Designing Efficient Shortcut Architecture for Improving the Accuracy of Fully Quantized Neural Networks Accelerator

Baoting Li, Longjun Liu*, Yanming Jin, Peng Gao, Hongbin Sun, Nanning Zheng
Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, China
*longjunliu@xjtu.edu.cn

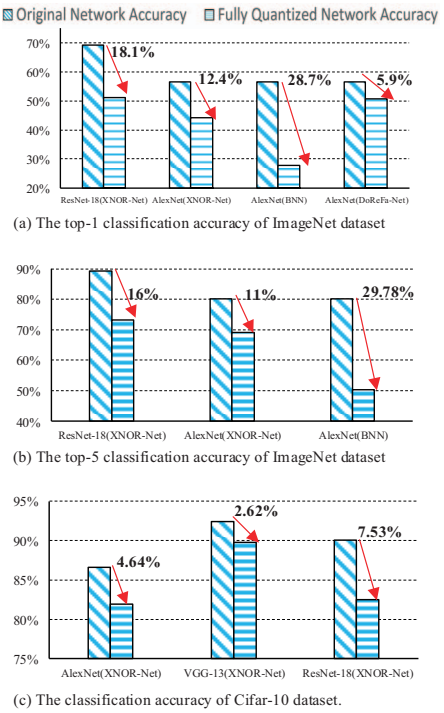
Abstract - Network quantization is an effective solution to compress Deep Neural Networks (DNN) that can be accelerated with custom circuit. However, existing quantization methods suffer from significant loss in accuracy. In this paper, we propose an efficient shortcut architecture to enhance the representational capability of DNN between different convolution layers. We further implement the shortcut hardware architecture to effectively improve the accuracy of fully quantized neural networks accelerator. The experimental results show that our shortcut architecture can obviously improve network accuracy while increasing very few hardware resources (0.11× and 0.17× for LUT and FF respectively) compared with the whole accelerator.

I Introduction

Deep neural networks have made dramatical improvements in various computer vision tasks, speech recognition, and natural language processing [1-4], etc. It also attracts an increasing number of studies to deploy state-of-the-art DNN models to embedded devices and custom circuits. However, as networks get deeper, DNN models suffer from over-parametrization and large amounts of redundancy, which often require considerable storage and computational power for hardware implementation [5]. In addition, irregular floating-point operations also cause additional calculation burden. As a result, training neurons and weights with low width or low precision is a crucial method for reducing the computational complexity of neural networks while greatly cutting down memory overhead for the specific hardware accelerator.

A lot of work has demonstrated that the neural network itself has strong robustness. Networks can also achieve satisfying accuracy without full-precision floating-point or even half-precision floating-point. Therefore, in order to tremendously simplify the neural network, many network quantization proposals have been proposed to quantize both weights and the activations of feature maps in neural network. Quantization methods are very friendly to deploy and implement various DNN models in resource-limited hardware platform.

However, the loss of network accuracy of low-bit DNN model is significant. Fully quantized network with both activations and weights are constrained to low bits may easily lead to the loss of information representation ability along with layer-by-layer convolution operations. Figure 1 summarizes the loss of accuracy for the state-of-the-art quantization methods such as BNN, XNOR-Net and DoReFa-net [6-8]. Figure 1 (a) and (b) are the top-1 and top-5 classification accuracy of ImageNet before and after full quantization respectively. Figure 1 (c) is the classification accuracy of Cifar-10 with XNOR-Net full quantization. According to the accuracy comparison of the full quantization operation in the experiments, we can see that full quantization methods will lead to a 5.9% to 28.7% decline in top-1 accuracy of ImageNet, which is unacceptable in performance sensitive applications.



(a) The top-1 classification accuracy of ImageNet dataset

(b) The top-5 classification accuracy of ImageNet dataset

(c) The classification accuracy of Cifar-10 dataset.

Fig. 1. The comparison of the accuracy variation of the fully quantized neural networks and original conventional neural networks in state-of-the-art DNN model and different datasets.

