Designing Deep Learning Hardware Accelerator and Efficiency Evaluation

Zhi Qi, Weijian Chen, Rizwan Ali Naqvi, and Kamran Siddique

1Department of Information and Communication Technology, School of Computing and Data Science, Xiamen University Malaysia, Sepang 43900, Malaysia
2Department of Intelligent Mechatronics Engineering, Sejong University, Seoul 05006, Republic of Korea

Correspondence should be addressed to Kamran Siddique; kamran.siddique@xmu.edu.my

Received 7 May 2022; Accepted 22 June 2022; Published 13 July 2022

1. Introduction

With the striking expansion of the Internet and the swift development in the big data era, artificial intelligence has been widely developed and used in the past thirty years [1]. People’s continuous significant breakthroughs in deep learning algorithms have brought the rapid evolution of artificial intelligence technology which play an essential role in the considerable variety of machine learning algorithms. Deep learning is a technology that simulates human brain neural transmission, extracts feature hierarchically, and continuously adjusts parameters from experience and data. It has a strong learning ability. The training of deep learning network model parameters needs the support of massive data and supercomputing power. We must find a better hardware computing acceleration scheme to meet the increasing amount of data and the expanding network scale. For deep learning, the central processing unit (CPU) is highly flexible, but the computing power is insufficient, and for the accelerator, the situation is entirely opposite. A combination of CPU and accelerator is a popular solution to balance flexibility and efficiency [2]. Meanwhile, manufacturers are studying the architecture of various deep learning accelerators, including GPU accelerators and DSP accelerators.

At present, the most widely used solution is to use a graphics processing unit (GPU) cluster as a heterogeneous computing system composed of general computing graphics processing unit (GPUPU) and CPU for computing acceleration. However, there are some problems in using GPU clusters. With the continuous expansion of cluster scale, colossal power consumption and heat dissipation have become the limiting factors of its scale expansion. Therefore, we need to find a better acceleration scheme with higher computing performance and lower energy consumption. A field-programmable gate array (FPGA) fully meets our high performance and low power consumption requirements as a new heterogeneous parallel computing acceleration device.
As a newly developing hardware architecture, FPGA has its unique advantages. With the continuous improvement of FPGA, it has an excellent performance power consumption ratio compared with GPU.

In addition, FPGA developers can flexibly program hardware according to specific applications, and by comparison, the structure of CPU cannot be altered once the design is completed. Lack of flexibility makes GPUs unable to apply well to most scenarios. Simultaneously, the emergence of advanced synthesis tools, such as high-level synthesis (HLS) and OpenCL, further shortens the cycle of developing a convolutional neural network (CNN) accelerator on FPGA. In recent years, FPGA has been deployed on a large scale in the data centers of Microsoft, Amazon, Alibaba, and other companies. Nowadays, cloud computing servers only not provide deep learning computation acceleration with conventional GPU but also supply customizable FPGA plans. In addition, researchers also start to apply FPGA in the domain of edge computing such as auto pilot and the industrial robot [4]. Although the accelerator based on FPGA has attracted attention in deep learning, non-ignorable weaknesses still exist to overcome. Hence, the design of the FPGA deep learning accelerator needs to be further improved.

At present, most researches accelerate the hardware based on the forward propagation characteristics of neural network. By analyzing the parallelism of convolution algorithm of neural network, we can adapt the flexible programming of FPGA to reconstruct the suitable framework in order to achieve optimal utilization of the parallelism of convolutional algorithm. This approach can essentially help to further improve the computing speed and efficiency.

### 2. Related Work

Hardware acceleration is to make full use of the inherent characteristics of hardware, use hardware computing module to replace the software algorithm run by general CPU, share the workload of CPU, and improve the overall efficiency. Typical hardware accelerations include ASIC, FPGA, and GPU. Because FPGA not only has the advantages of high speed and stability like ASIC but also has the flexibility of customizing circuit structures like a software program, FPGA has attracted the attention of most scholars [4].

Chen et al. [5] proposed that matching computing throughput and memory bandwidth should be fully considered in implementing the CNN model on FPGA. Therefore, they designed the analysis method according to the roof model. The experiment implemented a CNN model on the VC707 FPGA development board. The results showed that the peak performance was 61.62 GOP/s at 100 MHz, which was better than the previous work.

