Journal of Electronic Science and Technology 21 (2023) 100204

Contents lists available at ScienceDirect

Journal of Electronic Science and Technology

journal homepage: www.keaipublishing.com/en/journals/journal-of-electronic-science-and-technology/

FPGA-based acceleration for binary neural networks in edge computing

Jin-Yu Zhan [a], An-Tai Yu [a], Wei Jiang [a,*], Yong-Jia Yang [a], Xiao-Na Xie [b], Zheng-Wei Chang [c], Jun-Huan Yang [d]

[a] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, 610054, China
[b] School of Automation, Chengdu University of Information Technology, Chengdu, 610225, China
[c] State Grid Sichuan Electric Power Research Institute, Chengdu, 610095, China
[d] Department of Information Sciences and Technology, George Mason University, Fairfax, VA22030, USA

ARTICLE INFO

Publishing editor: Xuan Xie

Keywords:
Accelerator
Binarization
Field-programmable gate array (FPGA)
Neural networks
Quantification

ABSTRACT

As a core component in intelligent edge computing, deep neural networks (DNNs) will increasingly play a critically important role in addressing the intelligence-related issues in the industry domain, like smart factories and autonomous driving. Due to the requirement for a large amount of storage space and computing resources, DNNs are unfavorable for resource-constrained edge computing devices, especially for mobile terminals with scarce energy supply. Binarization of DNN has become a promising technology to achieve a high performance with low resource consumption in edge computing. Field-programmable gate array (FPGA)-based acceleration can further improve the computation efficiency to several times higher compared with the central processing unit (CPU) and graphics processing unit (GPU). This paper gives a brief overview of binary neural networks (BNNs) and the corresponding hardware accelerator designs on edge computing environments, and analyzes some significant studies in detail. The performances of some methods are evaluated through the experiment results, and the latest binarization technologies and hardware acceleration methods are tracked. We first give the background of designing BNNs and present the typical types of BNNs. The FPGA implementation technologies of BNNs are then reviewed. Detailed comparison with experimental evaluation on typical BNNs and their FPGA implementation is further conducted. Finally, certain interesting directions are also illustrated as future work.

1. Introduction

With the increasing development of 5G and intelligent technologies, the number of mobile terminals and Internet of things (IoT) devices is exploding in human's life, making edge computing as a key topic in the research community. With the emergence of edge computing, data processing can be completed at edge nodes instead of concentrating on cloud servers. With the development of technologies and the surge of data (e.g. intelligent driving), it is increasingly difficult for the cloud to meet the timeliness of data processing. Edge computing greatly reduces the delay of data processing and transmission, and reduces the cost and energy consumption. With the continuous development of deep neural networks (DNNs), the combination of edge computing and DNNs has

* Corresponding author.
E-mail addresses: zhanjy@uestc.edu.cn (J.-Y. Zhan), 202021090211@std.uestc.edu.cn (A.-T. Yu), weijiang@uestc.edu.cn (W. Jiang), yangyongjia@std.uestc.edu.cn (Y.-J. Yang), xxn1213@cuit.edu.cn (X.-N. Xie), changzw@ustc.edu.cn (Z.-W. Chang), jyang71@gmu.edu (J.-H. Yang).

https://doi.org/10.1016/j.jnlest.2023.100204
Received 5 January 2023; Received in revised form 15 May 2023; Accepted 26 May 2023
Available online 9 June 2023

become more and more popular. DNNs have a strong learning ability and usually achieve a satisfactory performance. This is due to their deep structures with multiple layers and millions of parameters. For example, the Visual Geometry Group network 16 (VGG16) contains about 140 million 32-bit floating point parameters, and can achieve the image classification task with the accuracy of 92.7% on the ImageNet dataset, which is the top-5 test accuracy. VGG16 needs to occupy more than 500 megabytes (MB) of storage space and perform 1.6×10^{10} floating point operations. Although some commonly used methods improved the DNN accuracy (such as increasing the number of layers and using the residual network), the DNN complexity also increased. This fact makes DNNs rely heavily on high-performance hardware, such as the graphics processing unit (GPU), and it takes more time to calculate. While in reality, usually only the devices (e.g., mobile phones and embedded devices) with limited computational resources are available. Embedded devices based on FPGAs usually only have a few thousands of computing units, far from dealing with millions of floating point operations in the common deep models. Although there is a lot of special hardware for deep learning, which provides effective acceleration for convolution calculations, the heavy calculation and storage costs still inevitably limit the application of DNNs in practice. Therefore, researchers explore more efficient DNNs, like binary neural networks (BNNs) [1] for resource-limited edge computing systems.

A typical deployment of edge computing is illustrated in Fig. 1, where DNN inference is performed on edge nodes in order to meet the low latency.

DNN training, which requires a lot of computing resources, is carried out in the cloud. The edge node can download the trained neural network models from the cloud servers, and use the cloud to expand storage space and computing power. This will undoubtedly result in a huge challenge for edge computing devices with limited resources. Binarization converts activations and weights into {+1, −1} through specific methods, i.e., reducing the 32-bit floating point number to 2 bits or even 1 bit. Thus it can greatly compress the model and bring huge acceleration potential. The calculation operations of floating point number convolution include a lot of multiplication, addition, and subtraction operations, which are not friendly to resource-constrained devices. After the weight is quantified to {+1, −1}, all multiplication operations can be omitted [1,2], and the accuracy of DNN is only slightly dropped. When both the activation and the weight are quantified to {+1, −1}, the XNOR-popcount operation can be used to completely replace the multiply-accumulate operation [3], leading to a greatly decrease in hardware consumption. Although weights and activations binarization will result in a small decrease in accuracy, the accuracy can be guaranteed within an acceptable range and even close to the accuracy of the full-precision network by careful designs [4–6].