In order to improve the accuracy and efficiency of fully quantized DNN accelerator, we make the following contributions:

- We present a new shortcut strategy to compensate the information losing as the fully quantized operations, which fully considers the hardware implementation and will not incur too much hardware resources overhead.
- To save memory resources and improve shortcut efficiency, we further propose a Max/Avg pooling operations to transmit the feature information between different convolution layers.
- We implement the strategy with Verilog HDL to

978-1-7281-4123-7/20/$31.00 ©2020 IEEE

comprehensively evaluate our design. Our hardware module can be easily integrated in the current hardware accelerators for improving the accuracy of fully quantized neural networks.

- The experimental results show that the range of accuracy improvement of fully quantized networks is up to 0.81% to 18.18% while increasing very few hardware resources (0.11× and 0.17× for LUT and FF) compared with the ordinary accelerator.

The rest of the paper is organized as follows: Section II presents the related work. Section III proposes the detailed algorithm and hardware design of our shortcut for fully quantized DNN. Section IV shows experiment methodology and evaluation results. Section V concludes the paper.

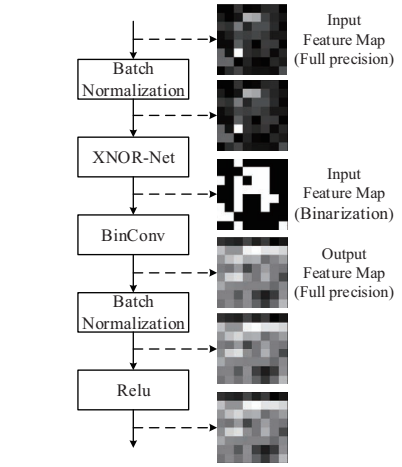
II. Related Work

Weights quantized networks. BWN [7] trains neural networks with weights constrained to +1 or -1, and each layer shares a scaling factor. In contrast, TWN [9] quantizes weights to 1, 0 and -1, which improves the representational capability of quantized network without increasing the computational complexity. INQ [10] converts full-precision network model into a low-precision version whose weights are constrained to be either powers of two or zero. All the quantization methods try to trade-off between expected bit-width (model size) and accuracy (model performance).

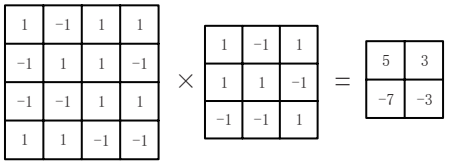
Weights and activations fully quantized networks. In order to further reduce model storage consumption and computational complexity, a great deal of existing work also quantize the activations of feature maps except for quantizing weights. XNOR-net [7] reduces information loss by introducing quantization coefficients and adjusting the internal order of CNN feature extraction module. As the binarized outputs will reduce accuracy, DoReFa-Net [8] uses *k*-bits to quantize weights and activations. The fixed-point design [11] leverages back-propagation-based retraining and adds a coefficient to each quantized value to improve the accuracy of fully quantized networks.

Current DNN accelerators. One of the key goals for quantized neural networks is to deploy DNN into resource limited hardware systems. Currently, most of the existing DNN accelerators (e.g. [12-15]) support the acceleration of traditional networks such as AlexNet [2] and VGG [16], but lack the dedicated acceleration for fully quantized networks, which may lead to unnecessary computing and storage overhead. The emerging bit-level accelerators with variable data bit width (e.g. [17] [18]) can implement the quantized neural networks. However, the accuracy is not satisfied due to the loss of feature information caused by the fully quantized network itself.

In brief, this paper distinguished itself with other work is that we fully study the quantized methods for neural networks and focus on how to improve the accuracy of fully quantized neural networks while avoiding incur additional hardware resources overhead. According to software-hardware co-design method, we propose an efficient shortcut architecture to transfer the feature information of weights and activations from the former layer to the next convolution layer. Our shortcut design can be easily integrated into current accelerator without redesign, nor increase the additional data bandwidth of original accelerator. To our best knowledge, this is the first work to improve the accuracy and efficiency for fully quantized neural network accelerators with efficient shortcut methods.