Li et al. [6] proposed a way to find parallel strategies for each layer. In addition, with two distinctive matrix calculation modes on the FC layer, this design can significantly reduce the on-chip cache space required. Finally, the AlexNet network was run on Xilinx VC709, which realized the peak performance of 565.94 GOP/s and 391 fps at 156 MHz clock frequency.

Kalms et al. [7] conducted a comparative study on implementing feature detection algorithm using OpenCL on a heterogeneous system including CPU, GPU, and FPGA, and compared differences in energy efficiency and architecture performance under different optimization strategies. It showed how to use Xilinx’s Edexcel tool to maximize performance and maintain low resource utilization on the premise that FPGA reached the bandwidth limit. The results demonstrated the speed of FPGA was increased by 62.8 times, and the energy consumption was reduced by 49 times in comparison with the optimized single threaded CPU.

Liang et al. [8] pointed out that the traditional CNN algorithm (i.e., spatial convolution algorithm and GEMM algorithm) is often limited by the number of computing units of FPGA. Winograd fast algorithm can still reduce the computational complexity significantly. The authors first proposed a new method of deploying the Winograd algorithm on FPGA to reuse data using line buffer structure efficiently. At the same time, they also adopted pipeline and parallel technology. They proposed an analysis model for complex design space to predict resource utilization and performance. The experiments obtained the convolutional layer average performance of 1006.4 GOP/s and the average performance of 854.6 GOP/s on AlexNet using Xilinx ZCU102 development board. The convolutional layer average performance of 3044.7 GOP/s and the overall average performance of 2940.7 GOP/s were obtained on VCC 16.

Cong et al. [9] of the University of California analyzed the characteristics of convolution operation from the mathematical level and used matrix multiplication to calculate convolution. In their study, several optimization methods for convolution matrix multiplication were proposed and compared with traditional matrix multiplication.

Intel proposed the FPGA deep learning accelerator (DLA) based on Intel OpenCL [10]. The accelerator adopted the Winograd algorithm and used the feature that a DSP on the Altera platform was able to handle the multiplication of two half-precision floating-point numbers. DLA adopted a one-dimensional Winograd algorithm in the specific implementation which reduced the multiplication times significantly. Finally, the performance of 1.38 TOP/s was achieved on the Intel Altera Aria 10 platform.

### 3. Field-Programmable Gate Array Architecture

A field-programmable gate array (FPGA) is a kind of integrated circuit, which can be modified repeatedly to design different circuit functions [3]. Designers can achieve the desired logic operation by configuring the switch state. Symbolically, FPGA acts as a pile of building blocks. Developers can reconstruct its hardware structure through programming to build FPGA into any shape required. Programmable logic (PL), the FPGA’s core part, is mainly comprised of the configurable logic block (CLB) and input/output block (IOB). As shown in Figure 1(a), CLBs are arranged in a two-dimensional array. Each CLB includes two logic slices and is adjacent to a switch matrix. Logic slice is mainly composed of the lookup table and flip-flop. They can
be configured into different composition logic, which gives FPGA reconfigurable characteristics.

Figure 1(b) shows the basic design of the CNN FPGA accelerator. From the perspective of data, FPGA can be roughly divided into three layers: off-chip storage, on-chip cache, and register [11]. The off-chip storage usually uses DRAM memory, and the on-chip cache is block RAM. Because the on-chip cache cannot store the whole input dataset, it usually holds part of the data. The input buffer relates to the computing unit through the data transmission control unit, which transmits data required to the internal register of the computing unit each time. FPGA has the characteristics of parallel computing. Many internal computing units can calculate simultaneously, and the efficiency can be significantly improved.

FPGA is widely used in more and more fields because of its high programming flexibility and powerful computing performance. Compared with GPU, FPGA has the following advantages:

1. FPGA uses fine-grained parallel architecture. After determining the computing clock cycle time method, the designer can establish a more optimized output mode that eliminates the need to save data in the main memory to reduce memory reading delay.

2. Compared with GPU, FPGA has muscular programming flexibility. At the same time, FPGA supports dynamic reconfiguration of the algorithm, which has strong reconfigurability.

3. FPGA is much lower than GPU in power consumption and has better performance under the same power consumption, which can significantly reduce the heat dissipation problem of the processor.

4. Convolutional Neural Network

The CNN model imitates the structure of a biological vision system and is a feedforward artificial neural network. The core idea of CNN has three points: local perception, weight sharing, and pooling [13].

