Efficient Deep Convolutional Neural Networks Accelerator without Multiplication and Retraining

Conference Paper · April 2018

CITATIONS
0

READS
41

3 authors, including:

Weihong Xu
Southeast University (China)
7 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Chuan Zhang
Southeast University (China)
111 PUBLICATIONS **482** CITATIONS
SEE PROFILE

EFFICIENT DEEP CONVOLUTIONAL NEURAL NETWORKS ACCELERATOR WITHOUT MULTIPLICATION AND RETRAINING

Weihong Xu[1,2]*, Xiaohu You*[2]*, and Chuan Zhang*[1,2,*]

[1]Lab of Efficient Architectures for Digital-communication and Signal-processing (LEADS)
[2]National Mobile Communications Research Laboratory, Southeast University, Nanjing, China
Email: {wh.xu, xhyu, chzhang}@seu.edu.cn

ABSTRACT

Recently, low-precision weight method has been considered as a promising scheme to efficiently implement inference of deep convolutional neural networks (DCNN). But it suffers from expensive retraining cost as well as some accuracy degradation. In this paper, a low-bit and retraining-free quantization method, which enables DCNNs to deal inference with only shift and add operations, is proposed. The efficiency is demonstrated in terms of power consumption and chip area. Huffman coding is adopted for further compression. Then by exploring two-level systolic, an efficient hardware accelerator is introduced with respect to the given quantization strategy. Experiment results show that our method achieves higher accuracy than other low-precision networks without retraining process on ImageNet. $5\times$ to $8\times$ compression is obtained on popular models compared to full-precision counterparts. Furthermore, hardware implementation indicates good reduction of slices whereas maintaining throughput.

***Index Terms*—** Deep neural networks, hardware acceleration, systolic, low-precision weights.

1. INTRODUCTION

Deep convolutional neural networks (DCNNs) push various complex tasks a step further, including image recognition [1–3], machine translation, and so on. DCNNs show stronger competitiveness driven by up-to-date large scale datasets, such as *ImageNet* [4]. The deployment of DCNNs is divided into training and inference. After being trained, DCNN inference is crucial in commercial application scenarios.

State-of-the-art DCNNs achieve superior performance partly through increasing the network scale. Consequently, tremendous computing and storage resources are required in both training and inference phases. Focusing on efficiently performing the inference and model compression, a lot of work has been done based on low-precision weights representation. The low-precision representation allows the hardware to implement multiply-accumulate (MAC) operation without bulky multipliers. [5] proposes a method to perform training and inference with binary weights and activations. XNOR-Net in [6] evidently reduces the storage and computation complexity by binarizing both inputs and weights. To address the performance degradation of binary scheme [5], ternary weight networks (TWNs) in [7] adopts ternary weights with $+1$, 0, and -1.

However, the defect is that existing low-precision methods need prohibitive retraining to regain the accuracy, which may take weeks or months even using high-performance servers. Moreover, the fine-tuning for complex DCNNs is complicated. Although [8] presents a simple quantization method around 5 bits on log-domain, the accuracy degradation is still obvious. Hence, a retraining-free, low-complexity, and high-accuracy DCNN inference is highly required.

In this paper, the problems are well solved by using non-uniform quantization. The key contributions are as follows:

- We propose an retraining-free and low-quantization scheme on the basis of [9] through exploiting the redundancy of original method and DCNN weights.
- We propose a novel DCNN accelerator with two-level systolic structure, which is multiplication-free in inference.
- We demonstrate the high accuracy and $5\times$ to $8\times$ compression rate of proposed low-precision quantization scheme on state-of-the-art DCNN models.