(a) The convolution block of XNOR-Net.



(b) The results of bitcount calculation is not binarized.

Fig. 2. The feature information loss of full quantization effect.

III. Efficient Shortcut for the Fully Quantized DNN

In this section, we introduce the efficient shortcut strategy to improve the accuracy of fully quantized DNN, and the design of key hardware units with efficient shortcut.

A. Design of Efficient Shortcut

The information loss is obvious if there is no shortcut operation, as shown in Figure 2 (a). Taking XNOR-Net as an example, the input feature map is performed bitcount convolution with the binarized weights. The output feature map is not binarized, which is obtained by binarized input feature map and binarized kernel after bitcount convolution but it is full precision, as shown in Figure 2 (b). However, the output feature map will be binarized before the next convolution calculation. Therefore, the network with both quantized activations and weights will lose part of the feature information and affect the accuracy of the network.

The shortcut method proposed by ResNet [19] is very suitable for transmitting feature information between different convolution layers without increasing the computational complexity of the network. Inspired by the

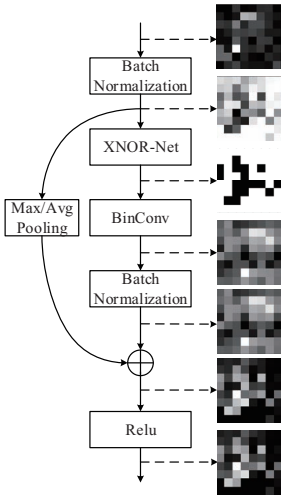


Fig. 3. The convolution block of XNOR-Net with Max/Avg pooling shortcut.

advantages of shortcut in ResNet, we propose to leverage the efficient Max/Avg pooling shortcut to transfer network feature information and improve the accuracy of fully quantized network while be benefit to hardware design and resource saving, as shown in Figure 3.

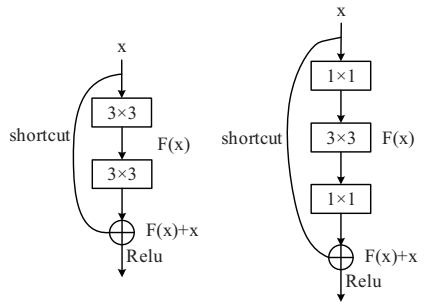
Previous experiments show that the residual blocks of original ResNet need more than two layers to enhance the accuracy of non-quantized DNN model. However, for the fully quantized neural networks, we find that the effect of quantization operation is just like a special regularization, so it is reasonable to add a shortcut branch to each convolution layer and transmit the feature map information from the previous layer to the next convolution layer. We formulate the operation with two equations: Eq. (1) is the expression of shortcut operation in ResNet, and Eq. (2) is the expression of a convolution operations with the proposed efficient Max/Avg pooling shortcut.

$$R(x) = F(x) + x \tag{1}$$

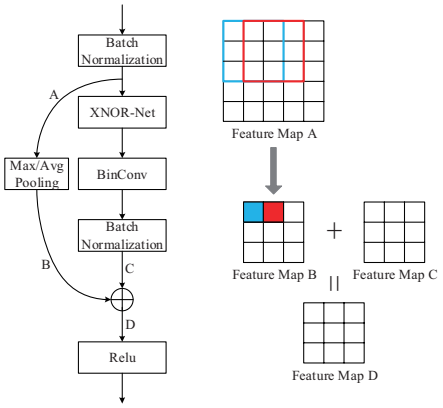
$$H(x) = S(x) + P(x) \tag{2}$$

In Eq. (1), ***R(x)*** is the output of ResNet block, ***F(x)*** is the output of input feature maps calculated by multi-layer convolution branch in block, and ***x*** is the input feature maps of each block. In Eq.(2), ***H(x)*** is the output of the improved convolution layer proposed in this paper, ***S(x)*** is the output of the convolution branch of each layer, and ***P(x)*** is the output of the maximum pooling shortcut or average pooling shortcut branch with the same kernel size and stride.