Local perception refers to the connectivity of network parts. Every neuron is only connected with part of the upper layer neurons and senses the local part instead of the entire image. The intermediary of upper and lower neurons is called the “convolutional kernel.” Weight sharing implies that all neurons use the same weight in the same convolutional kernel, remarkably reducing the number of network parameters and making it feasible to train large-scale networks. The model can learn from a local region and then convolute the whole image with the same convolution kernel. Pooling is a kind of downsampling that will separate the input images into a few rectangular regions and outputs the maximum values for each sub-region.

CNN comprises three common layers: convolutional layer, pooling layer, and fully connected layer. In general, the convolutional layer is the principal layer in the CNN model. It performs convolution operations, like the image filtering function. Convolution is a local operation that “rolls” on the feature map through a specific convolution kernel size to realize the feature extraction of the image. The pooling layer is a process of dimension reduction and extraction of the visual input image based on a biological vision system. The
fully connected layer classifies the data output in a CNN model.

4.1. CNN Multilayer Computing Model. The CNN model selected in this paper is AlexNet model, which is the champion of ImageNet competition in 2012 [14]. It contains five convolutional layers, three fully connected layers, and three maximum pooling layers. Each layer of the network uses the ReLU activation function except the last one. Table 1 shows the parameters of AlexNet model structure. \( N_{in} \) is the number of input feature maps, \( N_{out} \) is the number of output feature maps, \( \text{Size}_i \) and \( \text{Size}_o \) represent the sizes of input feature maps and output feature maps. \( \text{Size}_{kernel} \) is convolution kernel’s size, and \( \text{Stride} \) is window sliding size in the process of convolution and pooling.

Convolution is the most important part of CNN model and the most computationally intensive part, which usually accounts for more than 90% of the computation of the whole neural network. Convolution means the process of accumulating by the product of input feature maps \( N_{if} \) and \( K \times K \) convolutional kernels, and corresponding eigenvalues are extracted and output as neuron parameters, shown in (1).

\[
\text{out}(f_o, x, y) = \sum_{f=0}^{N_{if}} \sum_{k_x=0}^{K} \sum_{k_y=0}^{K} \text{wt}(f_o, f_i, f_x, f_y) \times \text{in}((f_i, x + k_x, y + k_y)). \tag{1}
\]

Pooling is aimed to lessen the size of network feature maps and decrease the computational complexity and the number of parameters while retaining the main features. The commonly used pooling method in CNN is maximum pooling. As shown in (2), this process calculates the maximum value of \( K \times K \) adjacent neurons in the feature map.

\[
\text{out}(f_o, x, y) = \max(\text{in}(f_i, x + k_x, y + k_y)). \tag{2}
\]

The fully connected layer is where to make classification, each of input neurons and output neurons is fully connected, and the values of output neurons are weighted sums of all input neurons, as shown in

\[
\text{out}(f_o) = \sum_{f=0}^{N_{if}} \text{wt}(f_o, f_i) \times \text{in}(f_i). \tag{3}
\]

4.2. CNN Computational Complexity Analysis. In the forward propagation calculation of single-layer network in CNN, we can estimate its time complexity by multiplication operations times [13]. In a convolutional layer, each convolutional kernel is a \( k \times k \) filter, which is applied to feature maps with the size of \( w \times h \). The number of convolutional kernels is \( N_{in} \times N_{out} \); therefore, the time complexity of a convolution layer can be expressed as

\[
C_{\text{CONV}} = O(N_{in} \times N_{out} \times k^2 \times w \times h). \tag{4}
\]

For the pooling layer, the number of input feature maps \( N_{in} \) and number of output feature maps \( N_{out} \) are equal, so the time complexity of the pooling layer and the fully connected layer can be

\[
C_{\text{Pooling}} = O(N_{in} \times w \times h),
\]

\[
C_{\text{FC}} = O(N_{in} \times N_{out}). \tag{5}
\]

From these equations, it can be clearly seen that the calculation of the convolutional layer in the CNN model occupies the primary position. Its time complexity is significantly higher than that of a fully connected layer. Therefore, we need to focus on how to accelerate the calculation of the convolution layer. Due to the unique structure of FPGA, the parallelism of computing can be realized through programming. We can consider realizing CNN parallel computing to reduce the time complexity and achieve the purpose of acceleration.