2. PRELIMINARIES

A deep convolutional neural network (DCNN) is built by iteratively stacking several types of functional layers: convolution layer (CONV Layer), pooling layer, and fully-connected layer (FC Layer). Here we briefly introduce the essential CONV layer. For an input feature map (ifmap) with a single channel, the CONV layer performs convolution of input feature map and a set of $K_w \times K_h$ filters. The output feature map (ofmap) of CONV layer can be computed as following equation:

$$\mathbf{O}_{x,y}^{conv} = \sum_{i=0}^{K_w-1} \sum_{j=0}^{K_h-1} \mathbf{w}_{i,j}^{conv} * \mathbf{x}_{x+i,y+j}^{conv} + b^{conv}, \quad (1)$$

where $\mathbf{w}^{conv} \in \mathbb{R}^{K_w \times K_h}$ denotes the filter weights of CONV layer and $\mathbf{x}^{conv} \in \mathbb{R}^{N_w \times N_h}$ is the ifmap. Note that the bias b^{conv} is optionally added to the result.

3. EFFICIENT NON-UNIFORM QUANTIZATION AND COMPRESSION SCHEME

3.1. Quantization

In general, DCNN parameters can be expressed by the tensor form $\mathbf{W} \in \mathbb{R}^{C \times W \times H}$, where C is the channels of this tensor. Similar to [9], the binary representation of DCNN weights is divided into N segments. The n-th segment, containing B_n bits, is the index of quantization value in the n-th codebook. Therefore, the i-th entry w_i of $\mathbf{W}$ can be quantized by following equation:

$$\hat{w}_i = \sum_{n=1}^{N} \phi_n \left[\mathrm{idx_{i,n}}\right], \tag{2}$$

where ϕ_n is the n-th codebook and $\mathrm{idx}_{i,n}$ denotes the n-th segment of $\hat{w}_i$. There are N codebooks in total.

Instead of directly quantizing the original weights into single binary sequence, the scheme in Eq. (2) implements quantization through looking up the codebooks. Each sequence stores N different indexes of codebooks. Hence, the length is L ($L = \sum_{n=1}^{N} B_n$) bits.

To realize efficient inference for DCNN, the codebook ϕ is constructed by the power of 2 and zero, which can be conveniently calculated by shift operation. Different from [9], the codebooks in this paper only include the negative power of 2 by removing all the positive power part. This is because the DCNNs parameters are near normal distributed within range $[-1, +1]$ after normalization. The positive power part of 2 is redundant in this case. Besides, we add an offset value β_n to each codebook to compensate the precision loss. Fig. 1 illustrates the distribution of codebook index of first FC layer in VGGNet-16. The lines depict the approximate distribution of index value. It indicates that the indexes of different codebooks vary in range. The offset helps to cover wider range of weights under low quantization bits compared to original scheme. The n-th codebook structure can be represented as follows:

$$\phi_n = \left[0, 2^{-1-\beta_n}, 2^{-2-\beta_n}, ..., 2^{-(2^{B_n}-1)-\beta_n}\right], \tag{3}$$

where there are total 2^{B_n} entries in each codebook.

The offset β_n in codebooks is adaptively selected to achieve optimal quantization precision. Mean squared error (MSE) is adopted to evaluate the fitness for a specific β_n. The quantization MSE is defined as:

$$Q_{\mathrm{MSE}} = \frac{1}{I} \sum_{i=0}^{I-1} ||\hat{w}_i - w_i||^2, \tag{4}$$

where I is the total number of weights in a layer.

Thus our goal is finding the optimal β_n combination to minimize Q_{MSE}:

$$\beta_n = \arg\min_{\beta_n} \frac{1}{I} \sum_{i=0}^{I-1} ||\hat{w}_i - w_i||^2, \tag{5}$$

where the optimal β_n can be obtained by the greedy search.

3.2. Compression

After the weights are quantized by proposed method in Section 3.1, some redundancy can be exploited to further compress the model. From Fig. 1, it can be seen that the index is not uniformly distributed. Hence, it is potentially to shrink the model size by some simple compression method.

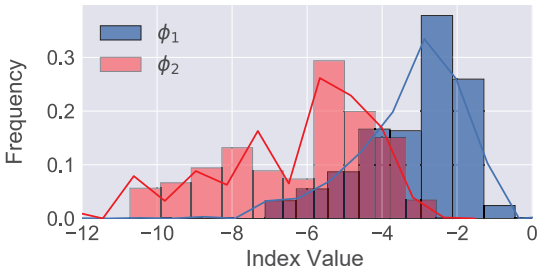

Fig. 1: Distribution of index idx for ϕ ($N = 2, B = 4$).