Figure 4 (a) further compares the two kinds of ResNet block with shortcut operations. It can be seen that the original shortcut of ResNet is to add the input and output feature maps of block directly, which may result in additional data bandwidth requirements for the hardware implements of DNN accelerator. Therefore, we improve the shortcut operation as shown in Figure 4 (b). The direct accumulation of input and output feature maps is changed to the maximum pooling or average pooling operations of the



(a) Two kinds of ResNet block with shortcut.



(b) The calculation process of Max/Avg pooling shortcut.

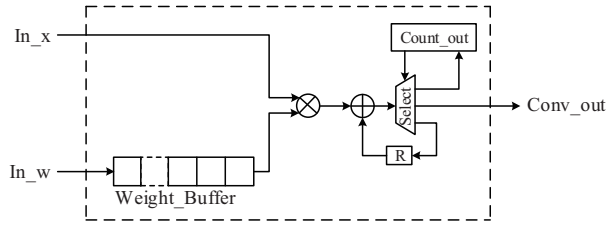
Fig. 4. Improvement of ResNet shortcut by efficient Max/Avg pooling shortcut.

input feature maps, which is synchronized with the convolution and then corresponds to the accumulation of the output feature maps. As a results, the feature map information is transmitted to the next fully quantized convolution layer. Figure 4 (b) shows the convolution with Max/Avg pooling shortcut in fully quantized DNN (XNOR-Net): Assuming that the convolution kernel size is 3×3. ***A*** is the full-precision input feature map. When bitcount convolution (BinConv) is performed, the maximum pooling or average pooling is also calculated with the same kernel size and stride on our shortcut branch. After that, a full-precision output feature map ***B*** is obtained after pooling operation, and a full-precision output feature map ***C*** is obtained after bitcount convolution. Then, by accumulating ***B*** and ***C***, the final full-precision output feature map ***D*** is obtained, which has obtained the information compensation from the previous layer.

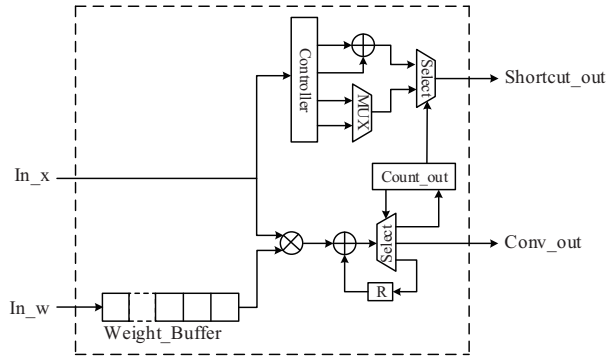
B. Implementation of Convolution Computing Module (CCM) with Improved Shortcut Architecture

Our Max/Avg pooling shortcut will not increase the data bandwidth of the hardware accelerator. Figure 5 shows the detailed implement of our hardware design.

Figure 5 (a) is the traditional PE unit architecture of the neural network accelerator. By changing the data bit width of the input neuron ***x*** and the weight ***w***, the accelerator can be used to calculate the fully quantized neural network.



(a) The traditional PE computing unit without shortcut



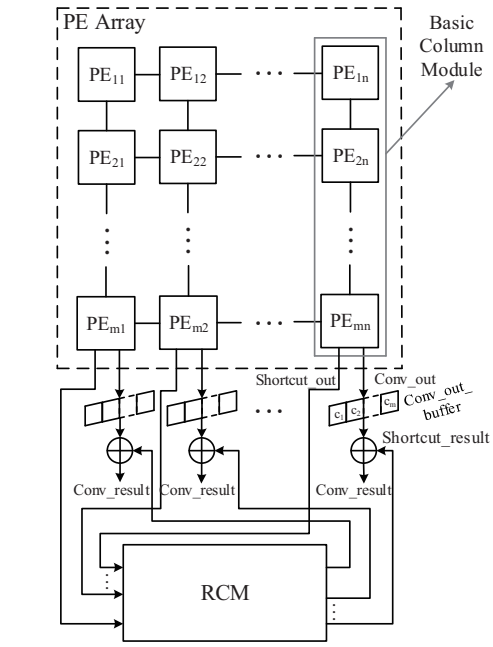
(b) The PE computing unit architecture with the efficient Max/Avg pooling shortcut structure

Fig. 5. Redesign of PE computing unit with shortcut unit.