Although BNN greatly reduces the size of the model, it still has huge redundancies due to its binarization characteristics. By trimming these redundancies [7–10], the hardware can achieve a higher performance with lower power consumption. This is extremely important for deploying DNNs on edge computing devices. Recent studies in Ref. [10] and Ref. [11] showed that it was feasible to quantify the element value to {0, 1}, and the performance is even better than {+1, −1}. Quantification with 0/1 also brings sparseness to BNN, with the reduction of data access by 58.8% [10]. In addition, BNN-based research can also help us explore the interpretability of DNNs. Someone has previously verified the robustness of neural networks. But full-precision verifiers face scalability challenges and cannot provide a correctness guarantee due to floating point errors. Jia et al. [12] proposed the efficient exact verification (EEV) system, which used BNN to accurately verify the neural network and consumed less time. Binarizing a specific layer to explore its impact on accuracy can also help us explain the neural network.

In this article, we give a comprehensive review of BNNs and the corresponding hardware accelerator design in edge computing environments. The work of binarization and the methods of acceleration are compared, with respect to their advantages and disadvantages. The performances of BNNs and their FPGA implementations are also evaluated through experiments. Finally, we highlight several future research directions in this field.

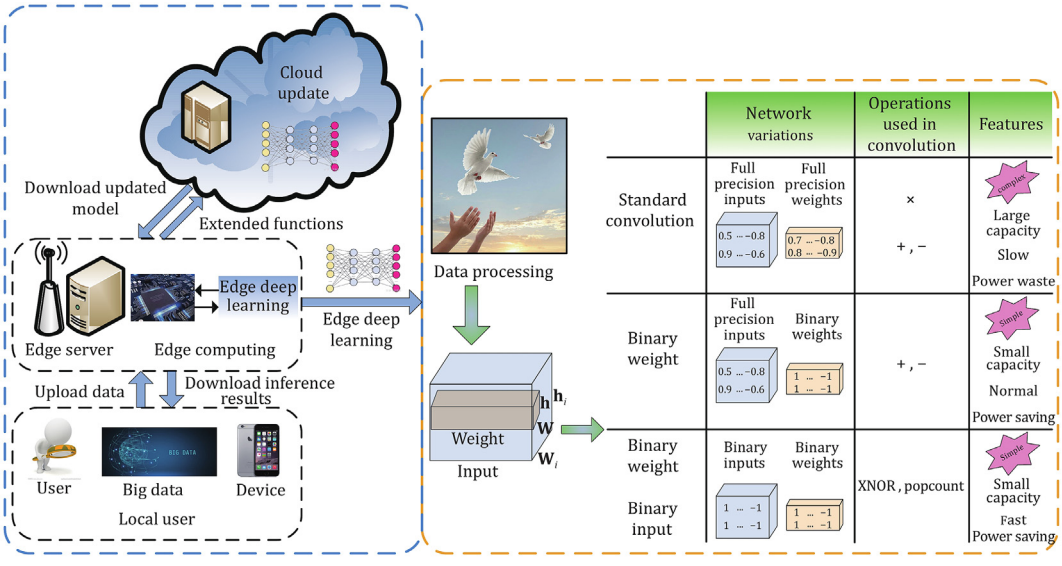

Fig. 1. Overview of binary quantitative neural networks in edge computing environments.

2. Preliminaries

2.1. DNN

DNNs are a branch of machine learning, which emphasize learning from continuous layers. After the input data have been learned through several layers, the predicted result will be generated. The "deep" in DNN means that there are many layers in the model. More layers will lead to a stronger learning ability for DNNs. Most DNNs usually contain dozens or even hundreds of continuous presentation layers, which are automatically learned from training data.

Due to their satisfactory performances, DNNs have been applied in various fields, such as face recognition and autonomous driving. However, DNNs evolve in the direction of more layers and nodes, which means that their model capacity is often extremely large, and their time consumption and energy consumption are huge.

2.2. Classical binarization function

DNNs are difficult to deploy into edge computing devices, because the storage capacity of edge computing devices usually cannot match the huge data volume of DNNs. Power consumption is also a major factor hindering the deployment of DNNs to embedded mobile terminals. In order to allow DNNs to be adopted on edge computing devices, neural network binarization has emerged. The sign(·) function is a commonly used function for binarization, which sets numbers greater than 0 as 1, and less than 0 as -1.

In the forward propagation of binary networks, the sign(·) function is often used to quantify the weight or activation to $\{+1, -1\}$. When the weight is $\{+1, -1\}$, simple addition and subtraction can be used instead of the floating point multiplication. And when the activation and weight are both in the set of $\{+1, -1\}$, XNOR-popcount can be used instead of the multiply-accumulate operation. In either case, the inference speed can be greatly improved.

2.3. Reasons for FPGA-based accelerator

As the structure of DNNs becomes more and more complex and the amount of calculations becomes larger and larger, the central processing unit (CPU) is already difficult to perform network training and inference tasks. Therefore, a new computing platform is needed. GPU has high parallelism with extremely high memory bandwidth. However, due to the higher costs and more energy consumption of GPUs, it is difficult to deploy GPUs on the resource-limited edge computing devices.

FPGA is a programmable logic device with a large number of distributed computing and storage resources. FPGA has highly flexible input/output (I/O), lower computing power consumption, and independent, parallel, and pipelined computing functions, which is very suitable for meeting the computing needs of edge devices. Highly flexible I/O allows connection to almost any type of interface and can handle many sensors. FPGA only needs very low power consumption to complete the work. And it can be reprogrammed to support the customized hardware design. With the development of edge computing, FPGA can be easily adapted to new requirements, applications, and functions without bringing a huge cost burden.

