HARDWARE-FRIENDLY CONVOLUTIONAL NEURAL NETWORK WITH EVEN-NUMBER FILTER SIZE

Song Yao, Kaiyuan Guo, Jianqiao Wangni, Yu Wang
Center for Brain-Inspired Computer Research
Department of Electronic Engineering, Tsinghua University
Beijing 100084, China
{songyao, yu-wang}@mail.tsinghua.edu.cn,
{gky15, wnjq11}@mails.tsinghua.edu.cn

Song Han
Department of Electrical Engineering
Stanford University
Stanford, CA 94315, USA
songhan@stanford.edu

ABSTRACT

Convolutional Neural Network (CNN) has led to great advances in computer vision. Various customized CNN accelerators on embedded FPGA or ASIC platforms have been designed to accelerate CNN and improve energy efficiency. However, the odd-number filter size in existing CNN models prevents hardware accelerators from having optimal efficiency. In this paper, we analyze the influences of filter size on CNN accelerator performance and show that even-number filter size is much more hardware-friendly that can ensure high bandwidth and resource utilization. Experimental results on MNIST and CIFAR-10 demonstrate that hardware-friendly even kernel CNNs can reduce the FLOPs by $1.4\times$ to $2\times$ with comparable accuracy; With same FLOPs, even kernel can have even higher accuracy than odd size kernel.

1 INTRODUCTION

In recent years, Convolutional Neural Network (CNN) has achieved great success in computer vision area. State-of-the-art performance of image classification and object detection are both driven by CNN ((He et al., 2015; Girshick et al., 2014; Ren et al., 2015)). However, the energy efficiency of existing hardware such as GPU is relatively low, thus researchers have proposed various customized CNN accelerator designs on FPGA or ASIC.

Efficient processing engine (PE) is vital to CNN accelerators. Architecture with few complex PEs ((Qiu et al., 2016; Sim et al., 2016)) or with many simple compute elements (Chen et al. (2016)) have been proposed. Special architectures such as a dynamically configurable architecture and a specific architecture for sparse compressed NN were also proposed ((Chakradhar et al., 2010; Han et al., 2016a)).

The efficiency of memory system in CNN accelerators also significantly affects the performance. The tiling strategy and data reuse are useful to reduce the total communication traffic (Chen et al. (2014a); Qiu et al. (2016)). Storing all the CNN model with on-chip memory can help minimize energy of memory access (Chen et al. (2014b); Du et al. (2015); Han et al. (2016a)). Compression and decompression techniques(Zhang et al. (2015); Chen et al. (2016); Han et al. (2015; 2016b)) and data quantization (Qiu et al. (2016)) are also useful techniques to improve bandwidth utilization.

Though techniques have been proposed to improve the performance of customized CNN accelerators, the odd-number filter size in existing CNNs still hinders higher hardware acceleration efficiency. From algorithm aspect, the advantage of odd-number filter size is obvious: symmetry. However, customized CNN accelerators may perform better with even-number Conv filters such as 2×2 and 4×4 and can achieve better configurability and resource utilization.

In this paper, we investigate the effects of Conv filter size on hardware acceleration efficiency of CNN accelerators. We propose the hardware-friendly CNN with only even-number Conv filters to maximize the efficacy of CNN accelerators. We show that hardware-friendly CNNs can achieve comparable or even better accuracy compared with CNN with odd-number Conv filters on MNIST and CIFAR-10.

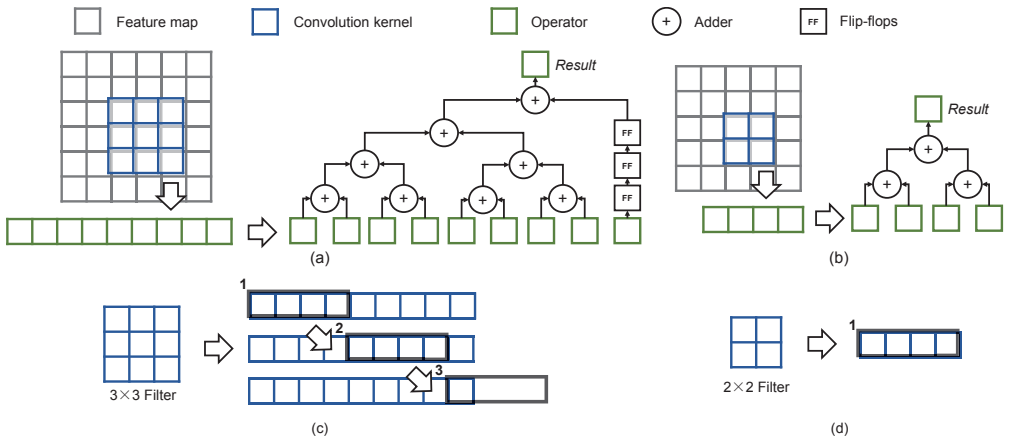

Figure 1: Influences of filter size on hardware design: Adder tree structure with (a) 3×3 filter and (b) 2×2 filter; Memory access pattern with (c) 3×3 filter and (d) 2×2 filter.