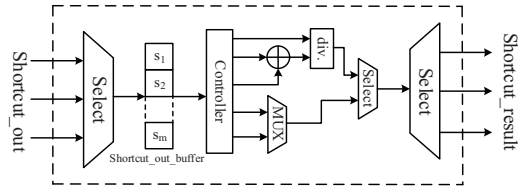
Figure 5 (b) is a PE unit architecture of our Max/Avg pooling shortcut architecture. The maximum pooling or average pooling shortcut can be selected according to the neural network configuration. When the input neuron ***x*** and weight ***w*** are sent to the multiplier, neuron ***x*** is also sent to Max/Avg pooling shortcut module to calculate the pooling result with the same convolution kernel size and stride. The final output of shortcut (*Shortcut_out*) can be obtained with synchronized clock of the convolution output (*Conv_out*).

We further implement the shortcut architecture in a whole Convolution Computing Module (CCM), as shown in Figure 6 (a). Each column of PE arrays is defined as Basic Column Module (BCM), which includes ***m*** PE units. The corresponding *Conv_out_buffer* is used to cache the convolution outputs of the BCM. The adder for each PE column units is used to accumulate the convolution outputs of different PE units under the same convolution window.

The Max/Avg pooling shortcut outputs of each BCM are transmitted to the Result Calculation Module (RCM), as shown in Figure 6 (b). Noted that if the computing parallelism pattern of the PE unit is based on a convolution window, the result of the comparator in the PE unit is the final maximum pooling shortcut result. The average value of the accumulator in the PE unit is the final average pooling shortcut result (the outputs of *Shortcut_out_buffer* are transmitted by the controller to the divider). If the computing parallelism pattern of the PE unit is based on a row or a column of a convolution window, the results of the comparator in the PE unit need to be compared with the outputs of corresponding multiple PE units in the same convolution window for obtaining the final maximum



(a) The CCM in a general neural network accelerator.



(b) The architecture of RCM.

Fig. 6. Redesign of Convolution Computing Module.

pooling shortcut result (The outputs of *Shortcut_out_buffer* are transmitted by the controller to the comparator to compare different rows in the same convolution window). The final average pooling shortcut result is the cumulative and recalculated average of the results of the corresponding accumulator of PE units in the same convolution window (the outputs of *Shortcut_out_buffer* are accumulated for different rows in the same convolution window by the accumulator, and then calculated the average value by divider). Finally, according to the addition of pooling shortcut results and corresponding convolution outputs, the information transfer is completed, which compensates the loss of accuracy caused by quantization.

In brief, according to our hardware design, the input neurons are quantized in the PE unit. When PE calculates the partial convolution operation, the shortcut circuit calculates the partial pooling operation. Then, the partial convolution output of PE unit and the partial pooling shortcut output are cached in *Conv_out_buffer* and *Shortcut_out_buffer* respectively. At last, the final convolution output and the final pooling shortcut output are calculated in each BCM, and the final output result is obtained by synchronously adding up the convolution outputs and pooling outputs.

