SIG Proceedings Paper in LaTeX Format*

Extended Abstract†

Ben Trovato‡
Institute for Clarity in Documentation
Dublin, Ohio
trovato@corporation.com

G.K.M. Tobin§
Institute for Clarity in Documentation
Dublin, Ohio
webmaster@marysville-ohio.com

Lars Thørväld¶
The Thørväld Group
Hekla, Iceland
larst@affiliation.org

Lawrence P. Leipuner
Brookhaven Laboratories
lleipuner@researchlabs.org

Sean Fogarty
NASA Ames Research Center
Moffett Field, California
fogartys@amesres.org

Charles Palmer
Palmer Research Laboratories
San Antonio, Texas
cpalmer@prl.com

John Smith
The Thørväld Group
jsmith@affiliation.org

Julius P. Kumquat
The Kumquat Consortium
jpkumquat@consortium.net

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings.[1]

CCS CONCEPTS

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

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference Format:
Ben Trovato, G.K.M. Tobin, Lars Thørväld, Lawrence P. Leipuner, Sean Fogarty, Charles Palmer, John Smith, and Julius P. Kumquat. 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 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 efficient, 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 efficient.

*Produces the permission block, and copyright information
†The full version of the author's guide is available as `acmart.pdf` document
‡Dr. Trovato insisted his name be first.
§The secretary disavows any knowledge of this author's actions.
¶This author is the one who did all the really hard work.
[1]This is an abstract footnote

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

2 RELATED WORK

3 METHODS

This study is based on MXNET, an open-source Deep Learning frame[**?**], 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 propose a new global BN (GBN) algorithm to perform the communication and obtain the global mean and variance. The switch from original BN algorithm to GBN algorithm is determined by our SVM classifier, as shown in algorithm **??**.

The classifier is trained based on the history data including batch size and training accuracy of a specific neural network. The classifier returns *True* if the accuracy is relatively high under certain batch size, and then the parameters will be updated according to GBN algorithm. In addition, the classifier returns *False* if the batch size is large enough and the parameters will be updated according to original BN algorithm.

In the GBN algorithm, we add two sections, pre-pocessing and reduce, and adjust normalization section accordingly. 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).

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

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 as is shown in equation **??**.

$$\begin{cases} \sigma^2 = v - u^2 \\ \hat{x}_{i,j} = \frac{x_{i,j}-u}{\sqrt{\sigma^2+\varepsilon}} \\ y_{i,j} = \gamma\hat{x}_{i,j} + \beta \end{cases} \tag{2}$$

The reduce section is implemented By NCCL, a communications library performance optimized for NVIDIA GPUs[**?**]. 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 shown in **??**. n indicates the count of devices.

$$\begin{cases} u = \frac{\sum_{i=1}^{n} u_i m_i}{\sum_{i=1}^{n} m_i} \\ v = \frac{\sum_{i=1}^{n} v_i m_i}{\sum_{i=1}^{n} m_i} \end{cases} \tag{3}$$

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

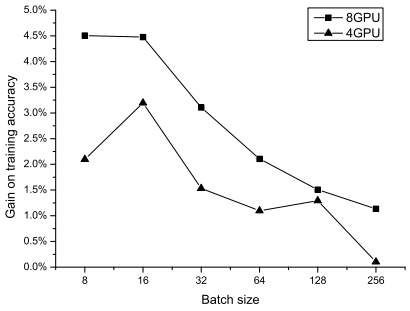

Figure 1: Gain of AGBN algorithm on training accuracy under different batch sizes

Figure 2: Gain of AGBN algorithm on training speed relative to GBN algorithm

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 reduce 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 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 $n * BZ_{mini}$.

ACKNOWLEDGMENTS