Real-time object detection towards high power efficiency

Jincheng Yu[1], Kaiyuan Guo[1], Yiming Hu[1], Xuefei Ning[1], Jiantao Qiu[1], Huizi Mao[1], Song Yao[2], Tianqi Tang[1], Boxun Li[1], Yu Wang[1], and Huazhong Yang[1]
[1]*Tsinghua University, Beijing, China* [2]*DeePhi Technology, Beijing, China*

Abstract—**In recent years, Convolutional Neural Network (CNN) has been widely applied in computer vision tasks and has achieved significant improvement in image object detection. The CNN methods consume more computation as well as storage, so GPU is introduced for real-time object detection. However, due to the high power consumption of GPU, it is difficult to adopt GPU in mobile applications like automatic driving. The previous work proposes some optimizing techniques to lower the power consumption of object detection on mobile GPU or FPGA. In the first Low-Power Image Recognition Challenge (LPIRC), our system achieved the best result with mAP/Energy on mobile GPU platforms. We further research the acceleration of detection algorithms and implement two more systems for real-time detection on FPGA with higher energy efficiency. In this paper, we will introduce the object detection algorithms and summarize the optimizing techniques in three of our previous energy efficient detection systems on different hardware platforms for object detection.**

I. Introduction

The visual recognition and classfication tasks are essential to computer vision. Thanks to the development of deep neural network (DNN) especially convolutional neural network (CNN), the image object classfication tasks have been improved greatly [1]. The great success in image classfication tasks introduces CNN in other computer vision tasks like object detection.

The object detection problem is a widely researched problem. The object detection task requires that the algorithm tells not only what objects are in the image but also the location of each object. The object detection is essential to a large variety of applications such as robotics, self-driving vehicles, and security systems. Recent work uses CNN to replace some of the components in tranditional object detection algorithms and enhances the accuracy from 33% to 58% in the VOC data set [2].

Though the accuracy of object detection methods is improved by CNN based methods, the computation complexity and memory consumption of CNN based algorithms also increase. A typical CNN, VGG-D [3], has 20M parameters. A single forward pass of VGG-D needs more than 30G operations [3]. It usually takes seconds to process a single image with the CNN models on CPU. GPU is introduced for real-time object detection. However, the high power consumption limits the application scene of GPU, and it is difficult to adopt GPU in moving scenes like automatic driving. Thus, the previous work proposes some optimizing techniques to lower the power consumption of object detection on mobile GPU or FPGA.

This work was supported by National Key R&D Program of China 2017YFA0207600, National Natural Science Foundation of China (No.61373026, 61622403, 61621091), and Joint fund of Equipment pre-Research and Ministry of Education (No. 6141A02022608).

There are three main factors that affect the performance of a CNN accelerating system [4]. One is the complexity of the CNN based algorithm, one is the peak performance of the accelerator, and the other one is the hardware utilization rate.

In the first Low-Power Image Recognition Challenge (LPIRC), our system achieved the best result with mAP/Energy [5]. This work adopts SVD to reduce the complexity of the algorithm and pipeline structure to improve the utilization of mobile GPU. Based on this work, we further research the high energy efficiency object detection algorithms on different hardware platforms. We use low-bit number format to reduce the problem complexity [6], introduce fast algorithm for convolution to promote the peak performance on FPGA platforms [7], and use memory optimizing techniques like cross layer scheduling to make full use of the hardware [7]. Furthermore, we develop an instruction set to support different object detection algorithms on the same hardware [4], tracking the upgrade of detection framworks [6], [7].

In this paper, we review three of our energy efficient detection systems [5]–[7], and summarize the optimization techniques on hardware with following contributions.

- We review the development of object detection algorithms and extract the basic components in CNN based detection algorithms.
- We review our previous work and summarize the optimization techniques on hardware for object detection.

The rest part of this paper is organized as follow. Section II introduces the background of CNN detection algorithms. Section III summarizes the optimizing methods for hardware. Section IV illustrates the performance of some real-time detection systems. Section V concludes this paper.