4.3. CNN Computing Parallelism Analysis. The core task of heterogeneous computing acceleration based on FPGA is to extract the parallelism characteristics of the network in the computing process. The CNN model consists of multiple layers. Hence, our first thought is to realize the parallelism of various layers [15]. However, in the calculation process of CNN, due to the transmission characteristics of data forward propagation, the data between adjacent layers have a strong correlation, which dramatically limits the computational parallelism between layers, making it difficult to achieve task parallelism between layers. Despite the low parallelism between different layers, there is high parallelism in the single-layer calculation. Moreover, each layer of CNN has a similar network structure, and the computation of each layer has a high degree of similarity. Therefore, the whole convolutional neural network can be achieved by completing the single-layer network design and then reusing these computation resources.

The procedure of CNN single-layer operations is shown in Figure 2, where IN_1, IN_2, \ldots, IN_n are the n input feature
maps in this layer, \( \text{OUT}_1, \text{OUT}_2, \ldots, \text{OUT}_n \) are \( m \) output feature maps in this layer, and \( K_{11} - K_{mn} \) are \( n \times m \) convolution kernels. Different layers in CNN have different values for \( n \) and \( m \), and each feature map is associated with the corresponding convolutional kernel \( K_{mn} \), performs convolution, accumulates the results, and adds the offset value, and then obtains the output of the convolutional layer through the nonlinear activation function sigmoid. The output will be used as the input of the pooling layer. After pooling, the output feature map of the layer will be obtained. In this process, CNN’s single-layer computing process has high parallelism, which can be divided into the following [16].

4.3.1. Parallelism in the Convolutional Kernel. When a single convolutional kernel convolutes the input feature map, due to the independent product of each pixel, the \( K \times K \) th product and accumulation operation in a single convolution window can be carried out in parallel, as shown in Figure 3. The parameters \( K_{mn} - W_{xy} \) represent the weight at position \((x, y)\) in the \( n \times m \) convolutional kernels.

4.3.2. Parallelism between Convolutional Kernels. For multiple pixels on the same input feature map, the identical convolutional kernels can execute in parallel, and multiple neurons of the output feature maps can be obtained at the same time, as shown in Figure 4.

4.3.3. Inter-Layer Parallelism. Figure 5 shows that each neuron in output feature maps corresponds to the product accumulation result of \( n \) input feature maps and various convolutional kernels. For the exact position in these distinct input feature maps, the convolutional kernels are able to convolute in parallel.

4.3.4. Inter-Output Parallelism. The same convolutional window on the input feature map and different convolutional kernels can perform convolution operations in parallel to obtain the values of neurons at the same position of distinct output feature maps, as shown in Figure 6.
By combining the above four parallel characteristics of convolution computing, we can exploit the programmability of FPGA hardware to realize the ability of parallel convolution computing and achieve a better acceleration effect.

5. FPGA Parallelism Design

5.1. Parallel Design and Analysis of FPGA Computing Unit. There are multi-level nested loops in the convolutional module of CNN, and hundreds of convolution operations need to be carried out. If the design of parallel operation is not adopted, its time complexity will be inestimable. Generally, FPGA acceleration will consider how to optimize the loop structure. When each loop is relatively independent, the loop can be expanded to improve the degree of parallelism. Because there are many product and accumulation operations in the convolutional layer, the higher the degree of parallelism we achieve, the fewer the clock cycles, and the convolutional layer needs to complete a convolution.

However, in the design, choosing different levels of loop expansion may produce different results. Further, to what extent the expanded execution units share data which affects the complexity of the generated hardware and finally impacts the number of replicas and hardware operation frequency. Firstly, we need to use the parallelism between CNN outputs to divide the convolution calculation of multiple nested loops and apply the method of loop tiling and loop expansion to construct the calculation circuit directly used in the CNN layer. Loop tiling technology transforms the program with nested loops, reconstructs the code and tiles, and reorders the iteration space to improve the data locality and block parallelism [17]. The loop expansion technology makes use of the parallelism between convolutional kernels to carry out multiple convolutional parallel processing for the multi-input and one-output feature extraction process of the loop in the tile. This method is conducive to reducing the execution time of extracting a feature map.

5.2. FPGA Overall Framework Design. Based on the previous work [6–10], the general framework of parallel computing of convolutional neural network based on FPGA can be summarized as follows: considering forward propagation, the design division in the overall architecture is determined by using the parallel computing characteristics of convolutional neural network and loop transformation. The system framework is divided into different hierarchical structures according to the characteristics of parallel and serial processing [18]. Figure 7 shows the hardware architecture of the CNN. The subsystem of the overall architecture mainly includes input cache module, weight data
storage module, convolution module, pooling module, fully connected layer, activation function module, and control module.