2 Influences of Filter Size on Hardware Acceleration Efficiency

2.1 Computation Logic Design

The combination of many multipliers and an adder tree is a fundamental unit for accelerating Conv layers. For the adder tree, if the number of data in a filter is not 2^n form, there will be extra register used. As shown in Figure 1 (a), the 3 extra register sets are needed to implement an adder tree with 9 inputs. If 16-bit quantization (i.e. each parameter is represented with 16 bits) is employed, this means $16 \times 3 = 48$ additional registers are needed. For a 2×2 filter, as shown in Figure 1 (b), there is no such waste of registers.

2.2 Data Distributor Design

State-of-the-art CNNs for large-scale object recognition tasks are too large to be store the model on-chip. Since CNN models are usually stored in the external memory, the bandwidth utilization efficiency is seriously concerned. Typically, DRAM offers 64-bit or 128-bit data port. If the length of the fetched data is folds of the data port width, full bandwidth utilization can be ensured.

It is hard to ensure high bandwidth utilization with odd-number filters. For a 3×3 filter with 16-bit quantization, 144 bits are needed to store the weights in a filter. For a 64-bit port, to load 144 bits, triple memory accesses are needed, as shown in Figure 1 (c), and the highest possible bandwidth usage is only 75%. For a 128-bit port, the highest possible bandwidth usage is only 56.25%. To fully utilize the bandwidth when the filters are in odd-number sizes, the data distributor design will be quite complicated.

Even-number filters can help ensure the bandwidth utilization. For a $2N \times 2N$ filter with 16-bit quantization, where N is a natural number, the total number of bits is $64N^2$. For a 64-bit port, the bandwidth utilization is definitely 100%. For a 128-bit port, the bandwidth usage can be up to 100% (loading two filters at the same time), 90%, and 96% when N is 1, 2, and 3 respectively. An example is shown in Figure 1 (d) where the filter size is 2×2. When the data port width is 64-bit and 16-bit quantization is employed, only one-time memory access is needed to load all the weights.

3 Hardware-Friendly Convolutional Neural Network

Since CNNs with even-number Conv filters can help improve the efficiency of customized CNN accelerators, we propose the hardware-friendly CNN with only even-number Conv filters. In this section, we evaluate the performance of hardware-friendly CNNs on MNIST (LeCun et al. (1998)) and CIFAR-10 (Krizhevsky & Hinton (2009)). All experiments are done with MXnet Chen et al. (2015). The experiment platform consists of an Intel Xeon E5-2690 CPUs@2.90GHz and the 2 NVIDIA TITAN X GPUs.

The notation is: *MP* means max pooling, *FC* means Fully-Connected layer, *lr* is the initial learning rate, *lr-factor* is the factor that times the learning rate for every *lr-factor-epoch*, and *batch-size* is the number of images in each mini batch. When training the networks, no data augmentation, pre-processing, or pre-training is employed.

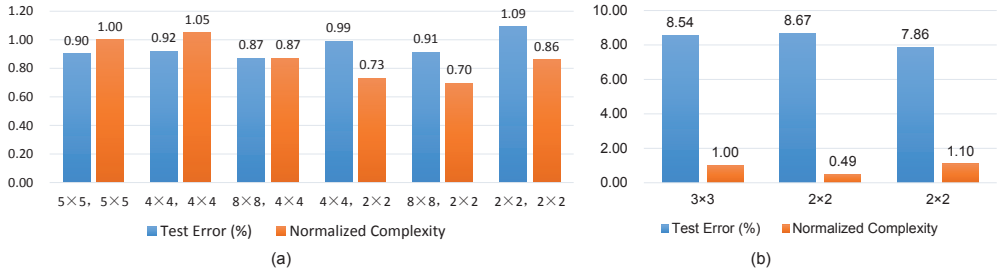

Figure 2: Test error and normalized computational complexities (FLOPs) of (a) LeNet5 on MNIST and (b) VGG11-Nagadomi on CIFAR-10. With comparable accuracy, even kernel can reduce the FLOP by 50% on cifar dataset and 30% on mnist dataset; with comparable FLOPs, even kernel can have higher accuracy than odd size kernel. .