Compared with GPU employed on the server, FPGA employed on the edge devices can greatly lower the bandwidth. When GPU is used to improve computing power, computing data needs to share the server's network bandwidth, memory bandwidth, etc., which affects the computing efficiency and exacerbates the server's bandwidth bottleneck. But the data required for FPGA calculations do not need to enter the server. The hardware acceleration process is decoupled from the bandwidth bottleneck of the server to avoid the bandwidth competition with CPU. At the same time, it can undertake some computing tasks originally belonging to CPU and reduce the amount of data and computational complexity entering the server. Using the parallel processing data characteristics of FPGA, the pipeline design can be formed to reduce the dependence on the cache and further reduce the latency. Unlike GPU, FPGA can be deeply customized for each specific application, which is very suitable for edge computing. At the edge node, FPGA serves a queue of requests from various end devices for DNN acceleration. The FPGA device can execute a set of DNN accelerators from a library of DNN accelerators, and each DNN accelerator is optimized for a certain objective, such as accuracy and performances, while GPU cannot. Due to the high degree of autonomous scalability and low power consumption, the FPGA-based inference accelerator has become a hot research topic.

3. BNN and its acceleration based on FPGA

In this section, we investigate several existing neural network binarization methods and FPGA-based acceleration strategies in edge computing environments. We classify them according to their design principles, briefly introduce their technical implementation details, and compare their resource requirements and performances.

3.1. BNN

We divide the binarization methods into four categories.

3.1.1. Quantify weights only

Previously, it was generally thought that binarization would bring catastrophic consequences to the neural network, because the information represented by 1 bit was too little. BinaryConnect [1] conducted a groundbreaking experiment, which narrowed the range

of the weights set as {+1, −1}, and obtained good accuracy on MNIST and CIFAR-10 datasets. Through quantization, not only the size of the parameters is reduced, but simple addition and subtraction can be used to replace the multiplication of weights and activations, as shown in Fig. 2 (a). This experiment [1] eliminated about two thirds of the multiplication requirement. It used two methods to quantify: One was deterministic binarization, which was a sign(·) function; the other was stochastic binarization, which was binarized according to the probability σ, where σ was calculated by the weight w. The stochastic binarization method has stronger robustness, and its performance is better than deterministic binarization. But the deterministic binarization method is simple and conducive to hardware implementation. Most following studies were conducted based on deterministic methods. Since the accuracy of BNN on large datasets (e.g., ImageNet dataset) is lower than that of the full-precision neural network, the binary-weight network (BWN) [4] added a scaling factor to the deterministic binarization to change the set of weights as $\{+a, -a\}$. The optimal value of the scaling factor a can be easily determined by solving the optimization problem. This method can greatly improve the accuracy of the network. However, since the scaling factor is the floating point number, it is impossible to fully exploit the benefits of quantization to 1 bit, and the value of the scaling factor for each weight must be additionally calculated.

3.1.2. Quantify weights and activations

As shown in Fig. 2 (b), when the weight and activation are both quantified to {+1, −1}, the XNOR-popcount operation can be used to replace the multiply-accumulate operation. The XNOR-popcount operation does not require the participation of digital signal processing (DSP) which is scarce in embedded devices. And it can significantly reduce the calculation time.

Binarynet [3] is the first work to quantify both weights and activations to {+1, −1}. Compared with 32-bit DNNs, this Binarynet can greatly reduce the energy consumption of edge computing devices, which can achieve 7 times faster than 32-bit DNNs running on GPU. Its accuracy is only slightly lower than that of BinaryConnect [1], but the accuracy drops more on large datasets. In order to solve this problem, XNOR-Net [4] multiplied the binarized weights and activations by the scaling factor, which can reduce the quantization error by making the quantized value as close as possible to the full-precision value. Although XNOR-Net has indeed greatly improved the accuracy of the neural network, its accuracy still has a large drop on the large dataset like ImageNet. Because the impact of quantifying activations is much greater than that of quantifying weights. In order to speed up the training speed, XNOR-Net even binarizes the gradient. But the scaling factor is still a floating point number, so some of the benefits of binarization are canceled out. The quantization error can also be reduced by adding the custom loss [5]. Unlike XNOR-Net, the work in Ref. [5] made the full-precision value of the weight as close as possible to the quantization value, instead of making the quantization value close to the full-precision value. BNN trained with the distribution loss (BNN-DL) [6] also adds the loss to improve the performance of BNNs. Different from [5], the loss is mainly used to solve the structural problems (i.e., degradation, saturation, and gradient mismatch issues) caused by the activation of binarization. Because BNN-DL uses the loss to change the distribution of the feature map close to the standard normal distribution, it enhances the robustness of BNN to hyperparameters, and the accuracy is higher than that of XNOR-Net. But it does not use different distribution coefficients for different loss, which will lead to the degradation of DNN performances.

3.1.3. 0/1 quantization

As shown in Fig. 2 (c), AND-popcount can be used to replace the multiply-accumulate after 0/1 quantization. It can improve the calculation speed through bit operations, and save the sign bit. Previously, whether +1/−1 quantization was the only option for binarization was largely overlooked. Xian et al. in Ref. [10] quantified activations and weights to {0, 1}, and deployed the network on FPGA. Experiments showed that the accuracy of the 0/1 quantized DNN on the MNIST dataset could achieve the accuracy of more than 98%, and the accuracy fluctuation was smaller than +1/−1 quantization. Sparsity-inducing BNN (Si-BNN) [11] proved that 0/1

