# An FPGA-Based Resource-Saving Hardware Accelerator for Deep Neural Network

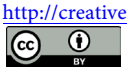
**Han Jia, Xuecheng Zou**

School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China
Email: jhjay2006@gmail.com

**How to cite this paper:** Jia, H. and Zou, X.C. (2021) An FPGA-Based Resource-Saving Hardware Accelerator for Deep Neural Network. *International Journal of Intelligence Science*, **11**, 57-69.
https://doi.org/10.4236/ijis.2021.112005

**Received:** January 12, 2021
**Accepted:** March 27, 2021
**Published:** March 30, 2021







## Abstract

With the development of computer vision researches, due to the state-of-the-art performance on image and video processing tasks, deep neural network (DNN) has been widely applied in various applications (autonomous vehicles, weather forecasting, counter-terrorism, surveillance, traffic management, etc.). However, to achieve such performance, DNN models have become increasingly complicated and deeper, and result in heavy computational stress. Thus, it is not sufficient for the general central processing unit (CPU) processors to meet the real-time application requirements. To deal with this bottleneck, research based on hardware acceleration solution for DNN attracts great attention. Specifically, to meet various real-life applications, DNN acceleration solutions mainly focus on issue of hardware acceleration with intense memory and calculation resource. In this paper, a novel resource-saving architecture based on Field Programmable Gate Array (FPGA) is proposed. Due to the novel designed processing element (PE), the proposed architecture achieves good performance with the extremely limited calculating resource. The on-chip buffer allocation helps enhance resource-saving performance on memory. Moreover, the accelerator improves its performance by exploiting the sparsity property of the input feature map. Compared to other state-of-the-art solutions based on FPGA, our architecture achieves good performance, with quite limited resource consumption, thus fully meet the requirement of real-time applications.

## Keywords

Deep Neural Network, Resource-Saving, Hardware Accelerator, Data Flow

## 1. Introduction

To meet the demands of computer vision applications (e.g., object recognition and classification), numerous deep convolutional neural network models have

emerged and achieved overwhelming performance [1] [2] [3]. Along with the development of deep learning algorithms, a broad variety of applications in the Internet of Things (IoT) rise. However, state-of-the-art DNNs based on deep network models put forward requests for enormous computation and memory access, which lead to great latency and computation resource consumption. Hence, hardware acceleration solutions with real-time processing become an increasingly urgent demand. Due to the intensive computation and huge external data access for the DNN algorithm, CPU processors are unable to meet real-time demands. Thus, the specialized accelerator for DNN model algorithms arouses great research interest. The DNN algorithm can be accelerated during both the training and inference phases. In this paper, we focus on the inference phase, which is widely used in embedded vision system.

Enormous multiply and accumulate (MAC) operations and a great number of parameters are the two main bottlenecks for DNN acceleration solutions. To solve this, researchers have been focused on application and specific integrated circuits (ASIC) [4] [5] [6] and FPGA [7] [8] [9] [10] [11]. Most researches have been mainly focused on instruction flow and data flow architectures. Among the researches, data flow architecture has become a key research area due to its high parallelism property. Several recent hardware accelerating solutions have achieved great performance. For example, tensor processing unit (TPU) [5] is a typical data flow based architecture, which accelerates the convolutional operations by utilizing systolic matrix multipliers. Compared to contemporary Graphics Processing Unit (GPU) and CPU, TPU achieves approximately 15× - 30× speedup. Apart from TPU, Eyeriss [4] exploits reuse strategy of input feature map and weight filters by proposing the row stationary (RS) data flow, which helps minimize energy consumption caused by the on-chip data movement and achieves high energy efficiency. Based on above excellent researches, data flow architecture shows great potential in DNN acceleration. In this paper, the proposed architecture also concentrates on data flow architecture.

Main target of this paper is to deal with computation and memory resource limitations in real-life applications, further reduce resource consumption comparing to existing works is the main research point, along with applying data reuse technique as the acceleration method.

In this paper, a novel resource-saving architecture is proposed to deal with acceleration demands of DNN model, and limited calculating resource consumption is also the main research point for further real-life applications such as IoT devices. Compared to previous research works, PE array of the proposed architecture can be reused to implement $1 \times 1$ or $3 \times 3$ convolution operations, thus resource consumption can be greatly reduced. On-chip buffer allocation strategy and corresponding data reuse technique is proposed, to reduce the data access between the on-chip buffer and double data rate (DDR). What's more, by utilizing the sparsity property of DNN, an effective utilization method is also introduced. Thus, a high-performance accelerator solution with limited resource

consumption is designed. This paper is organized as follows. Section 2 introduces the targeted network model. The proposed novel resource-saving DNN accelerator solution is presented in Section 3. Section 4 gives out the experimental results on FPGA with targeted DNN model and compares with state-of-the-art architectures. Section 5 gives a conclusion.