Since the occurrence probability of codebook index is fixed after the DCNN model is quantized, we take the advantage of lossless Huffman coding [10] that assigns variable-length code to encode the codebook index $\mathrm{idx}_{i,n}$.

3.3. Efficient Multiply-accumulate (MAC) Operation

One essential advantage of proposed non-uniform quantization method is low computation complexity. Instead of using bulky multiplier, the multiplication can be calculated by shift and addition. The MAC operation can be presented by following equation:

$$y = \hat{w}_i * x_i + b = \sum_{n=1}^{N} \phi_n \left[\mathrm{idx_{i,n}}\right] * x_i + b. \tag{6}$$

Since the elements of codebook are all the power of 2 or zero, the fixed-point MAC operation is replaced with shift and addition. One multiplication with shift and addition requires N shift and $N - 1$ addition when we have N quantization codebooks. Therefore, one shift-add MAC requires N shifts and N additions. N is chosen as 2 of this paper. In this case, Table 1 compares the normalized energy and area cost of a 16-bit MAC unit under commercial 90nm process. Shift-add MAC achieves significant reduction in both power and chip area.

Table 1: Comparison of normalized energy and area cost for a 16-bit MAC unit with 90nm technology.

	Power	Area
Shift-add MAC	1×	1×
Fixed-point MAC	7.3×	14.5×

4. DCNN ACCELERATOR DESIGN

In this section, the efficient hardware design of DCNN accelerator is introduced based on the proposed non-uniform quantization method.

4.1. System Overview

Fig. 2 shows the overall block diagram of proposed DCNN accelerator, consisting of a *two-level systolic array*, *on-chip buffer* and *Huffman coding module*. Note that the control logic is omitted in the figure. Inspired by [11], the vital part of proposed design is the two-level systolic array that will be described in next section.

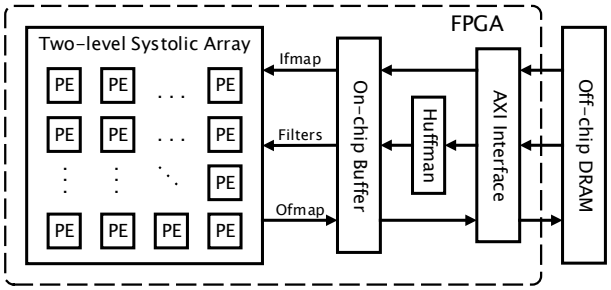

Fig. 2: Overall diagram of proposed DCNN accelerator.

The pre-trained model is preprocessed on CPU platform. After quantization and compression, the parameters of DCNN are stored on the off-chip DDR3 DRAM. Huffman coding module decodes compressed data from off-chip DRAM and then send it to on-chip buffer. The on-chip buffer is used as the cache of ifmaps, filters and ofmaps.

4.2. Two-level Systolic Design

The two-level systolic design is divided into two levels: the 2D systolic array and 1D systolic processing element (PE). The 2D systolic array contains 14×14 PEs. The two-level systolic design is regular and simple, which achieves good balance between data movement cost and computation. Proposed quantization strategy simplifies one MAC operation into two shift operation and two addition, enabling us to implement more basic computing cells in a single PE.

PE is the first-level systolic that implements 1D convolution with 5 basic cells. These cells are organized in the manner illustrated as Fig. 3, each containing a aforementioned

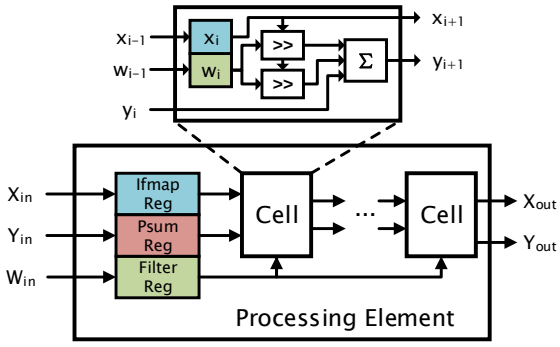

Fig. 3: Block diagram of processing element (PE).

shift-add MAC. According to [12], reducing high level memory access like off-chip buffer can reduce the power consumption caused by data movement. Hence, to reduce direct reading from off-chip DRAM, the filter weights are stored in the registers of PEs separately and stay still in cells while ifmap pixels and partial sums move systolically from left to right.