The control module controls the transmission sequence of data streams and communication between modules. Firstly, the control logic generates a data request signal. Under the control of the signal, the input data stream is synchronized with the system clock and updated after each cycle. Then, the input data are sent to the functional modules of each parallel or serial subsystem to complete the operation of several feature maps. Finally, after consequential feature extraction in each layer, feature classification combines multiple local feature data streams into global features and sends them to the fully connected layer module for classification.

6. Framework Verification and Result Analysis

The CNN heterogeneous accelerator based on FPGA discussed in this paper was implemented by OpenCL heterogeneous computing standards and aimed to complete the hardware acceleration of CNN forward propagation computing. This section tests and verifies the accelerator by using the parameters of the trained AlexNet model and images from the ImageNet dataset and then analyzes the experimental results and the performance of the FPGA accelerator system.

The accelerator function verification mode designed in this paper is CPU+ FPGA. The required functions are designed on FPGA and arranged into an accelerator structure in line with CNN operation. The operating system software of the computer sends the parameters required by the model and the image data to be processed to the FPGA accelerator via the PCIe bus. Afterward, the accelerator uses the PCIe bus again to send its running results back to the computer. The corresponding system will display running results and classification outcomes, and compare them with the benchmark solution to evaluate the performance of the accelerator. Figure 8 shows the overall structure of the FPGA accelerator.

The detailed steps are as below: firstly, the input data (mainly images) will be stored on the PC, and the reverse training of the neural network model has been completed on the computer. Then, the input data and model parameters will be directly passed to the FPGA with the PCIe bus.

After that, the controller of FPGA will carry out the mission to analyze the input data and AlexNet model parameter input by PC and keep them in external memory (LDDR3). When the control layer receives control instructions from the PC, the corresponding modules of the various kernels and output layers are executed to complete the computations of the neural network. The output result is returned to the PC when the forward propagation calculation is completed.

6.1. Methodology. We selected the FPGA hardware, the Xilinx Kintex-7 development board XC7K325T as our experimental platform, and OpenCL as the development environment. The experimental environment and the configuration information of the devices we used are listed in Table 2.
6.1.1. Model Parameter Acquisition. Here, we used the trained model CaffeNet under the Caffe framework, which is the AlexNet model [25, 26]. The design of the whole system was based on this model structure. The dataset used for model training is ImageNet [27]. Once the Caffe deep learning framework is installed, MATLAB has been used to extract the parameters of the network model.

6.1.2. Input Image Acquisition and Preprocess. The images used in the test are from the ImageNet image dataset. This image dataset and corresponding category tag file set_words.txt can be downloaded from the relevant website. The image needed to be preprocessed before testing, including image size adjustment and demean processing. The required image size was $227 \times 227$, and these preprocessing operations were processed by the script provided by the Caffe framework. The test method of the FPGA accelerator is to select a certain number of pictures from the ImageNet dataset for classification and record the accuracy of program classification and execution time to verify the accuracy and performance of the accelerator.

6.1.3. Data Transmission Process. In PC platform, data are stored in the inner memory and transferred automatically through data buses, which means CPU and GPU do not require extra data manipulation, but the situation for FPGA is different, and the time consumption of data transmission to FPGA board memory via the PCIe bus is essential. To eliminate these cost overheads, we implemented the scheme in which data processing and transmission take place simultaneously. It is admitted that this manipulation would make an impact on the execution performance but limited to the current architecture, most of FPGA board does not have an integrated storage, and thereby, the cost of data transmitting is deemed as unavoidable. It is noteworthy that we applied PCIe bus to complete the mission, which most devices are not equipped with, and the alternative way to do is through LAN or Thunderbolt 3.

6.1.4. Software Implementation. In our work, OpenCL [28] was chosen to develop the FPGA accelerator mainly because of the great convenience and efficiency it brought. The OpenCL programs could be divided into two parts, one for the PC and the other for the FPGA board. The part in the PC took responsibility for the controlling of the whole program, and the other oversaw the completion of allocated computing tasks. To achieve that, four main functions in the CNN model were rewritten as the strategy mentioned above, including memory reading/memory writing, conventional kernel, and pooling.

