Compressed CNN Training with FPGA-based Accelerator

ABSTRACT

Training convolutional neural network (CNN) usually requires large amount of computation resources, time and power. Researchers and cloud service providers in this region needs fast and efficient training system. GPU is currently the best candidate for DNN training. But FPGAs have already shown good performance and energy efficiency as CNN inference accelerators. In this work, we design a compressed training process together with an FPGA-based accelerator. We adopt two of the widely used model compression methods, quantization and pruning, to accelerate CNN training process.

The difference between inference and training brought challenges to apply the two methods in training. First, training requires higher data precision. We use the gradient accumulation buffer to achieve low operation complexity while keeping gradient descent precision. Second, sparse network results in different types of functions in forward and back-propagation phases. We design a novel architecture to utilize both inference and back-propagation sparsity. Experimental results show that the proposed training process achieves similar accuracy compared with traditional training process with floating point data. The proposed accelerator achieves 641GOP/s equivalent performance and 3x better energy efficiency compared with GPU.

KEYWORDS

FPGA, Convolutional Neural Network, Training

ACM Reference Format:

. 2018. Compressed CNN Training with FPGA-based Accelerator. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Convolutional neural networks (CNN) has made significant performance improvement in computer vision tasks [3, 6]. However, the accuracy improvement comes at significant cost of both inference and training computation. A wide range of work have bee proposed [1, 5] to achieve fast and energy efficient inference, showing outstanding performance over GPU. But the training phase of neural networks was not fully focused on.

Training neural networks requires large amount of computation resources and can take days or weeks. Fast and energy efficient training is important for researchers and cloud service providers. GPU is currently the most suitable platform for CNN training. Recently, a number of cloud service providers including Microsoft, Amazon, Alibaba, and Huawei have deployed large FPGA clusters in their data centers. This provides the environment for FPGA-based neural network training acceleration.

The training of neural network requires similar operations as inference. Previous work on inference accelerator design gives hint on training acceleration. Model pruning and quantization have proved to be effective to reduce the requirements of computation, bandwidth and memory footprint, and have almost no effect on the performance (accuracy metric) of neural network inference [2]. A series of the accelerator designs [1, 5] follow these ideas with specific hardware. But applying pruning and quantization to training are faced with challenges.

The first challenge lies in the optimization of training. In previous work [2]

The first challenge is the higher data precision required by training. Different from inference, training process fine steps of gradient descent to improve the model accuracy. Using low precision data can lead to convergence failure or accuracy decline. The large dynamic range of gradient and feature map through different layers and different training stages also make it hard to apply quantization.

The second challenge comes from pruning. In previous work, pruning is applied after the network

Although the GPU is suitable for training of dense neural networks, it can not be a proper user of the sparsity in the sparse neural network to further improve the performance (speed metric). Customized design hardware can make better use of the sparsity of the network to achieve the purpose of improving training efficiency. As a hardware programmable devices, FPGA has been widely deployed in the server cluster [4? ?], but there is still no training architecture can take advantage of sparsity. On the other hand, the current fixed-point neural network training algorithm is still not hardware-friendly, for example, need floating point number to calculate the gradients[7].

In this paper, we propose an architecture design for sparse neural network training accelerators on the FPGA platform, which reached a high computing capability and energy efficiency. At the same time, we provide a sparse convolution neural network transformation technology, making the sparse neural network more suitable for training and inference on customized hardware platform while maintaining the accuracy at almost the same level. The main contributions of this paper are summarized as follows:

- A hardware friendly training process is proposed with all-fixed-point operations.
- Dedicated processing element (PE) on FPGA is designed to utilize the sparsity in both feed forward and back propagation phases of training.
- We analyze the limitation on unroll parameters brought by sparsity and loop dimension variety between feed forward and back propagation phases. A corresponding flexible PE array structure is proposed to improve hardware utilization ratio.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- Data arrangement and schedule strategies are proposed to improve bandwidth utilization .

Experimental results show that Proposed hardware achieves 641GOP/s equivalent performance and 3x better energy efficiency compared with GPU.

The rest of this paper is organized as follows. Section **??** introduces the background of training of a CNN. Section **??** reviews previous work on software and hardware level CNN optimization. Section **??** and section **??** introduces the proposed training process and hardware platform respectively. Experimental results are shown in section **??**. Section **??** concludes this paper.

REFERENCES

[1] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al. 2017. ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA.. In *FPGA*. 75–84.
[2] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[4] Andrew Putnam. [n. d.]. Large-scale reconfigurable computing in a Microsoft datacenter. In *Hot Chips 26 Symposium (HCS), 2014 IEEE*. IEEE, 1–38.
[5] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, et al. 2016. Going deeper with embedded fpga platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 26–35.
[6] E Shelhamer, J. Long, and T Darrell. 2017. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 4 (2017), 640.
[7] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160* (2016).