II. Object Detection Methods Based on CNN

In this section, we will introduce the background of hardware CNN object detection design. We will give a brief introduction of CNN, and analyze some of the state-of-the-art CNN based object detection algorithms.

A. Basic Operations of CNN

CNN achieves state-of-the-art performance on a wide range of computer vision tasks. To help to understand CNN, the basic operations in CNN are introduced in this section. A typical CNN consists of some layers running in sequence. The data between different layers are called *feature maps*. There are kinds of layers in CNN: convolution layer, non-linear layer, pooling layer and FC layer.

Convolution layers process 2-d convolution with trained parameters. Convolution layers are usually cascaded to extract high-level features of input.

Non-linear layers apply nonlinear activation function to each element in the output feature maps of convolution layers. Non-linear layers increase the fitting ability of CNN. Rectified Linear Unit (ReLU) is a widely used nonlinear activation function in CNN.

Pooling layers increase the receptive field of output element and reduce the computational complexity of CNN. A pooling layer outputs the max or average value of all elements in an area of the input feature maps.

FC layers fully connect all neurons in the input and output layers. The weights of a FC layer can be denoted in a matrix, and the outputs can be computed from matrix-vector multiplication. Because of the dense data transfer of weights matrix, the performance of a FC layer is usually restricted by the bandwidth. Some CNN models have abandoned FC layers for high performance like SSD.

B. CNN based Object Detection Algorithms

The traditional object detection methods usually consist of three basic steps. The first step is the region proposal step. In this step, an algorithm is chosen to generate the coarse-grained possible region of the objects. The region proposal algorithms are usually based on the priors of object characters such as color and texture features. The second step is the feature extraction and classification algorithms. In this step, each possible region proposed in the first step is resampled to the proper size and sent into a classifier to determine whether the proposed region contains a target object. In the traditional face detection algorithm, cascaded Haar filters are adopted as the classifier [8]. The last step is the fine-grained bonding box merging and regression step, which generates fine-grained location for each object generated from the second step to calibrate the coordinates of output bounding box.

As CNN has already achieved high performance in object classification tasks, it is natural to use CNN in the classification step of object detection, and this is the idea of original RCNN [2]. In RCNN, a complete CNN forward is computed for each proposed bonding box, which is very computation intensive. Fast-RCNN [9] reuses the feature extraction layers for all the proposal boxes, and runs the classification layers respectively for each proposal box. Furthermore, Faster-RCNN [10] uses a network, called region proposal network (RPN), to extra possible regions. The RPN can also reuse the feature extraction layers in Fast-RCNN.

However, all of these object detection methods based on RCNN need to calculate the classification layers respectively for each proposed network. The repetitive computation of the classification layers (usually FC layers) is still computationally intensive. Some work design one shot frameworks to accelerate object detection, like YOLO [11] and SSD [12]. These one shot frameworks detect possible object for each category directly from the output of CNN. YOLO uses a simple CNN with FC layers without any branches while SSD generates possible objects from intermediate featuremaps without FC layers.

The feature extraction layers in RCNN and one shot frameworks are usually inherited from pretrained classification networks like AlexNet [1] and VGG [3]. The network structures of the object detection methods are illustrated in Fig 1.

Generally, there are five components in these object detection algorithms, region proposal methods, ROI pooling of the proposal, FC layers and convolution layers. With the development of algorithms, the object detection frameworks become more and more simple with fewer components. The components used in each framework is illustrated in Table I. To accelerate detection algorithms in GPU platforms, our previous work [5] uses pipeline structure to make full use of the computation resources and adopts SVD to accelerate FC layers. If the hardware platforms can be customized like FPGA, some more techniques can be adopted to accelerate CNN. Our previous work uses low-bit number format to save computation resources and leverage bandwidth [6]. Recently, we introduce cross layer scheduling method to eliminate intermediate data transfer and use the Winograd algorithms to accelerate convolution computation [7]. These two FPGA designs are both instruction driven and can run different detection algorithms. Table I shows the optimization techniques and the corresponding previous work.