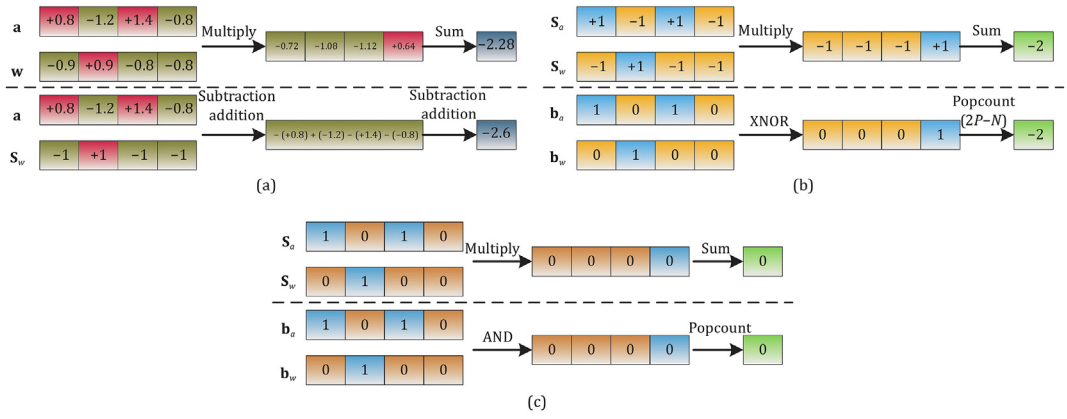

Fig. 2. Schematic diagram of calculation operations after different binarization, where **a** and **w** are the floating point representation of activations and weights, $\mathbf{S}_a$ and $\mathbf{S}_w$ are the binarized representation of activations and weights, and $\mathbf{b}_a$ and $\mathbf{b}_w$ are the representation of activations and weights in the hardware memory: (a) example of convolution with only addition and subtraction operations; (b) example of using XNOR-popcount instead of multiply-accumulate operations, where P is the value of the popcount result, and N is the number of bits in the vector; (c) example of using AND-popcount instead of multiply-accumulate operations.

binarization is not bad or even better than $+1/-1$ binarization. Si-BNN performs very well on large datasets, but it only quantifies the activation to {0, 1}, and the weight set is still $\{+a, -a\}$ (i.e., multiplied by a scaling factor). In order to improve the performance of the network, Si-BNN introduces a trainable threshold into the binarized forward and backward propagation functions to guide the gradient propagation. But this method also makes its training process more complicated.

3.1.4. Updated binary architecture expansion

Traditional BNNs represent original full-precision weights and activations into 1 bit with a sign function. The derivative of the sign function is 0 in many places and is not derivable at the center, so the sign function cannot meet the requirements of accurate back-propagation. Therefore, many methods use approximate gradients for back-propagation, such as the most commonly used straight-through estimator (STE) algorithm. However, these approximations corrupt the main direction of the fact gradient. Frequency domain approximation-BNN (FDA-BNN) proposed by Xu et al. [13] and the method proposed by Lee et al. [14] tried to use the combination of the sine function to approximate the gradient of the sign function in the Fourier frequency domain to train BNN. Lee et al. analyzed the effect of the period and the number of terms of the Fourier series representation on the network accuracy. But Lee et al. did not open-source the code. The code of FDA-BNN is published on GitHub. And FDA-BNN achieved 92.54% accuracy on CIFAR-10 and the top-1 accuracy of 66.0% on ImageNet with ResNet-18. The improvement of the sign function itself is studied, because the use of a single sign function will lead to the distortion of the output distribution. The adaptive binary method (AdaBin) [15] can adaptively obtain the weight of each layer and the activated optimal binary set, rather than the fixed set (i.e. $\{-1,+1\}$). AdaBin obtained 66.4% top-1 accuracy on the ImageNet using the ResNet-18 architecture. AdaBin can well fit different distributions and improve the representation ability of binary features, but it adds binary sets of different values, which increases the cost and difficulty of hardware deployment. Batch normalization (BN) plays an important role in BNNs, but it has a high computing cost for hardware, because it involves floating point division and other complex calculations. Chen et al. [16] for the first time demonstrated that BN can be completely removed from BNN training and reasoning. Through the insertion and customization technologies (including adaptive gradient clipping, scaling standardization, and the special bottleneck block), the precision of BNN without BN was only 1.40% lower than that of BNN with BN on ImageNet. They exposed the codes and models on GitHub. Jiang et al. [17] applied BNN without BN to the super-resolution (SR) field. A lightweight network architecture and multi-stage knowledge extraction strategy were developed to enhance the model representation ability. Experiments showed that the method proposed in Ref. [17] was superior to the latest binary method on the standard super-resolution (SR) network.

The rectified clamp unit (ReCU) [18] for the first time explored the influence of "dead weight" which refers to a group of weights that are barely updated during the training of BNNs. ReCU mathematically proves that restoring the "dead weight" can lead to a small quantization error, and the "dead weight" can be updated by introducing a correction clamping unit. The code was exposed on GitHub. Many studies have optimized and adjusted the binary network, but few people have completely binarized the input layer. FracBNN [19] uses additional binary convolution to calculate the characteristics of up to two bits, and binarizes the input layer using thermometer coding. By using the accelerator designed for the fractional convolution kernel, FracBNN achieves the high-performance real-time image classification capability on FPGA.

3.2. FPGA acceleration

The methods of hardware acceleration for BNNs can generally be divided into the following three types.

3.2.1. XNOR-popcount

After the activation and weight are binarized, the original multiply-accumulate operation of 32-bit floating point numbers can be solved by the XNOR-popcount operation, which has great potential for model acceleration. Fig. 2 (b) depicts the process of the XNOR-popcount operation. However, the popcount calculation process requires high costs of time and resources for long vectors. For this reason, FP-BNN [20] leveraged compressed trees to calculate popcount, which saves resources for long vectors. It sets the compression ratio to 6:3 to match the hardware. In order to speed up the processing speed, FP-BNN ignores the bias, and achieves a high performance on FPGA. But it has high power consumption, and the compression tree method is not suitable for extremely short vectors. Recently, 0/1 quantization has been proposed to reduce the source consumption. AND-popcount that requires fewer computational operations can replace XNOR-popcount through 0/1 quantization that introduces sparsity into BNN. If one of the weights and the activation multiplier is 0, they do not need to be transmitted to the processing element (PE) unit [10], because their product must be 0. It can reduce the power consumption of data transmission and obtain a higher performance on FPGA compared with the method in Ref. [20].

