Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Demonstrations Track

Efficient and Modularized Training on FPGA for Real-time Applications

Shreyas Kolala Venkataramanaiah[1] , **Xiaocong Du**[1] , **Zheng Li**[2] ,
Shihui Yin[1] , **Yu Cao**[1,2] and **Jae-sun Seo**[1]

[1]School of ECEE, Arizona State University, Tempe, AZ, USA
[2]School of CIDSE, Arizona State University, Tempe, AZ, US

{skvenka5, xiaocong, zheng.li.95, syin11, ycao, jseo28}@asu.edu

Abstract

Training of deep Convolution Neural Networks (CNNs) requires a tremendous amount of computation and memory and thus, GPUs are widely used to meet the computation demands of these complex training tasks. However, lacking the flexibility to exploit architectural optimizations, GPUs have poor energy efficiency of GPUs and are hard to be deployed on energy-constrained platforms. FPGAs are highly suitable for training, such as real-time learning at the edge, as they provide higher energy efficiency and better flexibility to support algorithmic evolution. This paper first develops a training accelerator on FPGA, with 16-bit fixed-point computing and various training modules. Furthermore, leveraging model segmentation techniques from Progressive Segmented Training, the newly developed FPGA accelerator is applied to online learning, achieving much lower computation cost. We demonstrate the performance of representative CNNs trained for CIFAR-10 on Intel Stratix-10 MX FPGA, evaluating both the conventional training procedure and the online learning algorithm. The demo is available at https://github.com/dxc33linger/PSTonFPGA_demo.

1 Introduction

The recent development of machine learning algorithms and computing hardware has enabled many modern edge applications, such as autonomous vehicles, surveillance drones, and robots. Training of these ML-edge applications is typically performed on cloud servers because of their high computing capability. Sending the data to the cloud incur large latency overhead and raises privacy/security concerns. Training at the edge enables limited data exchange with the cloud and helps in personalizing, improving energy efficiency and protecting the private data. The edge devices are also preferred to handle the learning from a data stream over time locally and in real-time, *i.e*. online learning.

In order to enable online learning at the edge for real-time applications, several major challenges need to be solved: (1) When new data arrives in a stream, there is very limited or even no access to previously learned data. Yet the learned knowledge (*i.e*. network parameters) from previous data should not be forgotten (*i.e*. overwritten or deteriorated due to the learning of new observations) [Kirkpatrick *et al.*, 2017; Chaudhry *et al.*, 2018; Li and Hoiem, 2017; Rebuffi *et al.*, 2017]. (2) The network should be able to update its parameters according to the incoming data stream. It is preferred that such adaption is completed locally and in real-time for an edge device [Du *et al.*, 2019a; Venkataramanaiah *et al.*, 2019]. (3) Although GPUs provide remarkably high parallelism and throughput making it a viable option for real-time learning, they are not suitable for power constrained platforms. Hardware design for flexible and energy efficient training at the edge is challenging due to design complexity, large computation/memory/power requirement and other resource budges [Han *et al.*, 2016; Han *et al.*, 2015; Du *et al.*, 2019b; Liu *et al.*, 2015; Li *et al.*, 2015].

FPGAs are well suited to exploit these algorithmic advances and tackle the above-mentioned challenges as they provide high energy efficiency, good flexibility, and large on-chip and off-chip memories. Several FPGA based training/inference accelerators have been proposed [Liu *et al.*, 2018; Gomperts *et al.*, 2011; Rafael *et al.*, 2005; Liu *et al.*, 2017; Zhao *et al.*, 2016; Choi *et al.*, 2018; Guo *et al.*, 2019] but they fail to show end-to-end training capability. [Venkataramanaiah *et al.*, 2019] proposes an FPGA based fixed-point training accelerator capable of demonstrating end-to-end training. An RTL generator is used to generate the architecture according to the network structure and design requirements. The proposed accelerator can also support novel training methodologies like PST and provides great flexibility to exploit optimizations.

In this work, we demonstrate online CIFAR-10 CNN learning on an FPGA based 16-bit fixed-point training accelerator [Venkataramanaiah *et al.*, 2019] on Intel Stratix-10 MX FPGA [Deo *et al.*, 2016]. The proposed accelerator is augmented to support PST [Du *et al.*, 2019a] which further improves the performance of online CNN training. We also demonstrate the PST algorithm by deploying the pretrained, segmented model (*i.e*. selected weights are frozen in the network) on the FPGA and training the network with new real-time data.

