SIG Proceedings Paper in LaTeX Format*

Extended Abstract†

Anonymous Author(s)

Figure 1. This is a teaser

Abstract

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings.

CCS Concepts • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Keywords ACM proceedings, LaTeX, text tagging

ACM Reference Format:
Anonymous Author(s). 1997. SIG Proceedings Paper in LaTeX Format: Extended Abstract. In *Proceedings of ACM Woodstock conference (WOODSTOCK'97).* ACM, New York, NY, USA, **??** pages. https://doi.org/10.475/123_4

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 and computational cost increase with the depth of network linearly, data parallelism across multiple devices is deployed to speed up the training process, whereas it is passively switch to data-model coupled parallelism because of the Batch Normalization (BN) layer[**?**]. 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. The data-model coupled parallelism leads to that the model is trained with mini-batch size, rather than original batch size, and the training accuracy is influenced correspondingly.

To obtain global mean and variance, communication across all devices is required in each BN layer, which reduces the training speed greatly. 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[**?**]. Therefore, for large batch size cases such as imagenet[**?**], the training accuracy loss resulting from local

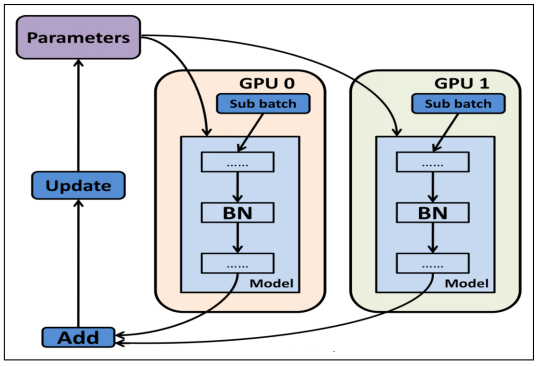

Figure 2. Data-model parallelism resulting from batch normalization layer

*Title note
†Subtitle note

WOODSTOCK'97, July 1997, El Paso, Texas USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

mean and variance is negligible compared to the communication cost. However, for memory-bounding cases such as semantic segmentation training on cityscapes dataset in self-driving field, the mini-batch size is only 3 at most with ResNet-50 in single GPU, and the training result is greatly limited under the original BN algorithm.

The aim of the study is achieving the balance between accuracy and efficiency. We need to reduce the cost between GPU's communication, so we propose and implement an 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; (2) until the training accuracy converges. The communication across devices is avoided as much as possible to improve the training efficiency.

2 Related work

Memonger[?] offers a more memory efficient training algorithm with a little extra computation cost by using automatic in-place operation and memory sharing optimization through computation graph analysis. It drops result of low cost operations and provides planning algorithm to give a sublinear memory cost model. The training upper bound of mini-batch size is increased to 8 with ResNet-50 in single GPU for cityscapes dataset.

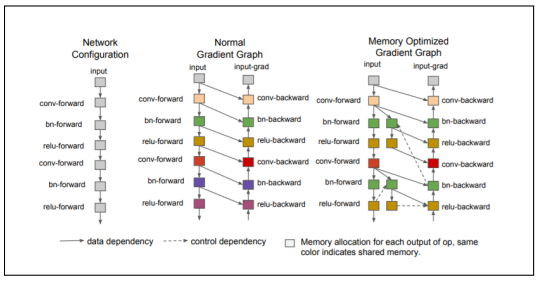

Figure 3. Memonger optimized gradient graph generation exmaple

Parameter server[?] is a framework for distributed machine learning problems, and has been deployed in MXNET, an open-source, flexible and efficient library for Deep Learning[?]. Parameter server has a server group and several worker groups. A server node maintains a part of the globally sharesd parameters and a worker stores a piece of training data. Workers only communicate with server nodes not among themselves for updating or retrieving the shared parameters.

NCCL[?], The NVIDIA Collective Communications Library, implements multi-GPU and multi-node collective communication primitives which is specially optimized for NVIDIA GPUs. It provides methods like allgather/allreduce/broadcast that are optimized to achieve high bandwidth over PCIe and NVLink high-spped interconnect, and automatic topology detection is used to determine optimal communication path.

3 Methods

This study is based on MXNET, and BN layer is treated as an operator here. The original BN algorithm consists of only one section:normalization. In our Global BN algorithm, we add two sections, pre-pocessing and reduce, and adjust normalization section accordingly to improve performance of BN algorithm.

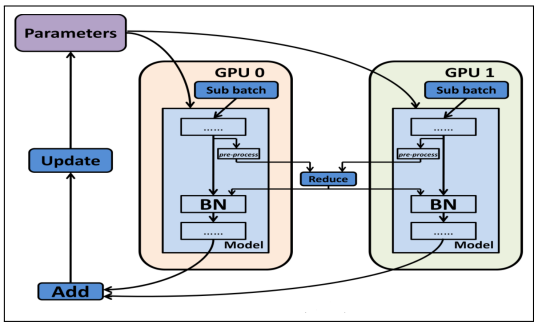

Figure 4. Schematic of the process of global BN operator

3.1 Forward modification

We first take a look at new forward prapagation in Global BN algorithm. In the pre-pocessing section, we compute the local mean and square of mean, here "local" means the output is computed from the data on the ith GPU, m_i indicates the mini-batch size of the ith GPU.