3.2.2. Partial quantization

TaiJiNet [2] proposed partial BNN for embedded systems, because sometimes the hardware can run well without binarizing all, and binarization will lead to a decrease in accuracy. TaiJiNet has formulated the mean-l_1 and mean-l_2 standards through research and comparison. The weight will be binarized only when the value of the weight is less than the threshold. TaiJiNet has high accuracy on large datasets. However, partial binarization means that the weights have both floating point numbers and 1-bit fixed-point numbers. Therefore, the hardware must be matched with the network structure to better exert the parallelism of FPGA.

3.2.3. Prune redundancy

The binary network has greatly reduced the size of the model, but it still has redundancies. Fujii et al. [7] pruned the binary weights of the dense layer directly, eliminated neurons below the threshold, and then trained again. They repeated these steps 12 times. This has

trimmed 60% of the dense neurons and achieved a high frame rate with low power consumption on the XC7Z020 development board. However, pruning neurons will make the accuracy of the model lower. O3BNN [8] chose to skip unnecessary hardware operations to trim the redundancy, which combined the batch normal and the sign(·) function into a threshold comparison operation. If the accumulation of convolution is already greater than the threshold, or it is impossible to reach the threshold, the following calculations will be skipped. O3BNN can dynamically trim irregular redundant edges during inference, obtaining the latency lower than that of [7]. But its premise is 0/1 quantization, and the max-pooling layer must be after the batch normal. Fu et al. [9] also chose to skip unnecessary hardware operations. Most images have spatial continuity, and the adjacent pixels are similar. In consequence, the binarized feature maps are also very likely to be spatially similar. Weights can be viewed as spatial coefficients of feature maps, so weights also have similarities. Fu et al. first checked the difference between the current feature map or weight and the previous one. Then, they updated the previous result based on this difference. In this way, 80% of calculations and 40% of buffer access were skipped on average. The previous hardware design must use at least two resistive random access memory (RRAM) units to store weights to carry out the XNOR operation between the binary weights and binary activations. Oh et al. [21] converted the calculation based on XNOR into the RRAM-friendly multiplication, which reduced the number of RRAM units required by half. They achieved an area-efficiency increase of about 1.9 times and an energy-efficiency increase of about 1.8 times.

4. Comparison and evaluation

4.1. Comparison of BNN

4.1.1. Comprehensive comparison

Table 1 summarizes the work of existing BNNs. It is obvious from the table that most existing studies [1,3–6,11] are designed based on the deterministic binarization method. This is because deterministic methods are conducive to hardware implementation. But in this way, the accuracy advantage of the stochastic binarization method cannot be enjoyed. It can also be seen from Table 1 that the accuracy of BNNs on the MNIST dataset is close to 100%, while the accuracy on large datasets is quite low. Therefore, many studies are devoted to increasing the accuracy of BNN on large datasets [4–6,11]. The method of adding a scaling factor [4,11] can greatly improve the accuracy and even bring the accuracy close to the result of the full-precision network. But it also destroys the benefits of the bitwise operation. The method of adding loss [5,6] can also greatly improve the accuracy of the network without destroying the benefits of binarization. Regardless of the binary method, the compression ratio of models is high. Binary weights can compress the model by at least 16 times, close to 32 times. When the weight and activation are binarized, the model size is almost compressed by 32 times. Table 1 shows that XNOR-Net [4] consumes more energy than Binarynet [3] and BNN-DL [6], because the scaling factor used by XNOR-Net [4] introduces floating point operations into convolution calculations.

4.1.2. Performance evaluation

In order to provide more intuitive comparison, we conduct experiments to evaluate the performances of BNC, Binarynet, XNOR-Net, and BNN-DL. Our experiments are performed on the CIFAR-10 dataset. We use the network structure of VGG16 which can be formulated

Table 1
Comparison of the BNN studies.

Dataset	Works	Binarization type	Weight/activation width (bit)	Accuracy (%)	Compression ratio	Energy consumption (convolution layer) (μJ)
MNIST	BinaryConnect [1]	Deterministic	1/32	98.71	At least 16 ×	/
	BinaryConnect [1]	Stochastic	1/32	98.82	At least 16 ×	/
	Binarynet [3]	Deterministic	1/1	99.04	About 32 ×	1.42
CIFAR-10	BinaryConnect [1]	Deterministic	1/32	90.10	At least 16 ×	/
	BinaryConnect [1]	Stochastic	1/32	91.73	At least 16 ×	/
	BWN [4]	Deterministic	1/32	90.12	At least 16 ×	/
	Binarynet [3]	Deterministic	1/1	88.60	About 32 ×	1.42
	BNN-DL [6]	Deterministic	1/1	89.90	About 32 ×	1.42
	XNOR-Net [4]	Deterministic	1/1	89.83	About 32 ×	4.34
	Min-QL [5]	Deterministic	1/1	89.10	About 32 ×	/
	Si-BNN [11]	Deterministic	1/1	90.20	About 32 ×	/
ImageNet	BWN [4]	Deterministic	1/32	56.8 (top-1), 79.4 (top-5)	At least 16 ×	/
	XNOR-Net [4]	Deterministic	1/1	44.2 (top-1), 69.2 (top-5)	About 32 ×	4.34
	BNN-DL [6]	Deterministic	1/1	47.8 (top-1), 71.5 (top-5)	About 32 ×	1.42
	Si-BNN [11]	Deterministic	1/1	50.5 (top-1), 74.6 (top-5)	About 32 ×	/

as: (2 × 128C)-MP2-(2 × 256C)-MP2-(2 × 512C)-MP2-(2 × 1024FC)-10SoftMax. (2 × 128C) represents the two convolution layers of 128 output channels, MP2 represents the max-pooling layer with a stride of 2, and (2 × 1024FC) represents two dense layers of 1024 nodes. We set the batch size to 256 and the initial learning rate to 0.001 (for the BNC experiment, we set it to 0.003), and train for 200 epochs. We use the learning rate decay to improve learning efficiency.