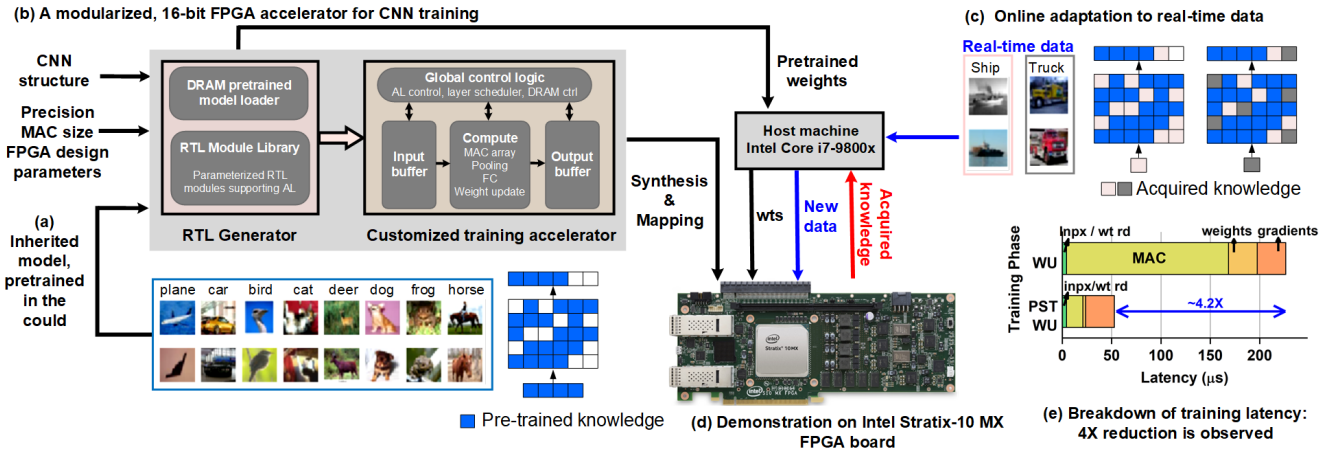

Figure 1: The demonstration system consists of Intel Straix-10 FPGA initialized with pre-trainined model parameters. The new data is streamed to the FPGA and learned locally in real-time using PST algorithm.

2 System Overview

2.1 Demo System

Figure 1 depicts the overall system setup to demonstrate training of CNNs using PST algorithm. First, a large amount of knowledge is pretrained and important model parameters are frozen in the network (Figure 1a) following the process described in [Du *et al.*, 2019a]. The pretrained model is sent to RTL generator which generates the customized training accelerator and HBM2 memory initialization files (Figure 1b). The generated training accelerator uses the frozen weights stored in HBM2 and performs the inference. This forms an inherited model, which is used to acquire new knowledge; the model is then exposed to a new unlearned data stream and the network parameters are updated accordingly in real-time on the FPGA (Figure 1c). The entire system is demonstrated on Intel Stratix-10 MX FPGA board (Figure 1d). Benefiting from the model inheritance, the online training of new observations requires much less computation cost and lower latency, as compared to traditional continual learning scheme that learns from scratch. PST greatly aids in improving the computation cost by updating only the required weights instead of updating all the network parameters in the traditional training schemes. Latency breakdown graph (Figure 1e) shows the latency benefit of using PST compared to conventional training in the weight update (WU) phase.

2.2 CNN Training Hardware

The RTL generator generates the CNN training hardware using the high-level network details given by the user. It uses a highly parameterized handwritten RTL module library designed to support various layers of CNN training. The user can also reconfigure the architecture by changing the FPGA design parameters such as precision, MAC array size, tiling, and layer scheduling. To support novel training algorithms like PST, the RTL generator is designed to read the pretrainied CNN model and generate the HBM2 initialization files to load the frozen weights.

The CNN training hardware is flexible to support forward pass (FP), backward pass (BP) and weight update (WU) phases of training. The hardware consists of a global control logic that governs all the modules and enables layer by layer execution by using the parameters generated by the RTL generator. The HBM2 stores all the initial weight parameters (or weights from a pretrained model), activations and computed weight gradients/new weights. The input/output on-chip buffer is used to store the input/output parameters required for a given layer. For example, while computing a convolution layer the input buffers stores the input activations, weights and output buffers store the convolved outputs.

The core compute blocks reads the data from the input buffers and perform the computation based on the layer type and the outputs are sent to output buffers. The convolution block uses a 2D systolic MAC array flexible to support all three phases of the training. The weight update block computes and accumulates the weight gradients. At the end of the batch, the accumulated weight gradients are scaled and new weights are computed using the stochastic gradient descent algorithm. To support PST where we need to only update the selected weights, the control logic was augmented to skip the HBM2 access if the frozen weights thereby reducing the off-chip communication. The weight updates and weight gradient computation was performed only for the selected weights.

