Training Deep Nets with larger batch size on multi-GPUs

Hongbin Liu
Tusimple co.ltd
Beijing, P. R. China
lhb8125@gmail.com

Yifan Gong
Tusimple co.ltd
Beijing, P. R. China
yifan.gong@tusimple.com

Jidong Zhai
Department of Computer Science and Technology,
Tsinghua University
Beijing, P. R. China
xxx@mail.tsinghua.edu.cn

Jiangming Jin
Tusimple co.ltd
Beijing, P. R. China
jiangming.jin@tusimple.com

ABSTRACT

As deeper and more complex models are explored, the GPU memory has reached its extremity. Data parallelism across multi-GPUs is switched to data-model coupled parallelism by the Batch Normalization (BN) layer. We proposed an adaptive global Batch Normalization (AGBN) algorithm to realize thoroughly data parallelism. In addition, the AGBN algorithm can be transformed to original BN algorithm adaptively with batch size and convergence into consideration.

KEYWORDS

batch normalization, NCCL, data parallelism, Deep learing

ACM Reference Format:
Hongbin Liu, Yifan Gong, Jidong Zhai, and Jiangming Jin. 2018. Training Deep Nets with larger batch size on multi-GPUs. In *Proceedings of ACM SC18 Conference,* Hongbin Liu (Ed.). ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep neural networks received much success in many domains in the past years, and deeper architectures are adopted because of the complex application scenarios. However, since the memory cost increases with the depth of network linearly, the batch size is limited by the GPU memory. As is well-known that the training accuracy increases with batch size generally and there exists a threshold after which the quality of model will be deteriorated[4], hence the memory becomes the bottleneck of deep learning under small-batch regime. In our semantic segmentation training on cityscapes dataset, the batch size is only 3 at most with ResNet-50 in single GPU, and the training result is greatly limited.

Data parallelism with multiple devices enables us to break through the memory limitation, whereas it is passively switch to data-model coupled parallelism because of the Batch Normalization (BN) layer[2] as is shown in figure1. The BN layer is generally introduced into the deep neural networks because of its significant margin no matter for rate of convergence or maximum of training accuracy. However, the mean and variance computed in this layer only stands for the sub mini-batch of one specific device, resulting in the separately model training of each device. Therefore, the training accuracy cannot reach the theoretical value under the original BN algorithm.

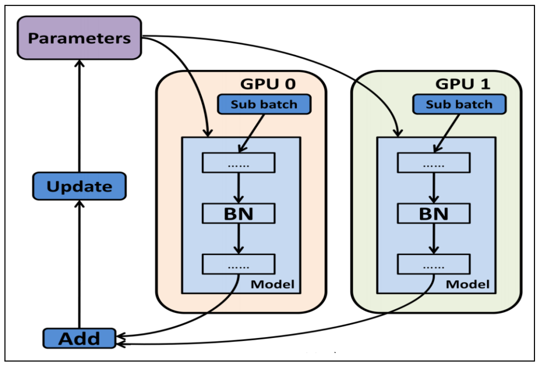

Figure 1: Data-model parallelism resulting from batch normalization layer

The aim of the study is to propose and implement a new adaptive global batch normalization (AGBN) algorithm across multi-GPUs. In the AGBN algorithm, variables such as mean and variances are computed based on the data across the whole devices to realize thoroughly data parallelism. Here the "adaptive" has two meanings: the AGBN algorithm degrades into the original BN algorithm (1) under large-batch regime to prevent the degradation in the quality of the model; (2) until the training accuracy converges to reduce the communication overhead and improve the training speed.

2 METHODS

This study is based on MXNET, an open-source Deep Learning frame[1], and BN layer is treated as an operator here. Take the forward propagation of BN layer as example, the original BN algorithm consists of only one section: normalization. In our AGBN algorithm, we add two sections, pre-pocessing and reduce, and adjust normalization section accordingly as shown in figure 2.

In the pre-pocessing section, we compute the local mean and square of mean, here "local" means the output is computed from

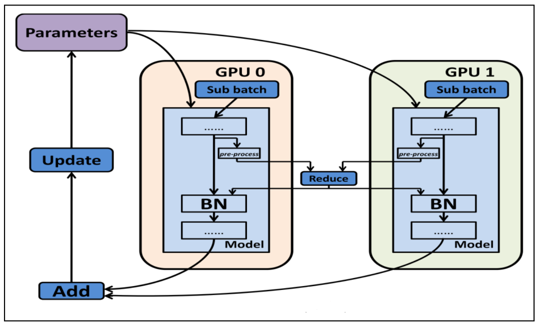

Figure 2: Schematic of the process of global BN operator

Algorithm 1 Pre-pocessing

```
1: For the $i$th GPU
2: Input: $x_{i,j}$
3: Output: $u_i$,$v_i$
4: Get **local** mean and square of mean:
```

$$u_i = \frac{1}{m_i} \sum_{j=1}^{m_i} x_{i,j} \tag{1}$$

$$v_i = \frac{1}{m_i} \sum_{j=1}^{m_i} x_{i,j}^2 \tag{2}$$

Algorithm 2 Reduce

```
1: Parameter: $BZ_{max}$,$IsConverged$
2: Input: $u_i$,$v_i$
3: Output: $u$,$v$
4: Get **global** mean and square of mean:
5: **if** $BZ < BZ_{max}$ && $IsConverged$ **then**
```