The experiment results of trained network accuracy are shown in Fig. 3. In our experiment, the accuracy of BNC is significantly higher than that of other binarization methods, and it approximates the accuracy of the full-precision network. Since BNC only quantifies the weights while ignoring the quantification of activations, the impact on the accuracy of DNN is relatively small. In our experiment, the accuracy of Binarynet decreases significantly, because the influence of binarization of activations is much greater than that of weight binarization. This shows that binarizing activations that contain a lot of feature information will indeed lead to the problem of information loss. As shown in Fig. 3, the accuracy of BNN-DL is similar to that of XNOR-Net, i.e., both around 0.88, which is much higher than that of Binarynet. This can be attributed to their efforts to solve the problem of missing information. But BNN-DL has better portability, because it only needs to add loss based on the original network structure. If the method of BNN-DL is used on the basis of the structure of XNOR-Net, the accuracy can be further improved. Their training curve is shown in Fig. 4. The fitting speed of the full-precision network is fast, and it reaches the saturation state quickly and is smooth. Because BNC quantifies the weight, its fitting curve produces greater jitter compared with the full-precision network. Binarynet, XNOR-Net, and BNN-DL binarize the activations and weights, resulting in huge jitter in the fitting curve due to the unstable change of network parameters.

4.2. Comparison of FPGA acceleration

Table 2 gives the comparison of the indicators among the acceleration studies. Since different studies may have different choices on indicators, some of which may not be fairly comparable, such as performance indicators. FP-BNN [20] improves the XNOR-popcount operation and obtains a great performance, but it has high power consumption which is much higher than other studies. Directly pruning the nodes of the dense layer of neurons [7] can obtain a weight compression rate of more than 40 times, and achieve a high frame rate with low power consumption. But its accuracy is much lower than other studies, and it only has 81.80% accuracy on CIFAR-10. Therefore, most studies [8–10] have chosen to skip unnecessary computations instead of pruning directly. According to the sparsity of the network, many calculation steps can be skipped to reduce power consumption [10]. In terms of the performance and power consumption ratio (performance/power), Ref. [10] achieved 1624 GOP/(s·W) (Giga operations per (second·Watt)) on MNIST. Fu et al. [9] achieved 2347 GOP/(s·W) based on the similar weights and activations, which is much higher than that of Ref. [20]. TaiJiNet [2] is a partial binarization method for embedded devices. It can compress the weight by more than 20 times, and the compression rate can be manually adjusted according to hardware requirements. Its accuracy is only 0.8% lower than that of the full-precision network as shown in Ref. [2], but it has not been tested on FPGA or other hardware.

5. Future research directions

Although BNNs and corresponding accelerators have shown promising performances in the edge computing environments, the accuracy of BNN on large datasets and the efficiency of hardware calculations still have the potential to improve.

5.1. BNN perspective

Most existing BNN studies have chosen to quantify the weight and activation to {+1, −1} instead of {0, 1} as in Si-BNN [11]. Though {+1, −1} quantization greatly reduces the parameter size, it needs a sign bit to be represented for the hardware implementation. Since one bit of the hardware actually represents {0, 1}, 0/1 quantification [10,11] is more suitable to be deployed on edge computing

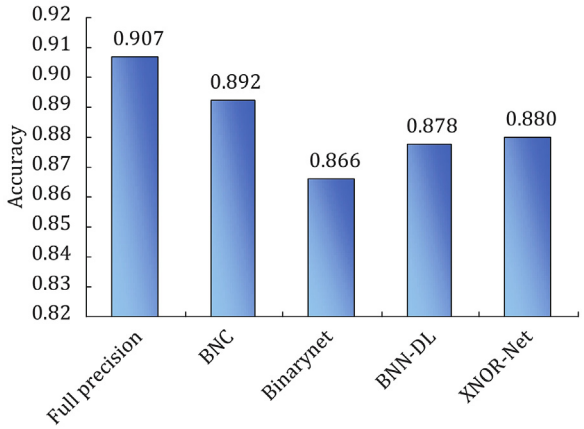

Fig. 3. Accuracy comparison on CIFAR-10 dataset.

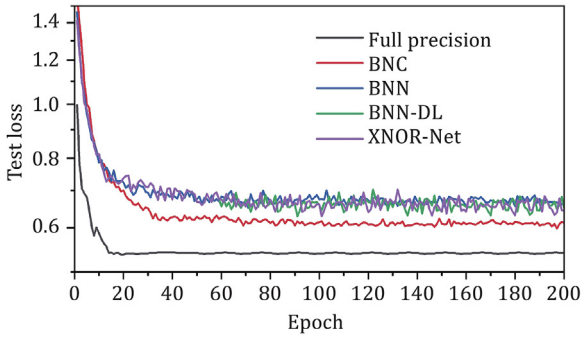

Fig. 4. Fitting curve of each network on the CIFAR-10 dataset.

Table 2
Comparison of the acceleration studies.

Dataset	Method	Platform	Clock (MHz)	Performance	Power (W)	Accuracy (%)	Parameter compression rate
MNIST	FPGA'20 [10]	Vertex-690T	500	3378 (GOP/s)	2.08	98.52	/
CIFAR-10	FP-BNN [20]	Stratix-V 5SGSD8	150	17646.50 (GOP/s)	26.2	86.31	31.9 ×
	O3BNN [8]	ZC706	200	1.82×10^5 (Image/kJ)	/	88.50	/
	FPGA'19 [9]	Zynq XCZU7EV	200	539.9 (GOP/s)	0.23	88.81	/
	FPGA'18 [7]	XC7Z020	143	408 (frames per second (FPS))	2.2	81.80	42.4 ×
ImageNet	TaiJiNet [2]	/	/	/	/	55.8 (Top-1), 78.7 (Top-5)	22.1 ×

devices. Experiment results show that 0/1 quantization introduces sparsity into BNN, which can be further exploited to improve the efficiency of hardware accelerators. In the future, researchers could explore quantifying the weights and activations to {0, 1}.