## 2. Targeted Network-VGG16

K. Simonyan and A. Zisserman from the University of Oxford proposed VGG-16 in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [12]. VGG-16 is a DNN model which achieves 92.7% top-5 test accuracy in ImageNet, an extremely challenging dataset consists of over 14 million images belonging to 1000 classes. Due to its state-of-the-art performance in various datasets, VGG-16 has become one of the most famous and widely studied models. Compared with AlexNet, another classic network model, VGG-16 makes great improvement by replacing large kernel-sized filters (11 × 11 and 5 × 5 in the first and second convolutional layer, respectively) with multiple 3 × 3 kernel-sized filters one after another. Training progress of VGG-16 takes weeks utilizing NVIDIA Titan Black GPU's. The architecture of VGG-16 is depicted below in **Figure 1**.

The input of VGG-16 to Conv1 layer is of fixed size 224 × 224 RGB image. The image is processed through a stack of convolutional layers, where the filters are with a very small 3 × 3 receptive field. Stride of convolutional layers is fixed to 1. To preserve spatial resolution after convolution operations, spatial padding is applied to convolutional layer input. For 3 × 3 convolutional layers utilized by VGG-16, padding is set to 1 pixel. In VGG-16, five max-pooling layers are introduced and follow some of the convolutional layers (not all the convolutional layers are followed by max-pooling). Max-pooling of VGG-16 is performed over a 2 × 2 kernel, with stride of 2.

At the end of convolutional layers follows three fully-connected (FC) layers. First two FC layers have 4096 channels each, and the third one with 1000. The final layer is a soft-max layer and then outputs the final result.

Rectified linear unit (ReLU) is implemented with all hidden layers as activation function to realize non-linearity. It is noticeable that Local Response Normalisation (LRN) is not included, as it leads to memory consumption and computation time increase, with quite limited performance improvement.

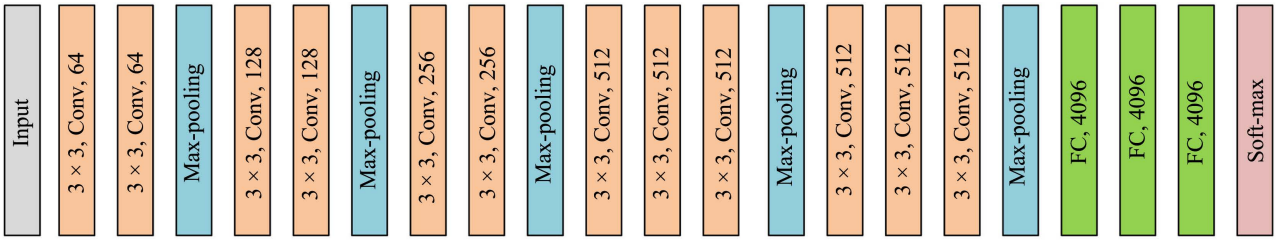



**Figure 1.** VGG-16 network architecture.

It has been demonstrated that the representation depth is beneficial for the classification accuracy. Utilizing conventional convolutional network architecture with substantially increased depth, state-of-the-art performance on the challenging ImageNet dataset can be achieved. Thus, VGG-16 network is a quite classic architecture for image classification. However, along with the benefits brought by deeper network architecture, comes with great amount of parameter and MAC operations. Moreover, utilization of FC layers also brings in enormous amount of parameters. Hence, hardware acceleration solution for VGG-16 is quite urgent for the implementation. Luckily, the single-column structure and simple kernel size variation make VGG-16 friendly for hardware implementation. Therefore, VGG-16 network model is widely used for evaluation of FPGA-based accelerators solutions. The proposed work is also based on VGG-16 network architecture.

## 3. Novel Resource-Saving DNN Accelerator Architecture

### 3.1. Architecture Overview

The proposed data flow based resource-saving architecture is shown in Figure 2. Convolutional operations are conducted in special designed PE array, which can be reused for 1 × 1 and 3 × 3 convolution. In the proposed architecture, the image buffers and weight buffers are directly linked to the PE. Computational results of PE array are streamed into the special function buffer.

The batch normalization (BN), activation and pooling operations are streamed and conducted in the special function buffer, and hence minimize data access between on-chip buffer and DDR. The special function buffer is composed of many static random-access memory (SRAM) banks, which allows it to receive computed results from the PE array in parallel.