$$u = \frac{\sum_{i=1}^{n} u_i m_i}{\sum_{i=1}^{n} m_i} \tag{3}$$

$$v = \frac{\sum_{i=1}^{n} v_i m_i}{\sum_{i=1}^{n} m_i} \tag{4}$$

Algorithm 3 Batch normalization

```
1: For the $i$th GPU
2: Parameter: $\gamma$,$\beta$
3: Input: $u$,$v$,$x_{i,j}$
4: Output: $y_{i,j}$
5: Get the output of BN layer:
```

$$\sigma^2 = v - u^2 \tag{5}$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - u}{\sqrt{\sigma^2 + \varepsilon}} \tag{6}$$

$$y_{i,j} = \gamma \hat{x}_{i,j} + \beta \tag{7}$$

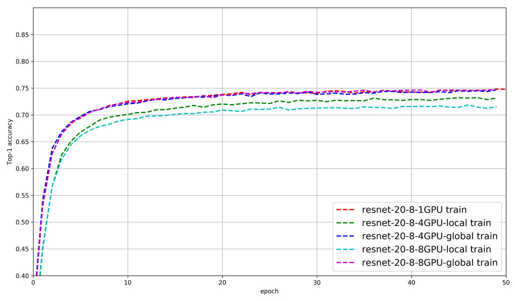

Figure 3: Cifar10: Training accuracy under different device counts and different algorithms.

the data on the ith GPU, m_i indicates the mini-batch size of the ith GPU). The last section is similar to the original BN algorithm except the operation of getting global variance and the output is identical to the original BN algorithm ignoring the reduce section.

The reduce section is implemented By NCCL, a communications library performance optimized for NVIDIA GPUs[3]. In this section we compute the global mean and square of mean and "global" means the output is based on the data of whole devices. n indicates the count of devices. This section is skipped if the batch size BZ is sufficiently large or the model has no convergence. BZ_{max} is user-defined and $IsConverged$ is determined by the convergence criterion

3 RESULTS

We evaluate the performance of global BN algorithm on eight GTX 1080Ti graphics cards. We compare the training histories of "local" and "global" cases on the identical dataset and neural network, ResNet[6]. It should be noted that the AGBN algorithm is deployed since the beginning of training because we prefer to compare the convergence curves under different BN algorithms. "Local" and "global" indicates the original BN algorithm and AGBN algorithm we proposed respectively.

Firstly, we evaluate the proposed algorithm on Cifar10 dataset[5] for image classification and the depth of ResNet is 20. The batch size is 8 for the whole devices, so the mini-batch size in a single device $BZ_{mini} = 8/n = 1$, where $n = 8$ indicates the count of devices. For the "local" cases, the training accuracy fall off about 2.1% and 4.4% for 4 devices and 8 devices, respectively. While for the "global" cases, we obtain the same training accuracy no matter for single or multiple devices.

Then we compare the training accuracy of AGBN algorithm under different batch size, and it is clear that the earning reduces monotonously as batch size grows no matter for 4GPUs or 8GPUs as the figure 4 shows. Therefore, adaptive switch from AGBN to original BN algorithm is appropriate to gain a trade-off between accuracy and efficiency.

Then we apply the proposed algorithm to self-driving scenario, where batch size is limited by memory because of the high-quality and abundant images. The cityscapes dataset is deployed to perform semantic segmantation and the depth of ResNet is 50. Only two images are assigned to one device while three pieces will run out of memory due to its high-quality, so the batch size is 16 for whole devices. With global BN algorithm, the maximum of training-accuracy

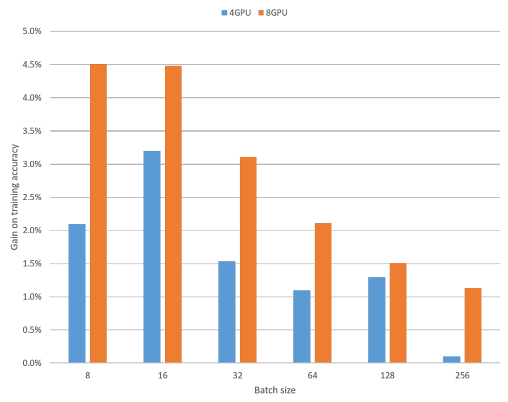

Figure 4: Gain of AGBN algorithm on training accuracy under different batch sizes

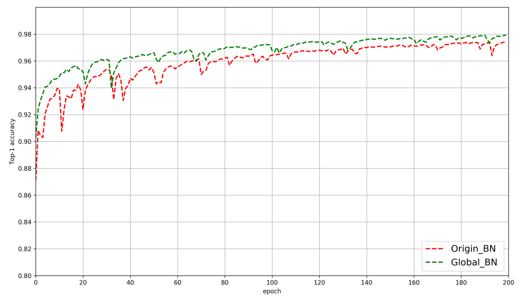

Figure 5: Cityscapes: Training accuracy under different algorithms

is increased by 0.6%, and about half of the epochs are needed under the same training accuracy as origin BN algorithm.

4 CONCLUSIONS

Since the batch size of neural network is limited by the memory size of single device and original BN algorithm results in data-model coupled parallelism in multi-GPUs, we propose and implement AGBN algorithm to eliminate the training loss introduced by BN layer under multi-GPUs. This algorithm is adaptively deployed to combine the accuracy and efficiency. For n GPUs that mini-batch size equals to BZ_{mini}, we obtain the training accuracy equivalent to the result of single GPU that batch size equals to $n * BZ_{mini}$.

ACKNOWLEDGMENTS

REFERENCES

[1] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274* (2015).

[2] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. 448–456.

[3] Sylvain Jeaugey. 2017. NCCL 2.0. (2017).

[4] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836* (2016).

[5] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html* (2014).

[6] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning.. In *AAAI*, Vol. 4. 12.