3.1 MNIST

For experiments on MNIST, we used the LeNet (LeCun et al. (1998)). The architecture of the original LeNet is:

$$20Conv5 \rightarrow Tanh \rightarrow MP2 \rightarrow 50Conv5 \rightarrow Tanh \rightarrow MP2 \rightarrow FC500 \rightarrow Tanh \rightarrow FC10.$$

We train the LeNet for 300 epochs, in which the *lr* is 0.002, *lr-factor* is 0.995, *lr-factor-epoch* is 1, and *batch-size* is 128.

We report he best validation error rate of LeNet with different settings on MNIST in Figure 2 (a), in which blue and orange columns represent test errors and computational complexities respectively. As shown in the figure, replacing the 5×5 Conv filters in LeNet with 4×4 or other even-number ones does not introduce high error rate. Since smaller Conv filter demands few multiplications in one Conv operation, generally, the total number of operations can be reduce by using smaller even-number Conv filters. But the increase of feature map size lead to the increase of total computations.

3.2 CIFAR-10

We used the VGG11-Nagadomi network (nag) in experiments on CIFAR-10. The architecture of the original VGG11-Nagadomi network is:

$$2 \times (64Conv3 \rightarrow ReLU) \rightarrow MP2 \rightarrow 2 \times (128Conv3 \rightarrow ReLU) \rightarrow MP2 \rightarrow$$
$$4 \times (256Conv3 \rightarrow ReLU) \rightarrow MP2 \rightarrow 2 \times (FC1024 \rightarrow ReLU) \rightarrow FC10.$$

We train the VGG11-Nagadomi for 2000 epochs, in which the *lr* is 0.01, *lr-factor* is 0.995, *lr-factor-epoch* is 2, and *batch-size* is 256.

Results of VGG11-Nagadomi network on CIFAR-10 are shown in Figure 2 (b). For the original VGG11-Nagadomi network, the validation error on CIFAR-10 is 8.54%. After replacing the 3×3 Conv filters with 2×2 ones, the size of feature maps in the network changes. We remove the padding in the later Conv layer in each pair of Conv layers to ensure the input feature map of each MP layer remains the same. As the middle columns in Figure 2 (b) show, the validation error rises to 8.67%, but the total computations is reduce to 49% of the original network. Since the total computation number is reduced when simply replacing 3×3 Conv filters with 2×2 ones, we increase the filter numbers and the out feature vector length of FC layers by 1.5× to balance the total operations. In this case, the total computations rise to 1.10× compared with the original network but the test error is reduced to 7.86%. We notice that, keeping the original ratio between filter number in different layers when balancing the total computations may be favorable to achieve the best accuracy.

4 Conclusion

In this paper we propose hardware-friendly convolution neural network using even-sized kernel and its advantage over traditional odd-sized kernel. We analyzed the hardware benefit of even sized kernel w.r.t both arithmetic unit and memory system. Even sized kernel greatly reduces the number of computation while maintaining comparable prediction accuracy: on mnist on cifar-10 it reduced the computation by 1.4× to 2× with less than 0.1% loss of accuracy. On the other hand, shrinking the kernel from 3x3 to 2x2 at the same time of increasing the number of channels, such that the total number of computation remains the same, will result in better prediction accuracy. This will facilitate building hardware inference engine with higher efficiency.

REFERENCES

URL https://github.com/nagadomi/kaggle-cifar10-torch7.

Srimat Chakradhar, Murugan Sankaradas, Venkata Jakkula, and Srihari Cadambi. A dynamically configurable coprocessor for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 38, pp. 247–257. ACM, 2010.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ACM SIGPLAN Notices*, volume 49, pp. 269–284. ACM, 2014a.

Yu-Hsin Chen, Tushar Krishna, Joel Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *ISSCC*. IEEE, 2016.

Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao: A machine-learning supercomputer. In *MICRO*, pp. 609–622. IEEE, 2014b.

Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. Shidiannao: shifting vision processing closer to the sensor. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pp. 92–104. ACM, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pp. 580–587. IEEE, 2014.

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

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. *arXiv preprint arXiv:1602.01528*, 2016a.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*, 2016b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, Yu Wang, and Huazhong Yang. Going deeper with embedded fpga platform with convolutional neural network. In *ACM Symposium on Field Programmable Gate Array (FPGA)*, pp. 1–12. ACM, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

Jaehyeong Sim, Jun-Seok Park, Minhye Kim, Dongmyung Bae, Yeongjae Choi, and Lee-Sup Kim. A 1.42top-s/w deep convolutional neural network recognition processor for intelligent ioe systems. In *ISSCC*. IEEE, 2016.

Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of ISFPGA*, pp. 161–170. ACM, 2015.