There have been many studies [4,6,11] that have achieved high accuracy on large datasets, but most of them use scaling factors, which makes it impossible to use XNOR-popcount to completely replace the multiply-accumulate operation. And it needs type conversion. The accuracy can also be improved to a large extent by adding loss [5,6], which optimizes the DNN training process without affecting the maximum convenience that is brought by the binarization to the hardware. Because an important purpose of binarization is to provide favorable conditions for the hardware acceleration design, improving the accuracy of BNNs without affecting the benefits brought to the hardware will also be a direction point for future work.

Binarization is an effective method to solve the problem of large models, but it usually only appears in classification networks due to the problem of missing information. However, classification tasks are only one of many fields of machine learning. Limiting the application of BNNs to classification tasks will limit the benefits of BNNs for machine learning to a large extent. Ma et al. [22] applied binarization to the task of image super-resolution enhancement. The network was still inadequate in terms of high-frequency details, but it was not easy to notice with the naked eye. Although they only used binarized weights with scaling factors in the residual blocks, they could reduce the model size by almost 80% on SRResNet, and it was possible to increase the inference speed by 5 times. Heng et al. [23] applied BNN to traffic sign detection. It plays an important role in the field of automatic driving. The attempt not only enhances the real-time response of automatic driving, but also has an impact on the practical significance of BNNs. Therefore, the application of binarization methods outside of classification tasks (e.g., pixel-level image reconstruction tasks) will be of great significance.

5.2. FPGA accelerator perspective

Binarization methods include deterministic binarization and stochastic binarization, but current hardware accelerators are almost all based on deterministic binarization. Although this is convenient for hardware implementation, it also gives up the high precision brought by stochastic binarization. Lammie et al. [24] accelerated stochastic binarization on FPGA and achieved 602 FPS with the power consumption of 6.6 W. This shows that stochastic binarization can also be accelerated on FPGA. In the future, it is possible to explore the use of stochastic binarization to maintain the high accuracy of the network, while achieving low-power acceleration.

How to use sparsity is a future research focus of FPGA hardware accelerators. The emergence of 0/1 quantization [10,11] brings a lot of sparsity to BNNs, as a large number of activations and weights elements are 0 by using 0/1 quantization. Due to the product of 0 and any number does not affect the resulting value, lots of elements can be omitted. By exploiting the sparsity that 0/1 quantification brings, hardware can skip 58.8% of unnecessary memory access or computational operations [10] to achieve lower power consumption and higher performances. In the future, using sparsity to reduce unnecessary steps should be explored by hardware accelerator researchers.

6. Conclusions

Combining binarized hardware accelerators can greatly reduce the power consumption and parameter size, thus improving the inference speed. This allows DNNs to be much friendlier to be deployed in edge computing. In this article, we comprehensively overviewed BNNs and the FPGA-based hardware accelerator design in edge computing environments. We divided BNNs into four categories, and then grouped the FPGA-based hardware acceleration schemes into three categories. We reviewed the up-to-date technologies and evaluated the performances of the binarized neural network through experiments. Finally, we discussed future directions for improving the efficiency and performance in edge computing environments.

Funding

This work was supported by the Natural Science Foundation of Sichuan Province of China under Grant No. 2022NSFSC0500; the National Natural Science Foundation of China under Grant No. 62072076.

Declaration of competing interest

The authors declare no conflicts of interest.