2.3 Demonstration Setup

We showcase our system with CIFAR-10 [Krizhevsky *et al.*, 2009] dataset. The CIFAR-10 dataset consists of 60,000 32×32 color images in 10 classes, with 5,000 training images and 1,000 testing images per class. The classes include common objects such as plane, bird, truck, etc. We demonstrate online learning on FPGA with a CNN structure of 16C3-16C3-MP-32C3-32C3-MP-64C3-64C3-MP-FC, where 'NCk' represents convolution layer with 'N' output feature maps and kernel size of 'k', 'MP' represents max pooling layer and 'FC' represents a fully connected layer. The accelerator was synthesized by Intel Quartus 19.2 at 150 MHz frequency. We used Stratix-10 MX equipped with HBM2 as the target hardware and Intel(R) Core(TM) i7-9800X as a host machine. All the parameters used 16-bit fixed point precision.

Acknowledgments

This work was supported in part by the Semiconductor Research Corporation (SRC) and DARPA. It was also partially supported by National Science Foundation (NSF) under CCF #1715443.

References

[Chaudhry *et al.*, 2018] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[Choi *et al.*, 2018] Seungkyu Choi, Jaehyeong Sim, Myeonggu Kang, and Lee-Sup Kim. TrainWare: A memory optimized weight update architecture for on-device convolutional neural network training. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, 2018.

[Deo *et al.*, 2016] Manish Deo, Jeffrey Schulz, and Lance Brown. Intel stratix 10 mx devices solve the memory bandwidth challenge. *Intel White Paper*, 2016.

[Du *et al.*, 2019a] Xiaocong Du, Gouranga Charan, Frank Liu, and Yu Cao. Single-net continual learning with progressive segmented training. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1629–1636, Dec 2019.

[Du *et al.*, 2019b] Xiaocong Du, Zheng Li, Yufei Ma, and Yu Cao. Efficient network construction through structural plasticity. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(3):453–464, 2019.

[Gomperts *et al.*, 2011] Alexander Gomperts, Abhisek Ukil, and Franz Zurfluh. Development and implementation of parameterized FPGA-based general purpose neural networks for online applications. *IEEE Transactions on Industrial Informatics*, 7(1):78–89, 2011.

[Guo *et al.*, 2019] Kaiyuan Guo, Shuang Liang, Jincheng Yu, Xuefei Ning, Wenshuo Li, Yu Wang, and Huazhong Yang. Compressed cnn training with fpga-based accelerator. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 189–189, 2019.

[Han *et al.*, 2015] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[Han *et al.*, 2016] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[Kirkpatrick *et al.*, 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[Li and Hoiem, 2017] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[Li *et al.*, 2015] Zheng Li, Chenchen Liu, Yandan Wang, Bonan Yan, Chaofei Yang, Jianlei Yang, and Hai Li. An overview on memristor crossabr based neuromorphic circuit and architecture. In *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 52–56. IEEE, 2015.

[Liu *et al.*, 2015] Chenchen Liu, Bonan Yan, Chaofei Yang, Linghao Song, Zheng Li, Beiye Liu, Yiran Chen, Hai Li, Qing Wu, and Hao Jiang. A spiking neuromorphic design with resistive crossbar. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015.

[Liu *et al.*, 2017] Zhiqiang Liu, Yong Dou, Jingfei Jiang, Qiang Wang, and Paul Chow. An FPGA-based processor for training convolutional neural networks. In *Proceedings of the International Conference on Field Programmable Technology (ICFPT)*, pages 207–210, 2017.

[Liu *et al.*, 2018] Qiang Liu, Jia Liu, Ruoyu Sang, Jiajun Li, Tao Zhang, and Qijun Zhang. Fast neural network training on FPGA using quasi-newton optimization method. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(8):1575–1579, 2018.

[Rafael *et al.*, 2005] G Rafael, C Ricardo, C Joaquín, C Angel, and Wakamura M Maeda. FPGA implementation of a pipelined on-line backpropagation. *Journal of VLSI Signal Processing*, 40(2):189–213, 2005.

[Rebuffi *et al.*, 2017] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[Venkataramanaiah *et al.*, 2019] Shreyas Kolala Venkataramanaiah, Yufei Ma, Shihui Yin, Eriko Nurvithadhi, Aravind Dasu, Yu Cao, and Jae-sun Seo. Automatic compiler based fpga accelerator for cnn training. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 166–172. IEEE, 2019.

[Zhao *et al.*, 2016] Wenlai Zhao, Haohuan Fu, Wayne Luk, Teng Yu, Shaojun Wang, Bo Feng, Yuchun Ma, and Guangwen Yang. F-CNN: An FPGA-based framework for training convolutional neural networks. In *Proceedings of the IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 107–114, 2016.