6.1.5. Measurement Method. To compare the performance of different devices, we exploited various aspects of variables. System commands like nvidia-smi or ps were utilized to measure the behaviors of CPU and GPU. Vivado [29] was also deployed, helping monitor the FPGA board. With the hardware performance statistics, throughput was strictly calculated by the general formula. By contrast, MATLAB was employed to run the original AlexNet model as the controlled group. The accuracy of model on distinct devices was assessed automatically by the output part of the programs.

6.2. Result Analysis. The CNN heterogeneous accelerator structure proposed in this article achieves the image classification function, but the batch processing operation of
images is not considered in the design procedure. Hence, only a single image can be classified at a time. After classifying 100 test images (randomly chosen) in the dataset, the accurate rate reached 54%, which was basically consistent with the standard results in Caffe. To evaluate the acceleration performance of the system, an experimental comparison was carried out on three platforms: FPGA, CPU, and GPU. The batch size was set to 1 to keep the conditions consistent. The test results under the three platforms were compared, as shown in the following table. The energy efficiency ratio in Table 3 refers to the ratio of speedup to power. The listed power is the rated power used for comparison purpose, and it may be different from the actual power.

In Table 3, it can be found that for the same input image data, GPU has the best performance of 4.3 s but also with the highest power consumption of 120.0 W. Compared with the CPU, the speedup of our FPGA accelerator was 8.67, and the energy efficiency was 3 times that of GPU and 157 times that of CPU. Moreover, our implementation achieved almost double throughput as much as that of the previous work [6] and even with lower energy consumption and higher energy efficiency.

Although GPU had the shortest running time and the highest speedup, the energy consumed was also the maximum. With the same power supply, FPGA tends to have a better performance than GPU. In terms of energy efficiency, the heterogeneous acceleration implementation of CNN based on the FPGA platform discussed above was better than GPU, which also implied that FPGA performed extremely well on the balance of performance and energy consumption.

### 7. Limitations and Path Forward

Although our design of deep learning accelerator is innovative and progressive, there are some limitations in our work. The section presents such limitations and puts forward possible future works.

#### 7.1. Data Transmission

Up until now, unlike conventional computers, most FPGA boards do not merely integrate external storage. Hence, when it requires manipulating a vast dataset, the FPGA board cannot store all of them in advance but receives each piece of it successively, which essentially led to slow down the operation speed. Hence, in the future, with the further shift in the data transmission mode, it may produce better performance because time consumption on PCIe would likely to be demolished.

#### 7.2. Performance Measurement

In the experimental evaluation, we evaluated the performance with the support of other software, and it was manifested that the software is also supposed to occupy some resources more or less. Therefore, for the future work, it is suggested that the assessment of performance should be carried out in a more controlled and sophisticated environment such as with external gauges or appropriate virtual machines [30].

#### 7.3. Experimental Implementation and Dataset Selection

To make a clear comparison between different schemes, we opted to use the same FPGA board as in the previous work [6]. Generally, replicating the hardware configurations poses difficulty as some of the resources may not be available in the market. Nevertheless, it is necessary to take the progress of FPGA architectures into consideration when measuring the speedup brought by different methods. Additionally, our work is limited to one dataset where a bias may exist. In the future work, it is suggested to utilize multiple datasets in order to further improve the comparison reliability [31].

### 8. Conclusion and Future Prospects

Several accelerator approaches for CNN computation based on the field-programmable gate array (FPGA) platform have shown great progresses in many areas because they have edges of high parallelism, low power consumption, and better programmability. In our research, we illustrated some important concepts of parallel computing (PC) in convolution algorithm and figured out how to realize parallel computing in FPGA hardware structure.

On the basis of theoretical analysis, we applied the verification model of FPGA combined with PC to verify the acceleration function of FPGA and compared the performance of FPGA with traditional computing processors in terms of operation efficiency and energy consumption ratio. The accelerator realized a complete forward propagation process of CNN, and its convolution unit combined multiple parallel computing operations to effectively improve the overall performance.