### 3.2. Reusable PE Array

According to previous researches, PE architecture plays crucial role in DNN

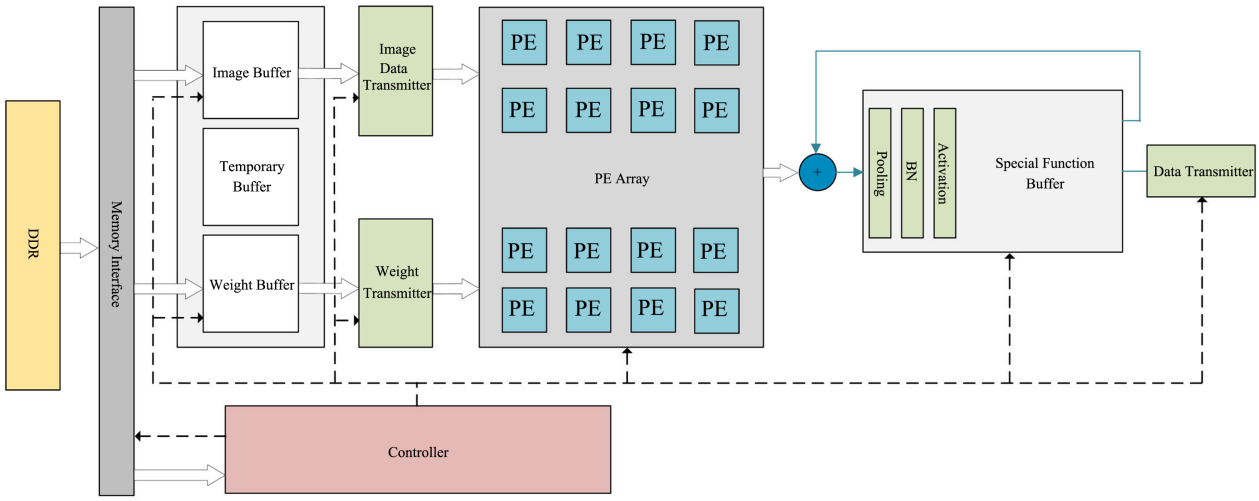



Figure 2. Resource-saving accelerator architecture.

hardware accelerator, as it determines resource consumption and throughput of the entire architecture. Most recent works are based on two types of PE architectures to realize convolutional operations: spatial 2D-PE array, and systolic matrix. Research work of TPU has demonstrated that systolic matrix is not highly efficient dealing with small layer size [5], while the spatial 2D-PE array performs much better. Hence, making the tradeoff between throughput, complexity and resource consumption, a novel spatial 2D-PE array is designed.

Different from other normal PE designs employed by recent research works that contain only one multiplier per PE [5] [6], the proposed PE consists of nine multipliers. Such novel design explores the reuse technique of PE array to conduct 1 × 1 and 3 × 3 convolution, which perfectly suits the VGG-16 network architecture. Figure 3 exhibits the proposed PE structure.

Apart from the nine multipliers, nine weight registers and nine image registers are linked to the multipliers. It is worth noting that the image registers are designed and connected in shift register mode, and transmits image data between PEs. Besides, enable signals are linked to multipliers to decide work status of them, to reduce energy consumption. Multipliers' outputs are set to zero for those are disabled.

With the special design above, two types of results can be outputted from one PE: sum of nine multipliers results, or nine separate results. Therefore, either one 3 × 3 convolution or nine 1 × 1 convolution operations can be conducted in the proposed PE structure. Moreover, with multiple PEs, bigger kernel size convolutions (Such as 5 × 5, 7 × 7, even 11 × 11) can be implemented with this architecture, hence improves the flexibility and scalability of the proposed architecture, more works to deal with various model architecture will be done in the future. Implementation of 5 × 5 and 7 × 7 convolution operations is shown in Figure 4. Peak multiplier utilization of proposed PE array when conducting 1 × 1