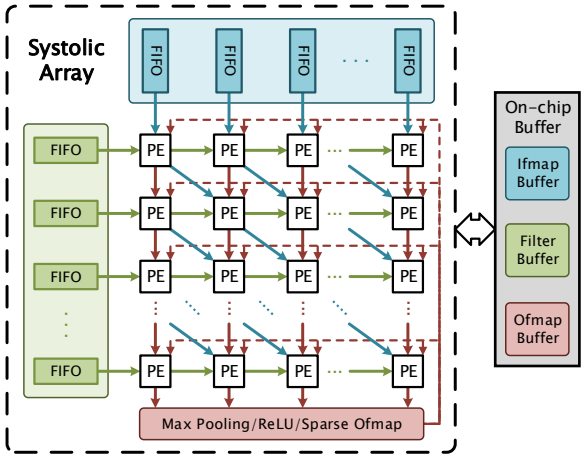

Fig. 4: Block diagram of systolic array.

Systolic Array is the second-level systolic that aims to reduce high-level data access. To maximize data reuse, PEs are organized in the grid form of Fig. 4, where adjacent PEs are connected by the illustrated pattern. The ifmap and filter FIFOs (First In First Out) fetch the input data from on-chip buffer. The FIFOs in the leftmost and uppermost broadcast filter weights and ifmap data to each PE in the same row or column, respectively. The intermediate partial sum is cached in the buffer on the bottom of the array. After the ofmap is calculated, max pooling and ReLU are optionally applied. The sparsity of ofmap is exploited by storing the ofmap in the sparse matrix form.

5. EXPERIMENT RESULTS

In this section, proposed non-uniform quantization scheme is first evaluated on popular DCNN models. Then performance

Table 2: Validation of various low-precision schemes on ImageNet.

Model	Method	Bit-width	Top-1/top-5 Accuracy	Degradation	Retraining
AlexNet [3]	Baseline	32	56.55%/79.09%	−/−	No
	Proposed	(**3**, **3**)	54.98%/77.89%	**−1.57%/ − 1.20%**	**No**
	LogNet [8]	5	−/74.6%	−/ − 3.70%	**No**
VGGNet-16 [2]	Baseline	32	71.59%/90.38%	−/−	No
	Proposed	(**3**, **3**)	69.36%/88.43%	**−2.23%/ − 1.95%**	**No**
	Fixed-point	16	68.01% / 87.89%	−3.58%/ − 2.49%	No
ResNet-18 [1]	Baseline	32	69.76%/89.08%	−/−	No
	Proposed	(**3**, **3**)	67.79%/87.91%	**−1.97%/ − 1.17%**	**No**
	ShiftCNN [9]	(4, 4)	64.24%/85.79%	−3.21%/ − 2.05%	No
	TWNs [7]	2	65.30%/86.20%	−2.56%/ − 1.80%	Yes

† Top-1/top-5 error are tested with single center crop. Degradation is taken from original papers.

analysis of hardware implementation is also shown.

5.1. Evaluation on ImageNet

Evaluation is conducted based ImageNet [4] validation dataset, which has 1000 object classes with 50,000 images. We use pre-trained DCNN models available from [13], including AlexNet [3], VGGNet-16 [2] and ResNet-18 [1]. Proposed quantization method is compared with other low-precision schemes as well as corresponding 32-bit full-precision baselines. Different combinations of codebook number N and index length B_n are tested and the offset β_n is selected adaptively by aforementioned rules. Separated offset β are applied to quantize each layer (CONV layer and FC layer). As a result, we don't observe much gain when $N > 2$ and $B > 4$. Hence, $N = 2, B_1 = B_2 = 3$ are selected.