In order to verify the advantages of FPGA over traditional processors, we compared the performance against traditional computation processors related to the operation

| Platform configuration | GPU | GPU | FPGA [6] | FPGA (proposed) |
|------------------------|-----|-----|----------|----------------|
| Data type              | Fp32 | Fp32 | Fix16    | Fix16/Fp32     |
| Clock frequency (MHz)  | 1845 | 1845 | 41.32    | 76.19          |
| Execution time (s)     | 176.2 | 3.9  | —        | 20.3           |
| Energy consumption (W) | 65   | 120  | 25.6     | 23.3           |
| Throughput (GOPS)      | 1359 | 171.4| 41.32    | 76.19          |
| Energy efficiency (GOPS/w) | 0.0209 | 0.978 | 1.65     | 3.27           |
| Speedup                | —   | 40.98| —        | 8.67           |
efficiency and energy consumption ratio. By combining the parallelism of CNN algorithm, FPGA significantly showed a better energy consumption ratio and efficiency than traditional CPU and GPU. For the same input image data, G GTX 1660 Ti Ultra 6G-GPU showed the best performance of 4.3 s but also with the highest power consumption of 120.0 W. Compared with the Intel i5-10400F-CPU, the speedup of our FPGA accelerator was 8.67, and the energy efficiency was 3 times that of GPU and 157 times that of CPU. At the clock frequency of 1818mhz, the peak computing rate of FPGA can reach 76.19 GOPS.

Although FPGA has made some achievements in accelerating deep learning with its advantages of reconfigurability and low energy consumption, it also has some disadvantages such as hardware programming difficulties and considerable time cost of the reconstruction process. Therefore, FPGA still has a long way to go to reach the stage of more widely used deep learning accelerators. As per the current development progress, some promising research directions can be outlined as follows:

8.1. Activation Function Optimization. At present, most of the research focuses on the repetition part of matrix operation in FPGA computing optimization [5, 6], but few on the improvement of activation function optimization, so this is a field to be excavated.

8.2. Data Optimization. The use of fixed-point data with fewer bits can improve the acceleration performance of the model, but it also brings the disadvantage of low precision [8]. Future research on dynamic precision data quantization can be strengthened to make different layers of neural networks. It requires different formats of fixed-point numbers, finding a structure of fixed-point numbers with suitable bits, and maintaining the necessary minimum accuracy.

8.3. Data Path Optimization. Data path optimization can be divided into on-chip and off-chip data transmission. Xuechao et al. [32] used a systolic array structure to transmit data and reduced the access of computing units to on-chip cache, making the design reach a very high frequency. Compared with today’s popular accelerator GPU, the bandwidth of FPGA is limited, i.e., only 4–20GB/s in general. The optimization goal of most existing studies is to reduce the off-chip memory access bandwidth requirements.

8.4. FPGA Cluster. The performance of integrating multiple FPGA chips is very promising. However, how to deal with the processing scheduling and allocation between chips is still a significant challenge. Further research work may be initiated from various fine-grained divisions and weight allocations between multiple chips.

It can be predicted that deep learning has a broad development prospect in the future as a revolutionary implementation method of machine learning. Similarly, the deep learning acceleration technology based on FPGA is also expected to get better with each passing day and would finally promote the reform and development of the whole field of artificial intelligence.

8.5. New Deep Learning Model. The proposed design in this paper focused on the parallelism of CNN model. Although it was the most popular deep learning solution, the researchers are encouraged to explore other trending models too such as the transformer, proposed by Google [33]. The transformers have also shown better parallelism than CNN in many areas; therefore, the functionality of both FPGA and transformers should also be explored as it may lead to a new trend.

Data Availability

The data used to support the findings of this study are available at https://image-net.org/download.php.

Disclosure

Zhi Qi and Weijian Chen are the co-first authors.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was supported by Xiamen University Malaysia Research Fund (Grant no. XMUMRF/2022-C9/IECE/0033).