III. Optimization for Hardware

In this section, we will introduce the optimization techniques to accelerate CNN on hardware platforms.

A. Pipeline design

To accelerate the Fast-RCNN algorithm on mobile GPUs, Mao [5] designs a pipeline based detection system on mobile GPUs with CPU cores (Nvidia TK1 platforms). In this implementation, the Fast-RCNN consists of three stages, the proposal algorithm, the CNN inference, and the NMS process. The proposal algorithm and the NMS process run on the CPU cores, and the CNN model is distributed on the mobile GPU. The system runs these three stages in pipeline and achieved the best results with mAP/Energy in the first Low-Power Image Recognition Challenge (LPIRC). However, it is not necessary to detect objects from each frame in practice, since the images usually come in sequence with little difference in contiguous frames. Tracking techniques are introduced for higher speed. Wang [6] designs an embedded system with CNN detection and KCF tracking on FPGA. KCF tracking algorithm is 10x faster than the CNN detection model, so the tracking of 10

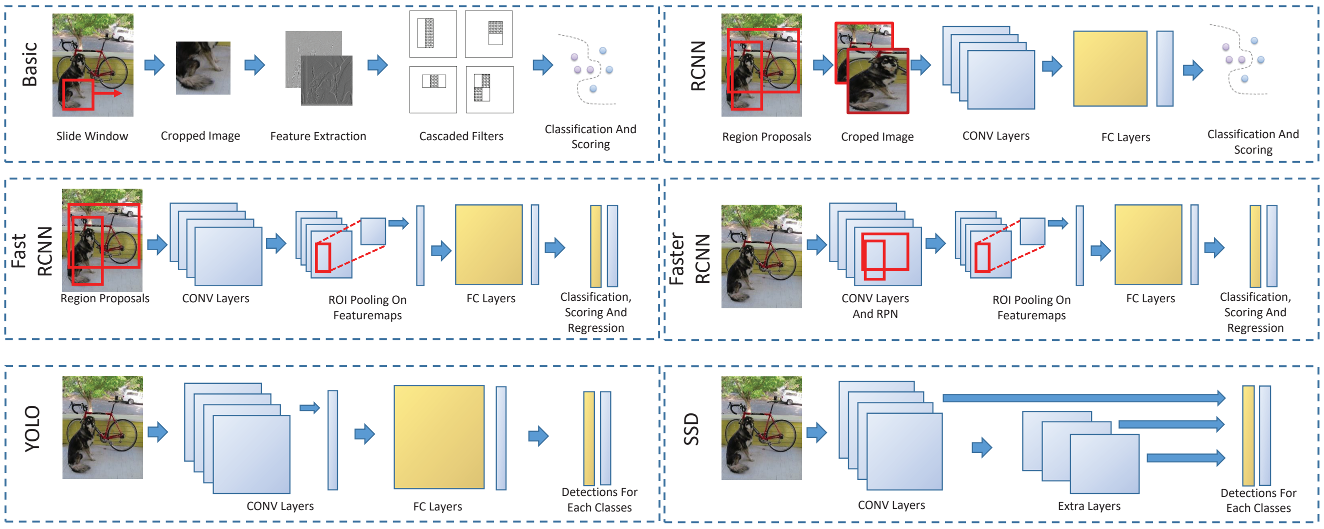

Fig. 1. Network structure of detection algorithms expanded from [5]

TABLE I
THE COMPONENTS USED IN DIFFERENT DETECTION FRAMWORKS AND THE COMPLEXITY AND ACCURACY ON VOC [13] OF EACH FRAMEWORK AND THE OPTIMIZATION TECHNIQUES AND THE CORRESPONDING PREVIOUS WORK.