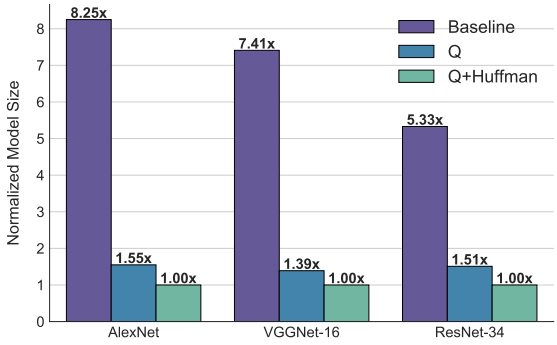

Fig. 5: Compression of model size with proposed quantization scheme (Q: Quantization).

Table 2 summarizes the detailed results. Compared with full-precision DCNN, acceptable degradation is introduced by our quantization scheme. Besides, this scheme is simple and easy to operate, which saves lots of time and computing resources without retraining. The other advantage of proposed quantization scheme is the dramatic reduction of intensive storage requirement. Fig. 5 shows the compression effect on floating-point models after adopting quantization and Huffman coding. As a result, $5\times$ to $8\times$ model compression is obtained.

5.2. Performance Analysis

The DCNN acceleration design combined with proposed quantization method is implemented on Xilinx Virtex-7 VC709 FPGA platform with Virtex-7 XC7VX690T chip. We compare the convolution computation of our accelerator with up-to-date designs in Table 3. VGG-16 is adopted in our test. Notably, proposed design consumes zero DSP modules, benefiting from the low bit-width shift-add MAC. The other costing slices is also apparently much less than [14] and [15].

Table 3: Performance comparison of various CNN acceleration designs on FPGA.

Design	[14]	[15]	This work
Platform	Zynq XC7Z045	Virtex-7 VX690t	Virtex-7 VX690t
Clock(MHz)	150	150	150
Quantization	16-bit fixed	16-bit fixed	(**3**, **3**)
LUT	186,251	≈ 300,000	**107995**
FF	127,653	≈ 300,000	**117795**
DSP	2240	2833	**0**
BRAM	1024	1248	1279
Throughput (GOP/s)	187.8	636.0	238.2

6. CONCLUSION

This paper presents a DCNN acceleration strategy with low complexity and promising performance. We first introduce a low-precision and retraining-free quantization scheme, which removes multiplication during DCNN inference. Then an efficient two-level systolic acceleration architecture is given based on proposed quantization scheme. Experiment results demonstrate the advantages of proposed design in terms of accuracy and throughput.

7. REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2016, pp. 770 – 778.

[2] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems(NIPS)*, 2012, pp. 1097–1105.

[4] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2009, pp. 248 – 255.

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations," *Advances in neural information processing systems(NIPS)*, pp. 1–9, 2015.

[6] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *arXiv preprint arXiv:1603.05279*, 2016.

[7] Fengfu Li and Bin Liu, "Ternary Weight Networks," *arXiv preprint arXiv:1605.04711*, 2016.

[8] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5900–5904.

[9] Denis A. Gudovskiy and Luca Rigazio, "Shiftcnn: Generalized low-precision architecture for inference of convolutional neural networks," *arXiv preprint arXiv:1706.02393*, 2017.

[10] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[11] H. T. Kung, "Why systolic architectures?," *IEEE Computer*, vol. 15, no. 1, pp. 37–46, 1982.

[12] Yu-Hsin Chen, Tushar Krishna, Emer, et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, pp. 127–138, 2017.

[13] Pytorch, "Torchvision models," `http://pytorch.org/docs/master/torchvision/models.html`.

[14] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, et al., "Going deeper with embedded FPGA platform for convolutional neural network," in *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 26–35.

[15] C. Zhang, Zhenman Fang, Peipei Zhou, Peichen Pan, and Jason Cong, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.