References

[1] M. Courbariaux, Y. Bengio, J.P. David, BinaryConnect: training deep neural networks with binary weights during propagations, in: Proc. of 28th Intl. Conf. on Neural Information Processing Systems, Montreal, 2015, pp. 3123–3131.
[2] Y.-J. Ling, K. Zhong, Y.-S. Wu, et al., TaiJiNet: Towards partial binarized convolutional neural network for embedded systems, in: Proc. of Computer Society Annual Symposium on VLSI, Hong Kong, 2018, pp. 136–141.
[3] M. Courbariaux, Y. Bengio, Binarynet: Training deep neural networks with weights and activations constrained to +1 or −1 [Online]. Available, https://arxiv.org/pdf/1602.02830v1.pdf, March 2016.
[4] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-net: ImageNet classification using binary convolutional neural networks, in: Proc. of 14th European Conf. on Computer Vision, Amsterdam, 2016, pp. 525–542.
[5] F. Zheng, C. Deng, H. Huang, Binarized neural networks for resource-efficient hashing with minimizing quantization loss, in: Proc. of 28th Intl. Joint Conf. on Artificial Intelligence, Macao, 2019, pp. 1032–1040..
[6] R.-Z. Ding, T.W. Chin, Z.-Y. Liu, D. Marculescu, Regularizing activation distribution for training binarized deep networks, in: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition, Long Beach, 2019, pp. 11408–11417.
[7] T. Fujii, S. Sato, H. Nakahara, A threshold neuron pruning for a binarized deep neural network on an FPGA, IEICE Trans. Info. Syst. E101.D (2) (Feb. 2018) 376–386.
[8] T. Geng, T.-Q. Wang, C.-S. Wu, et al., O3BNN: An out-of-order architecture for high-performance binarized neural network inference with fine-grained pruning, in: Proc. of the ACM Intl. Conf. on Supercomputing, Phoenix, 2019, pp. 461–472.
[9] C. Fu, S.-L. Zhu, H. Su, C.-E. Lee, J.-S. Zhao, Towards fast and energy-efficient binarized neural network inference on FPGA, in: Proc. of ACM/SIGDA Intl. Symposium on Field-Programmable Gate Arrays, Seaside, 2019, pp. 306–317.
[10] Z.-K. Xian, H.-G. Li, Y.-L. Li, Weight isolation-based binarized neural networks accelerator, in: Prof. of IEEE Intl. Symposium on Circuits and Systems, Seville, 2020, pp. 1–4.
[11] P.-S. Wang, X.-Y. He, G. Li, T.-L. Zhao, J. Cheng, Sparsity-inducing binarized neural networks, in: Proc. of 34th AAAI Conf. on Artificial Intelligence, New York, 2020, pp. 12192–12199.
[12] K. Jia, M. Rinard, Efficient exact verification of binarized neural networks, in: Proc. of 34th Intl. Conf. on Neural Information Processing Systems, Vancouver, 2020, pp. 151:1–14.
[13] Y.-X. Xu, K. Han, C. Xu, Y.-H. Tang, C.-J. Xu, Y.-H. Wang, Learning frequency domain approximation for binary neural networks, in: Proc. of 35th Conf. on Neural Information Processing Systems, Online, 2021, pp. 25553–25565.
[14] S.Y. Lee, H. Kwak, J.S. No, Effect of the period of the Fourier series approximation for binarized neural network, in: Proc. of 4th Intl. Conf. on Artificial Intelligence in Information and Communication, Jeju Island, 2022, pp. 262–265.
[15] Z.-J. Tu, X.-H. Chen, P.-J. Ren, Y.-H. Wang, AdaBin: Improving binary neural networks with adaptive binary sets, in: Proc. of 17th European Conf. on Computer Vision, Tel Aviv, 2022, pp. 379–395.
[16] T.-L. Chen, Z.-Y. Zhang, X. Ouyang, Z.-C. Liu, Z.-Q. Shen, Z.-Y. Wang, "BNN − BN = ?": Training binary neural networks without batch normalization, in: Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops, Nashville, 2021, pp. 4614–4624.
[17] X.-R. Jiang, N.-N. Wang, J.-W. Xin, K.-Y. Li, X. Yang, X.-B. Gao, Training binary neural network without batch normalization for image super-resolution, in: Proc. 35th AAAI Conf. on Artificial Intelligence, Online, 2021, pp. 1700–1707.
[18] Z.-H. Xu, M.-B. Lin, J.-Z. Liu, et al., ReCU: Reviving the dead weights in binary neural networks, in: Proc. of IEEE/CVF Intl. Conf. on Computer Vision, Montreal, 2021, pp. 5178–5188.
[19] Y.-C. Zhang, J.-H. Pan, X.-H. Liu, H.-Z. Chen, D.-M. Chen, Z.-R. Zhang, FracBNN: Accurate and FPGA-efficient binary neural networks with fractional activations, in: Proc. of ACM/SIGDA Intl. Symposium on Field-Programmable Gate Arrays, Online, 2021, pp. 171–182.
[20] S. Liang, S.-Y. Yin, L.-B. Liu, W. Luk, S.-J. Wei, FP-BNN: Binarized neural network on FPGA, Neurocomputing 275 (Jan. 2018) 1072–1086.
[21] H. Oh, H. Kim, N. Kang, Y. Kim, J. Park, J.J. Kim, Single RRAM cell-based in-memory accelerator architecture for binary neural networks, in: Proc. of IEEE 3rd Intl. Conf. on Artificial Intelligence Circuits and Systems, Washington, 2021, pp. 1–4.
[22] Y.-L. Ma, H.-Y. Xiong, Z. Hu, L.-Z. Ma, Efficient super resolution using binarized neural network, in: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition Workshops, Long Beach, 2019, pp. 694–703.
[23] E. Heng, M. Vemparala, N. Fasfous, et al., Investigating binary neural networks for traffic sign detection and recognition, in: Proc. of IEEE Intelligent Vehicles Symposium, Nagoya, 2021, pp. 1400–1405.
[24] C. Lammie, W. Xiang, M.R. Azghadi, Accelerating deterministic and stochastic binarized neural networks on FPGAs using openCL, in: Proc. of IEEE 62nd Intl. Midwest Symposium on Circuits and Systems, Dallas, 2019, pp. 626–629.

Jin-Yu Zhan was born in Heilongjiang, China in 1978. She received the B.S., M.S., and Ph.D. degrees from University of Electronic Science and Technology of China (UESTC), Chengdu, China in 2000, 2003, and 2006, respectively. She is currently an associate professor with the School of Information and Software Engineering, UESTC. Her current research interests include embedded AI, dependable AI, FPGA accelerators, and embedded system designs.

An-Tai Yu was born in Henan, China in 1996. He received the B.S. degree from China Jiliang University, Hangzhou, China in 2020. He is currently pursuing the M.S. degree with the School of Information and Software Engineering, UESTC. His research interests include dependable AI and embedded system designs.

Wei Jiang was born in Sichuan, China in 1981. He received the B.S., M.S., and Ph.D. degrees from UESTC in 2003, 2006, and 2009, respectively. He is currently an associate professor with the School of Information and Software Engineering, UESTC. His current research interests include dependable AI, embedded AI, hardware/software co-designs, and embedded system designs.

Yong-Jia Yang was born in Sichuan, China in 1996. He received the B.S. degree from UESTC in 2018. He is currently pursuing the M.S. degree with the School of Information and Software Engineering, UESTC. His research interests include dependable AI and embedded system designs.

Xiao-Na Xie was born in Henan, China in 1978. She received the Ph.D. degree from UESTC in 2014. She is currently an associate professor with the School of Automation, Chengdu University of Information Technology, Chengdu, China. Her current research interests include embedded AI, dependable AI, and intelligent computing.

Zheng-Wei Chang was born in Henan, China in 1981. He received the Ph.D. degree from UESTC in 2009. He is currently a professor with the State Grid Sichuan Electric Power Research Institute, Chengdu, China. His current research interests include embedded AI, dependable AI, embedded system designs, and smart grids.

Jun-Huan Yang was born in Sichuan, China in 1993. He received his B.S. and M.S. degrees in software engineering from UESTC in 2015 and 2018, respectively. He is currently a Ph.D. candidate with the Department of Information Sciences and Technology, George Mason University, Fairfax, USA. His current research interests include embedded AI and embedded system designs.