	Year	Components				Complexity	Accuracy	Optimization				
		Proposal	ROI	FC	Conv	(GOP)	(mAP)	Pipeline	Fixed number	Cross layer	Winograd	SVD
Fast-RCNN (RCNN)	2014	✓	✓	✓	✓	-	70%	[5]				[5]
Faster-RCNN	2015		✓	✓	✓	300	73%	[6]	[6]			
YOLO	2016			✓	✓	63	63%	[7]	[7]	[7]	[7]	
SSD	2016				✓	52	74%	[7]	[7]	[7]	[7]	

frames can run in pipeline while detecting objects from one frame. The pipelined architecture can make full utilization of the hardware platforms with heterogeneous computation units. The pipeline structure of previous work is illustrated in Fig 2.

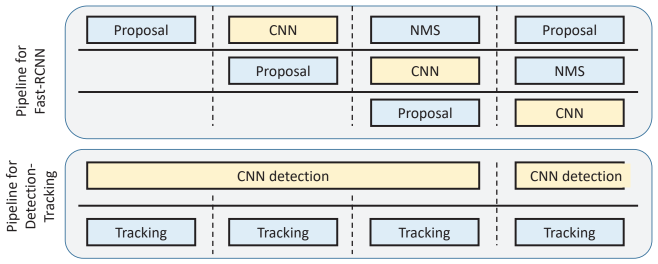

Fig. 2. The pipeline design of previous work on heterogeneous architecture, Fast-RCNN [5] and detection with tracking [6]

B. Data Quatization

Previous work shows that 8-bit quantization for weights and featuremaps is enough for classification tasks [14], [15]. In previous work [6], [7], we adopt the 8-bit quantization technique to object detection tasks.

The original network and the reconstructed network are trained in float-point representation. After translating the float-point system to 8-bit fixed-point system, the accuracy descends. So we use the fixed-point finetune method to improve the performance of fixed-point object detection method.

The **fixed-point finetune method** is based on the conventional method for CNN training, stochastic gradient descent method(SGD). There are two phase in SGD, feed forward and back propagation. In our fixed-point finetune method, we use fixed-point number in feed forward phase and keep float-point number in backpropagation phase. After updating all the weights in the back propagation step, we will analyze the dynamic range of the featuremaps and weights. The feed forward step of next iteration is processed in fixed-point with the updated quantization point position. Fig 3 shows the process of our fixed-point fine tune method.

C. SVD optimization

In the frameworks with FC layers like YOLO and RCNN, FC layers contribute to most of the weights [16], since each weight in FC layers is used only once. SVD can be adopted for accelerating FC layers also maintaining accuracy. The FC layer with n_{in} input elements and n_{out} output elements

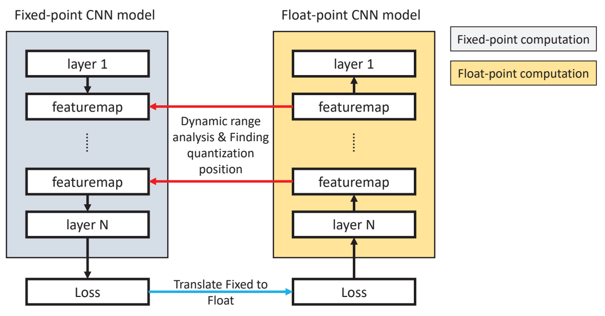

Fig. 3. The fixed-point fine tune method. Fixed-point forward and float-point backpropagation. The weights and data will be re-quantized after each backpropagation.

can be descriped as $f_{out} = W f_{in}$, where matrix W can be decomposed as $W \approx U_d \cdot S_d \cdot V_d = U_d' \cdot V_d'$, and d is the rank of U_d' and V_d'. The computation and storage complexity can be reduced from $(n_{in} \cdot n_{out})$ to $(n_{in} \cdot d + n_{out} \cdot d)$. Sinece the accuracy loss of the network can be ignored when d is much smaller than n_{in} and n_{out}, considerable reducetion of time and memory consumption can be achieved.

