-talk and far-field conditions. A 6-layer DBLSTM model is used, where each layer consists of 600 forward and 600 backward cells. Each input feature vector consists of 80 log Mel filter bank features. The output layer has 9404 senone nodes. The models are trained with CE criterion by an epochwise backpropogation through time (BPTT) algorithm [21]. During decoding, a 5-gram LM with about 100 million n-grams is used.

We evaluate the trained models with Cortana and Xbox test sets, where 341k and 66k words are included respectively. WER is used as evaluation metric. The baseline model is trained with 1 Nvidia Tesla V100 GPU with Adam. This model is trained for 6 epochs and its learning rate will be halved after each epoch. BMUF-Adam experiments are warmed up with 600hr data trained by Adam for 1 epoch. BMUF-Adam experiments use exactly the same learning rate as in baseline model training, and run 6 epochs as well.

We evaluate BMUF-Adam with 8, 16 and 32 workers, and block momentums are set to 0.9, 0.94 and 0.97 respectively. Three sync periods are tried on this task. WERs on test sets, respective relative WER reductions (WERRs) and training speedup ratios against the Adam-trained baseline model are listed in Table 3. For BMUF-Adam, with the increasing number of workers, linear speedups are achieved. Compared with baseline, the speedups are not as good as what we achieved on OCR task because sequences allocated to different workers are of quite different lengths. For 8- and 16-worker experiments, BMUF-Adam achieves similar or even lower WER compared with the baseline model. For 32-worker cases, the largest performance degradation happens when $\tau = 32$, which is only 1.18% and 1.53% of WERRs on 2 test sets respectively.

4. CONCLUSION

The proposed BMUF-Adam method can achieve almost a linear speedup with little recognition accuracy degradation and outperform Sync-Adam in terms of speedup ratio, scalability and accuracy. This approach can also be applied to other model architectures not verified in this study and other moment-based optimizers such as Nadam [24] and RAdam [25], which will be our future works. If we treat each compute node with multiple GPUs as a worker, we can always use an SSG-based approach for within-node mini-batch based parallel training to further speed up and scale out BMUF-based data parallel training.

5. REFERENCES

[1] D. P. Kinagma, J. Ba, "Adam: a method for stochastic optimization," *Proc. ICLR-2015*, arXiv:1412.6980

[2] J. Chen, X. Pan, R. Monga, S. Bengio, R. Jozefowicz, "Revisiting distributed synchronous SGD," *arXiv:1604.00981*, 2016

[3] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis, "Accurate, large minibatch SGD: training Imagenet in 1 hour," *arXiv:1706.02677*, 2017

[4] M. Ott, S. Edunov, D. Grangier, M. Auli, "Scaling neural machine translation," *arXiv:1806.00187*, 2018

[5] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, "On large-batch training for deep learning: generalization gap and sharp minima," *Proc. ICLR-2017*, arXiv:1609.04836

[6] K. Chen, Q. Huo, "Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering," *Proc. ICASSP-2016*, pp.5880-5884.

[7] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, J. Mao, "Deep crossing: web-scale modeling without manually crafted combinatorial features," *Proc. SIGKDD-2016*, pp.255-262.

[8] W. Li, B. Zhang, L. Xie, D. Yu, "Empirical evaluation of parallel training algorithms on acoustic modeling," *arXiv:1703.05880*, 2017

[9] S. Zhang, M. Lei, Z.-J. Yan, L. Dai, "Deep-FSMN for large vocabulary continuous speech recognition," *Proc. ICASSP-2018*, pp.5869-5873.

[10] P. Ladkat, O. Rybakov, R. Arava, S. H. K. Parthasarathi, I-F. Chen, N. Strom, "Two tiered distributed training algorithm for acoustic modeling," *Proc. INTERSPEECH-2019*, pp.1626-1630.

[11] Z. Chen, M. Jain, Y. Wang, M. L. Seltzer, C. Fuegen, "Joint graphme and phoneme embeddings for contextual end-to-end ASR," *Proc. INTERSPEECH-2019*, pp.3490-3494.

[12] Z. Huang, T. Ng, L. Liu, H. Mason, X. Zhuang, D. Liu, "SNDCNN: self-normalizing deep CNNs with scaled exponential linear units for speech recognition," *arXiv:1910.01992*, 2019.

[13] H. Ding, K. Chen, Q. Huo, "Compressing CNN-DBLSTM models for OCR with teacher-student leanring and Tucker decomposition," *Pattern Recognition*, vol. 96, December 2019.

[14] J. Wang, V. Tantia, N. Ballas, M. Rabbat, "SlowMo: improving communication-efficient distributed SGD with slow momentum," *arXiv:1910.00643*, 2019.

[15] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijling, S. Popov, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, K. Murphy, "OpenImages: A public dataset for large-scale multi-label and multi-class image classification," https://github.com/openimages, 2017.

[16] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *Proc. CVPR-2017*, pp.6517-6525.

[17] M. Cai, W. Hu, K. Chen, L. Sun, S. Liang, X. Mo, Q. Huo, "An open vocabulary OCR system with hybrid word-subword language models," *Proc. ICDAR-2017*, pp.519-524.

[18] Sclite toolkit: http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm.

[19] A. Paszka, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, "Automatic differentiation in PyTorch," *NIPS 2017 Workshop Autodiff*

[20] A. Sergeev, M. D. Balso, "Horovod: fast and easy distributed deep learning in TensorFlow," *arXiv:1802.05799*, 2018

[21] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent neural networks and their computational complexity," in Y. Chauvin and D. E. Rumelhart (Eds.), *Backpropagation: Theory, Architectures and Applications*, pp.433-486, 1995.

[22] J. Li, R. Zhao, Z. Chen, C. Liu, X. Xiao, G. Ye, Y. Gong, "Developing far-field speaker system via teacher-student learning," *Proc. ICASSP-2018*, pp.5699-5703.

[23] C. Liu, J. Li, Y. Gong, "SVD-based universal DNN modeling for multiple scenarios," *Proc. INTERSPEECH-2015*, pp.3269-3273.

[24] T. Dozat, "Incorporating Nesterov momentum into Adam," *Workshop track - ICLR 2016*

[25] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv:1908.03265*, 2019