Algorithm 1 Forward Pre-processing

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

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

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

The reduce section is implemented By NCCL allreduce routine. 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.

The last part is batch normalization, which 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.

Algorithm 2 Forward Reduce

1: Input: u_i,v_i
2: Output: u,v
3: Get **global** mean and square of mean

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

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

Algorithm 3 Batch normalization

1: For the ith GPU
2: Parameter: γ,β
3: Input: u,v,$x_{i,j}$
4: Output: $y_{i,j}$
5: Get the output of BN layer:

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

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

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

3.2 Backward modification

New backward propagation also consists of three parts. In pre-preprocessing part we need to calculate the mean of gradient ψ_i and the mean of product of gradient and input data ϕ_i on ith device and store them.

Algorithm 4 Backward pre-processing

1: For the ith GPU
2: Input: $\frac{\partial l}{\partial y_{i,j}}$,$x_{i,j}(j = 0, 1, 2 \cdots n)$
3: Output: ψ_i,ϕ_i

$$\psi_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \frac{\partial l}{\partial y_{i,j}} \tag{8}$$

$$\phi_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \frac{\partial l}{\partial y_{i,j}} \cdot x_{i,j} \tag{9}$$

Then in reduce part, we will calculate the **global** mean of ψ and ϕ across whole devices

Algorithm 5 Backward reduce

1: Input: ψ_i,ϕ_i,$m_i(i = 0, 1, 2 \cdots n)$
2: Output: ψ,ϕ

$$\psi = \frac{\sum_{i=1}^{n} \psi_i m_i}{\sum_{i=1}^{n} m_i} \tag{10}$$

$$\phi = \frac{\sum_{i=1}^{n} \phi_i m_i}{\sum_{i=1}^{n} m_i} \tag{11}$$

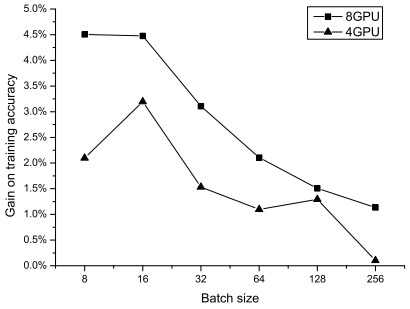

Figure 5. Gain of AGBN algorithm on training accuracy under different batch sizes

As for the batch normalization part is similar to the original BN algorithm except using ψ and ϕ calculated in Backward reduce part.

3.3 Reduce communication overhead

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.

4 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[**?**]. "Local" and "global" indicates the original BN algorithm and AGBN algorithm we proposed respectively.

We evaluate the proposed algorithm on Cifar10 dataset[**?**] for image classification and the depth of ResNet is 20. The batch size is BZ for the whole devices, so the mini-batch size in a single device $BZ_{mini} = BZ/n$, where n indicates the count of devices.

We compare the gain on training accuracy of AGBN algorithm under different batch size as the figure **??** shows, and for case $BZ = 8$, the training accuracy raises about 2.1% and 4.4% for 4 devices and 8 devices, respectively. It is clear that the earning reduces monotonously as batch size grows no matter for 4GPUs or 8GPUs. Therefore, switch from GBN to original BN algorithm under large batch size case is appropriate to gain a trade-off between accuracy and efficiency.

The communication across all devices in re duce section leads to the speed loss inevitably. Relative to the original BN algorithm, the training speed of GBN algorithm is about 75% for 4 GPUs and 70% for 8 GPUs. In our AGBN algorithm, the communication will not occur until the training process converges. For training on Cifar10 with 50 epochs, the average training speed of AGBN algorithm under different batch

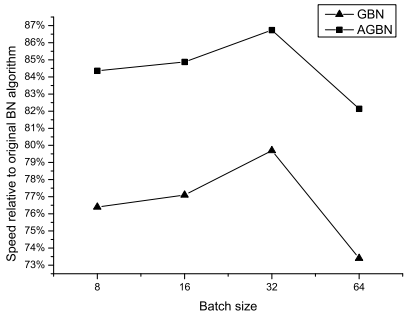

Figure 6. Gain of AGBN algorithm on training speed relative to GBN algorithm

sizes is shown in figure**??** and the training speed increases by about 10%.

5 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 balance the accuracy and efficiency. For n GPUs that mini-batch size equals to BZ_{mini}, we obtain the training accuracy nearly equivalent to the result of single GPU that batch size equals to $n * BZ_{mini}$.