D. Fast algorithms for convolution

Convolution layers consume most of the computation in CNN. There are several methods to reduce computation complexity of 2-d convolution. Winograd's minimal filtering algorithm is a fast method to compute 2-d convolution especially when the convolution filters go small like 3×3 in CNN, and the Winograd algorithm uses a transformation to compute convolution [17]. For example, to calculate a convolution for which the image tile is 4×4 and the kernel tile is 3×3, the Winograd algorithm uses 16 multiplications. However, the standard algorithm of the same size uses 36 multiplications. The Winograd algorithm achieves a $2.25\times$ speedup with the same number of multiplications.

Winograd algorithm can be written in matrix form as:

$$Y = A^T \left[\left(G g G^T \right) \otimes \left(B^T d B \right) \right] A \tag{1}$$

The matrices are:

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix} A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix} \tag{2}$$

Where the $\otimes$ indicates elementwise multiplication, g is the 4×4 input tile, and d is the 3×3 convolution kernel. The multiplication with these transformation matrix elements can all be converted to shift operations (like$\times\frac{1}{2}$), which is very friendly to hardware. The larger the Winograd tile is, the acceleration rate of Winograd is higher. However, transformation matrices of a tile greater than 4×4 contain elements that cannot be simply converted to shift operations (like $\times\frac{1}{5}$ in a 5×5 Winograd tile). For this reason, some approximation should be done ($\frac{1}{5} \approx \frac{3}{16} = \frac{1}{8} + \frac{1}{16}$). The approximation will result in precision degradation in CNN, especially in detection tasks.

E. Bandwidth Optimization

CNN is both computation and storage consumptive. Some work focuses on optimizing the bandwidth as well as computation. Previous work [18], [19] considers the bandwidth and computation resources limitation and implements CNN acceleration system with respective hardware resources for each CNN layer so that the computation time and data transfer time can be overlapped with each other. In these implementations, the different layers run in the pipeline for different input images. The distribution of resources in different layers in the pipeline structure results in high latency when running a single image. Alwani [20] introduces a fuse-layer policy, enabling caching of intermediate data between the adjacent CNN layers, and reduces the transfer of intermediate data. However, Alwani implements the fuse-layer policy with pipeline structure. Recently, we modify the loop unrolling strategy and discard the pipeline structure so that all the computation units calculate the same layer at one time, reducing the latency of CNN models [7].

F. Flexibility

As described in Section II, the CNN models vary in different algorithms, and the network structures become more and more complicated. Some work designs instruction sets to run different CNN models on the same hardware. Cambricon [21] is an instruction set designed for neural networks. It translates all the calculation including convolutions to matrix multiplication and schedules CNN layer by layer. We also proposes an instruction-driven accelerator, but the massive data transfer between on-chip and off-chip memory limits the performance [4], [16]. Recently, we improve the instruction set for flexibility for implementing different CNN models with the fast algorithm Winograd algorithm and fuse-layer scheduling method [7].

IV. Performance Comparision

The performance of high power efficiency object detection systems is given and compared in this section.

In the first Low-Power Image Recognition Challenge (LPIRC), we implement the Fast-RCNN on the mobile GPU platform Nvidia TK1 [5]. Some other work adopts embedded FPGA to implement object detection [6], [22]. We also implement the object detection system with all the acceleration techniques summarized in this paper [7]. There are two popular data sets to evaluate the performance of detection algorithms, ILSVRC and VOC. ILSVRC is much more difficult than VOC as there are 200 classes in ILSVRC while only 20 in VOC. The mean average precision (mAP) is adopted as the final metric of detection and mAP indicates the mean accuracy of each class considering recall and precision.

For the fair comparison, the power of each FPGA implementation in the table is estimated from the XILINX toolchains. We only implement the convolution layers of YOLO in our recent design [7]. The performance [7] on YOLO will deteriorate when the FC layers are considered. The performance of YOLO is much lower than SSD, in both