References

[1] F. X. Diebold, “What’s the big idea? “Big Data” and its origins,” Significance, vol. 18, no. 1, pp. 36-37, 2021.
[2] V. Sze, Y. H. Chen, J. Emer, A. Suleiman, and Z. Zhang, “Hardware for machine learning: challenges and opportunities,” in Proceedings of the IEEE conference on Custom Integrated Circuits Conference, April 2017.
[3] J. Kalomiros and J. Lygouras, “Design and evaluation of a hardware/software FPGA-based system for fast image processing,” Microprocessors and Microsystems, vol. 32, no. 2, pp. 95–106, 2008.
[4] T. Wang, C. Wang, X. Zhou, and H. Chen, “An overview of FPGA based deep learning accelerators: challenges and opportunities,” in Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1674–1681, Zhangjiajie, China, August 2019.
[5] Z. Chen, P. Li, G. Y. Sun, Y. J. Guan, B. J. Xiao, and J. S. Cong, “Optimizing fpga-based accelerator design for deep convolutional neural networks,” in Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays, pp. 161–170, Monterey, CA, USA, February 2015.
[6] H. M. Li, X. T. Fan, L. Jiao, W. Cao, X. G. Zhou, and L. L Wang, “A high performance FPGA-based accelerator for large-scale convolutional neural networks,” in Proceedings of the 2016 26th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–9, Lausanne, Switzerland, August 2016.
[7] L. Kalms and D. Gühringer, “Exploration of OpenCL for FPGAs using SDAccel and comparison to GPUs and multi-core CPUs,” in Proceedings of the 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–4, Ghent, Belgium, September, 2017.

[8] Y. Liang, L. Lu, Q. Xiao, and S. Yan, “Evaluating fast algorithms for convolutional neural networks on FPGAs,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 4, pp. 857–870, 2020.

[9] J. Gong and B. J. Xiao, “Minimizing computation in convolutional neural networks,” Artificial Neural Networks and Machine Learning, vol. 8681, pp. 281–290, 2014.

[10] A. Utku, O. Shane, C. Davor, C. L. Andrew, and G. R. Chiu, “An opencl™ deep learning accelerator on arria 10,” in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 55–64, Monterey, CA, USA, February, 2017.

[11] A. Shawahna, S. M. Sait, and A. El-Maleh, “FPGA-based accelerators of deep learning networks for learning and classification: a review,” IEEE Access, vol. 7, pp. 7823–7859, 2019.

[12] Z. Q. Liu, P. Chow, J. W. Xu, J. F. Jiang, Y. Dou, and J. Zhou, “A uniform architecture design for accelerating 2D and 3D CNNs on FPGAs,” Electronics, vol. 8, no. 1, p. 65, 2019.

[13] L. Q. Lu, S. Zheng, Q. C. Xiao, D. M. Chen, and Y. Liang, “Accelerating convolutional neural networks on FPGAs,” SCIENTIA SINICA Informationis, vol. 49, no. 3, pp. 277–294, 2019.

[14] S. Y. Lu, Z. H. Lu, and Y. D. Zhang, “Pathological brain detection based on AlexNet and transfer learning,” Journal of computational science, vol. 30, pp. 41–47, 2019.

[15] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, “A 240 G-ops/s mobile coprocessor for deep neural networks,” in Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 696–701, Columbus, OH, USA, June, 2014.

[16] W. Wang, K. L Zhou, Y. C. Wang, G. Wang, Z. L. Yang, and J. Yuan, “FPGA parallel structure design of convolutional neural network (CNN) algorithm,” Microelectronics & Computer, vol. 36, no. 4, pp. 57–62, 2019.

[17] S. Liu, W. G. Wu, B. Zhao, and Q. Jiang, “Loop tiling for optimization of locality and parallelism,” Journal of Computer Research and Development, vol. 52, no. 5, pp. 1160–1176, 2015.

[18] L. Zong-ling, W. Lu-yuan, Y. Ji-yang, C. Bo-wen, and H. Liang, “The design of lightweight and multi parallel CNN accelerator based on FPGA,” in Proceedings of the IEEE conference on Information Technology and Artificial Intelligence Conference (ITAI), pp. 1521–1528, Chongqing, China, August, 2019.

[19] intel, “Intel® Core™ i5-10400F Processor,” 2020, https://www.intel.sg/content/www/xa/en/support/programmable/support-resources/design-guidance/opencl-bsp-support.html?countrylabel=Asia%20Pacific.

[20] msi, “MSI Z590-A PRO,” 2020, https://www.msi.com/Motherboard/Z590-A-PRO.

[21] kingston, “HyperX Savage 8GB DDR4 2400,” 2020, https://www.kingston.com/en/company/press/article/7828.

[22] colorful, “iGame GeForce GTX 1660 Ti Ultra 6G,” 2020, https://en.colorful.cn/product_show.aspx?mid=102&sid=1749.

[23] xilinx, “Xilinx Kintex-7 FPGA,” 2018, https://www.xilinx.com/products/silicon-devices/fpga/kintex-7.html.

[24] ubuntu, “Ubuntu 16.04,” 2018, https://ubuntu.com/16-04.