TABLE I
The Effect of Max/Avg Pooling Shortcut on the Accuracy of Different Fully Quantized Networks (Using XNOR-Net as the Quantitative Method)

Neural Networks	Accuracy (%)				
	Original	XNOR-Net	XNOR-Net with Avg pooling shortcut	XNOR-Net with Max pooling shortcut	Improvement
VGG-13	92.40	89.78	90.03	**90.59**	0.81
AlexNet	86.55	81.91	81.94	**84.06**	2.15
ResNet-18	90.04	82.51	84.05	**84.86**	2.35

TABLE II
The Results of VGG with Different Depths

Neural Networks	Accuracy (%)				
	Original	XNOR-Net	XNOR-Net with Avg pooling shortcut	XNOR-Net with Max pooling shortcut	Improvement
VGG-13	92.40	89.78	90.03	**90.59**	0.81
VGG-16	92.16	85.42	90.07	**90.20**	4.78
VGG-19	92.04	71.84	89.20	**90.02**	18.18

TABLE III
The Results of ResNet with Different Depths

Neural Networks	Accuracy (%)				
	Original	XNOR-Net	XNOR-Net with Avg pooling shortcut	XNOR-Net with Max pooling shortcut	Improvement
ResNet-18	90.04	82.51	84.05	**84.86**	2.35
ResNet-34	90.80	81.97	**87.06**	83.71	5.09

IV. Experimental Methodology and Evaluation Results

In this section, we evaluate our shortcut algorithm for fully quantized network by analyzing its accuracy and resource utilization in hardware. We first perform image classification on the Cifar-10 by integrating the shortcut architecture on the target neural network model. We further implement the fully quantized neural networks accelerator with efficient shortcut architecture by Verilog HDL and evaluate it on FPGA.

A. Image Classification with Fully Quantized Network

Taking XNOR-Net as an example, it quantizes both activations and weights as +1 and -1. We have implemented full quantization using XNOR-Net method with various neural network models to evaluate the image classification accuracy on Cifar-10.

Table 1 lists the experimental accuracy of various networks, including the original network, the fully quantized network, the fully quantized network with average pooling shortcut, and the fully quantized network with maximum pooling shortcut.

The Max/Avg pooling shortcut is used to compensate for the information loss caused by the quantization of feature results, it can be seen that the proposed method is very effective to improve the accuracy of the fully quantized network, which can lead to a 0.81% to 18.18% increase in accuracy. Among them, the effect of maximum pooling shortcut is better than that of average pooling shortcut.

Furthermore, we compare the fully quantized VGG and ResNet with different depths, as shown in Table 2 and Table 3. The maximum pooling shortcut is better for ResNet-18, while the average pooling shortcut is better for ResNet-34. This is because different pooling methods extract different feature information. From the experimental results, it can be seen that the deeper the fully quantized network is, the higher the improving accuracy of the proposed Max/Avg pooling shortcut is. This is because the deeper fully quantized network layers are, the more part of the feature information will be lost with layer-by-layer convolution, which has cumulative impact on the network accuracy. Increasing the pooling shortcut to transmit feature information will significantly improve the accuracy of the fully quantized networks.

B. Resource Analysis for Hardware Platform

We further use Verilog HDL to implement a 2-bit fully quantized neural network accelerator with our shortcut architecture and a 2-bit accelerator without our shortcut architecture. Only the CCM is different in the two designs. Figure 7 shows the system architecture design of the accelerator. We deployed and synthesize the two designs on Zynq UltraScale+ ZCU102 Evaluation Board

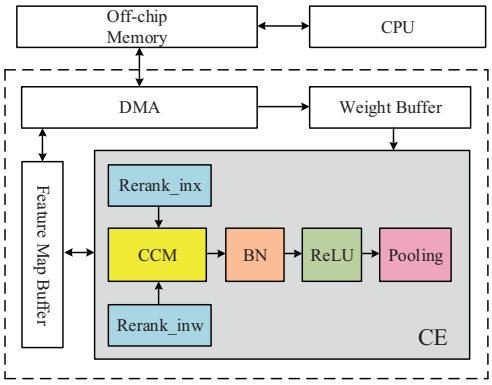


Fig. 7. System Architecture of Designed Accelerator.

TABLE IV
Hardware Resource of Different CE (Computing Engine)

Name	LUT	FF	BRAM	DSP
Ordinary architecture	16170 (5.90%)	6615 (1.21%)	72.50 (7.95%)	2 (0.08%)
With our shortcut architecture	17971 (6.56%)	7779 (1.42%)	72.50 (7.95%)	2 (0.08%)

(xczu9eg-ffvb1156-2-e) with Vivado 2018.3.