TABLE II
COMPARISON OF DIFFENT HIGH ENERGY EFFICIENCY OBJECT SYSTEMS

	Platform	Framework[1]	Data Fomat	Dataset	Complexity (GOP)	Latency (ms)	DSP	Power (W)	Accuracy mAP	Power Efficiency GOP/s/W
[5]	NVDIA TK1	fast-rcnn	float	ILSVRC	-	540	-	9	26%	-
[6]	XILINX Zynq 7045	faster-rnn	8-bit fixed	VOC	59	680	576	3	66%	28.9
[22]	XILINX Zynq 7045	YOLO	32-bit fixed	VOC	14	744	800	1.6	63%	11.8
[7]	XILINX KU115	SSD	8-bit fixed	ILSVRC	55	105	1024	13	36%	40.3
				VOC	52	92			73%	43.5
		YOLO		VOC	30	65			62%	35.5

[1] The CNN models used in the same framework may be various, and will result in diffence in complexity and accuracy.

accuracy and power efficiency. Compared to the mobile GPU implementation of Fast-RCNN, the FPGA implementation on SSD is much more accurate and faster on the same data set. The data format also has a strong effect on the energy efficiency. With the help of fixed point fine-tune techniques, the 8-bit fixed implementation has comparable accuracy with 2x higher energy efficiency than 32-bit fixed implementation.

V. CONCLUTION

We summarize some of the optimization techniques on high energy efficiency design for CNN based object detection in our previous work. We use low bit data format and SVD decomposition to lower the complexity of algorithms. We use the Winograd algorithm to promote the peak performance of the hardware. The pipeline design in the heterogeneous systems and the memory optimization like cross layer strategy can improve the utilization of the hardware platforms. Furthermore, the development of detection algorithm framework is essential to high accuracy and low energy consumption. The accelerating systems should be able to support various network structures, so we introduce the instruction set to the CNN accelerators for flexibility.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

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

[4] K. Guo, S. Han, S. Yao, Y. Wang, Y. Xie, and H. Yang, "Software-hardware codesign for efficient neural network acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 18–25, 2017.

[5] H. Mao, S. Yao, T. Tang, B. Li, J. Yao, and Y. Wang, "Towards Real-Time Object Detection on Embedded Systems," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[6] J. Wang, K. Yan, K. Guo, J. Yu, L. Sui, S. Yao, S. Han, and Y. Wang, "Real-time pedestrian detection and tracking on customized hardware," in *2016 14th ACM/IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia)*, pp. 1–1, Oct. 2016.

[7] J. Yu, Y. Hu, X. Ning, J. Qiu, K. Guo, Y. Wang, and H. Yang, "Instruction driven cross-layer cnn accelerator with winograd transformation on fpga," in *International Conference on Field-Programmable Technology*, 2017.

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–511–I–518 vol.1, 2001.

[9] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, June 2017.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision ECCV 2016*, Lecture Notes in Computer Science, pp. 21–37, Springer, Cham, Oct. 2016.

[13] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan. 2015.

[14] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv:1510.00149 [cs]*, Oct. 2015. arXiv: 1510.00149.

[15] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[16] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going Deeper with Embedded FPGA Platform for Convolutional Neural Network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, pp. 26–35, ACM, 2016.

[17] S. Winograd, *Arithmetic complexity of computations*, vol. 33. Siam, 1980.

[18] Z. Liu, Y. Dou, J. Jiang, J. Xu, S. Li, Y. Zhou, and Y. Xu, "Throughput-Optimized FPGA Accelerator for Deep Convolutional Neural Networks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 10, pp. 17:1–17:23, July 2017.

[19] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*, pp. 1–9, IEEE, 2016.

[20] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer CNN accelerators," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12, Oct. 2016.

[21] S. Liu, Z. Du, J. Tao, D. Han, T. Luo, Y. Xie, Y. Chen, and T. Chen, "Cambricon: An instruction set architecture for neural networks," in *Proceedings of the 43rd International Symposium on Computer Architecture*, pp. 393–405, IEEE Press, 2016.

[22] R. Zhao, X. Niu, Y. Wu, W. Luk, and Q. Liu, "Optimizing CNN-Based Object Detection Algorithms on Embedded FPGA Platforms," in *Applied Reconfigurable Computing*, Lecture Notes in Computer Science, pp. 255–267, Springer, Cham, Apr. 2017.