2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)

SimBNN: A Similarity-Aware Binarized Neural Network Acceleration Framework

Cheng Fu[1], Shilin Zhu[1], Huili Chen[1], Farinaz Koushanfar[1], Hao Su[1] and Jishen Zhao[1]
[1]University of California, San Diego
{*cfu, shz338, huc044, haosu, farinaz, jzhao*}*@ucsd.edu*

I. Introduction

Recent advances of Deep Neural Networks (DNNs), especially Convolutional Neural Networks (CNNs), is empowered by the advance of hardware accelerators, and neural network accelerators that are integrated into various embedded processors. There are three major challenges of accelerating classical floating-point CNNs: *(C1)* off-chip Dynamic Random-Access Memory (DRAM) access overhead, *(C2)* on-chip memory access, and *(C3)* computation cost [1].

Quantization of network parameters and input data is one of the most promising approaches to solve the above challenge. Recently proposed algorithms have successfully reduced the bitwidth of network weights while maintaining a high precision for image classification tasks [2, 3]. In particular, Binary Neural Network (BNN), a binary quantized version of CNN, attracts extensive attention, because it can significantly alleviate the DRAM access overhead (tackling C1). In BNN, the multiplication and addition in traditional floating-point CNN inference are replaced by more power-efficient, compact bit operations, which are suitable for reconfigurable logic like FPGA. However, though the bitwidth in both computation and storage has been considerably reduced in BNN [4], the total number of Multiplication and ACcumulation (MAC) operations still remains the same. For example, binarized VGG-16 neural network has reduced the network storage by around 5× but it still requires many computations (∼15.5 Giga MAC operations) to evaluate one input image [**?** 5].

Our goal in this paper is to address the remaining issues of BNN inference by reducing on-chip memory access *(tackling C2)* and computation cost *(tackling C3)*. We leverage the key property of BNN: As the input and kernel weights of BNN are -1/+1, they both exhibit high similarity. Intuitively, input similarity originates from the spatial continuity of the image to classify, while kernel similarity comes from the correlation of features represented by different binarized weight kernels. To prove this hypothesis with BNN, we investigate the similarity in input data and weight kernel across a variety of applications and networks. The kernel similarity is computed based on our proposed re-ordering algorithm. The average input and kernel similarity ratio is ranging from 78%∼84% and 59%∼64% for network models [3]. However, if the weights of BNN are binarized, whereas the input activations are finely quantized – which is a favorable setting in many current works [6] – the kernel similarity is much higher than the input similarity. Based on these observations, we propose *SimBNN*, a BNN acceleration framework that leverages input and kernel similarities to significantly reduce the number of MAC operations and memory access during inference. Instead of computing the XNOR between input activation and kernel weights at each cycle, SimBNN first checks the input or weight dissimilarity between the current operand and the previous one. The output from the previous computation is reused, if similarity is identified. In this case, memory access and MAC operations can be bypassed. In fact, our analysis shows that 78% of the computation and 42% of the memory access on average can be skipped using this method. As a result, evaluation power consumption can be significantly saved. SimBNN is also 2.2× more area-efficient compared to the state-of-the-art BNN accelerator [7]. In addition, we observe that similarities vary across different BNN models. Therefore, we provide analysis and insights to select the optimal one from our proposed two reuse strategies, namely, SimBNN customizes the hardware design based on a given BNN through performance profiling.

References

[1] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 367–379.

[2] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *CoRR*, vol. abs/1612.01064, 2016. [Online]. Available: http://arxiv.org/abs/1612.01064

[3] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[4] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 65–74.

[5] H. Kim, J. Sim, Y. Choi, and L.-S. Kim, "A kernel decomposition architecture for binary-weight convolutional neural networks," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 60.

[6] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *CoRR*, vol. abs/1511.00363, 2015. [Online]. Available: http://arxiv.org/abs/1511.00363

[7] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable fpgas," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 15–24.

2576-2621/19/$31.00 ©2019 IEEE
DOI 10.1109/FCCM.2019.00060