Table 4 shows the comparison of the hardware resources cost of the key Computing Engine (CE), which is used to calculate convolution, batch normalization, activation and pooling. It can be seen that the improved shortcut architecture does not increase too much hardware resource consumption (only increases by 0.11× and 0.17× for LUT and FF respectively).

V. Conclusions

Low-precision neural networks are promising solutions to reducing computation and storage resources for large-scale DNNs. Fully quantized networks that quantize both activations and weights may lose critical feature information along with layer-by-layer convolution operations, which will result in the accuracy degradation of the neural network. In this work, we propose a shortcut architecture inspired by the ResNet. Our shortcut architecture can be easily integrated in current hardware accelerators without additional data bandwidth and storage consumption, and can obtain the range of accuracy improvement of fully quantized networks up to 0.81% to 18.18%. Compared with the ordinary CE architecture, the hardware resource consumption of the CE with our shortcut only increases by 0.11× and 0.17× for LUT and FF respectively.

Acknowledgements

This research was supported by National Natural Science Foundation of China (No. 61722406, 61751401, 61602368), and by the Fundamental Research Funds for the Central Universities.

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, Vol. 61, pp. 85-117, 2015.
[2] Krizhevsky, Alex, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *NIPS Curran Associates Inc*, Vol. 25, 2012.
[3] Graves, Alex, and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, Vol. 18.5-6, pp. 602-610, 2005.
[4] Collobert R, Weston J, Bottou L, et al., "Natural Language Processing (Almost) from Scratch," *Journal of Machine Learning Research*, 2011.
[5] Chung, Jaeyong, T. Shin, and Y. Kang, "INsight: A Neuromorphic Computing System for Evaluation of Large Neural Networks," *Computer Science*, 2015.
[6] Courbariaux M, Hubara I, Soudry D, et al., "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *arXiv: Learning*, 2016.
[7] Rastegari M, Ordonez V, Redmon J, et al., "XNOR-Net: Imagenet Classification Using Binary Convolutional Neural Networks," *European Conference on Computer Vision*, pp. 525-542, 2016.
[8] Zhou S, Ni Z, Zhou X, et al., "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients," *arXiv: Neural and Evolutionary Computing*, 2016.
[9] Li, Fengfu, B. Zhang, and B. Liu, "Ternary Weight Networks," *arXiv: Computer Vision and Pattern Recognition*, 2016.
[10] Zhou, Aojun, et al., "Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights," *arXiv: Computer Vision and Pattern Recognition*, 2017.
[11] Hwang, Kyuyeon, and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and -1," *Signal Processing Systems (SiPS)*, 2014.
[12] Chen, Yu Hsin, et al., "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, Vol. 52.1, pp. 127-138, 2016.
[13] Han S, Liu X, Mao H, et al., "EIE: Efficient Inference Engine on Compressed Deep Neural Network," *Acm Sigarch Computer Architecture News*, Vol. 44(3), pp. 243-254, 2016.
[14] Zhang S, Du Z, Zhang L, et al., "Cambricon-X: An accelerator for sparse neural networks," *international symposium on microarchitecture*, pp. 1-12, 2016.
[15] Lee E H, Miyashita D, Chai E, et al., "LogNet: Energy-efficient neural networks using logarithmic computation," *international conference on acoustics, speech, and signal processing*, pp. 5900-5904, 2017.
[16] Simonyan, Karen, and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Computer Science*, 2014.
[17] Sharma H, Park J, Suda N, et al., "Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks," *International Symposium on Computer Architecture*, pp. 764-775, 2018.
[18] Sharify S, Lascorz A D, Siu K, et al., "Loom: Exploiting Weight and Activation Precisions to Accelerate Convolutional Neural Networks," *Design Automation Conference*, 2018.
[19] He K, Zhang X, Ren S, et al., "Deep Residual Learning for Image Recognition," *Computer Vision and Pattern Recognition*, pp. 770-778, 2015.