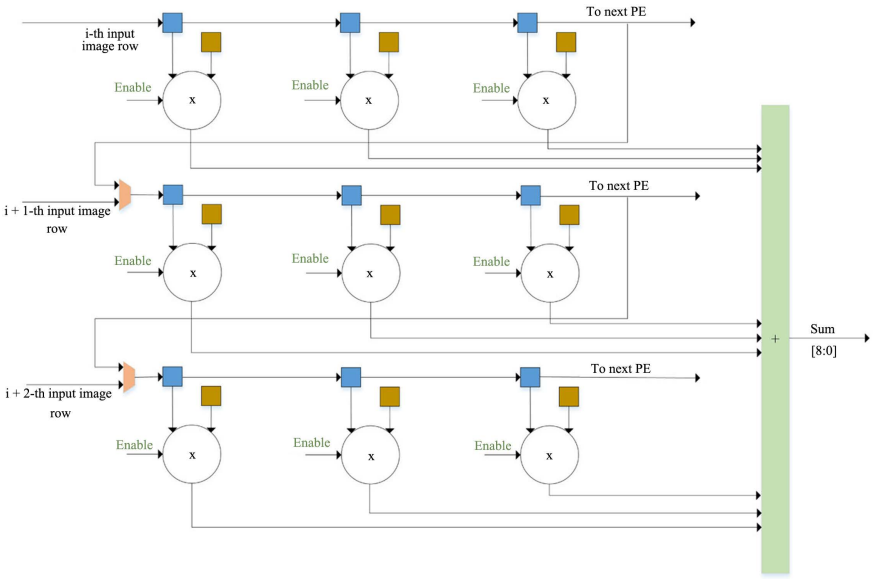



Figure 3. Reusable PE structure.

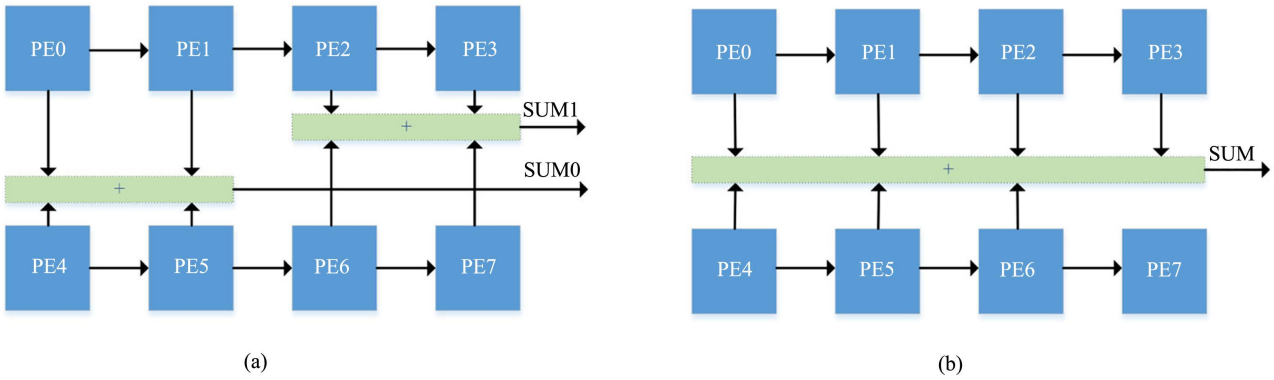



**Figure 4.** Convolutional operation configuration. (a) 5 × 5; (b) 7 × 7.

and 3 × 3 convolution operations are 100%, which ensures the full implementation efficiency of computation resources. For larger size kernels, certain percentage of multipliers are idle during calculation, however, such operations only occupy small portion of whole network, the waste can be considered tolerable, more details will not be discussed in this paper.

In the research work of Eyeriss, to reduce on-chip & off-chip data access, reuse technique of input feature map rows is introduced. By exploiting this reuse technique, on-chip energy cost is greatly lessened. During the computation of convolution operations, there are two vital parameters: stride and padding. For stride, it exhibits horizontally or vertically slides range of kernel. And padding is to add certain number of pixels with zero value at input images' edges. According to kernel size and input feature map size, different strategies that using padding or not can be utilized.

For the proposed architecture, similar row-reused method is implemented as shown in **Figure 5**, as 3 × 3 convolutional kernel with stride of 1 occupies most of VGG-16 convolution operation. As **Figure 5** presents, input data stored within on-chip buffer are organized in two-dimensional mode, and each row of input data can be reused multiple times. For instance, Row1 is reused twice, other rows such as Row2 can be reused even three times. Apart from image row data reuse, number of SRAM banks can be significantly reduced with the proposed technique, thus lessen complexity of the design work.

In the VGG-16 network model, three FC layers are employed apart from the convolutional layers. FC layers can be taken as special convolutional layers, with 1 × 1 input feature map, 1 × 1 kernel size, padding of 0 and stride of 1. For FC layers with small weight parameter size, such strategy is acceptable. However, FC layers in VGG-16 bring in enormous amount of weight parameter, and using this strategy would result in extensively increasing data access time. Thus, another special technique is used and will be narrated in the rest of paper.

### 3.3. Novel Designed On-Chip Buffers

For embedded implementation of DNN models, on-chip memory resources are usually quite limited. In addition to the on-chip memory intensity, data access

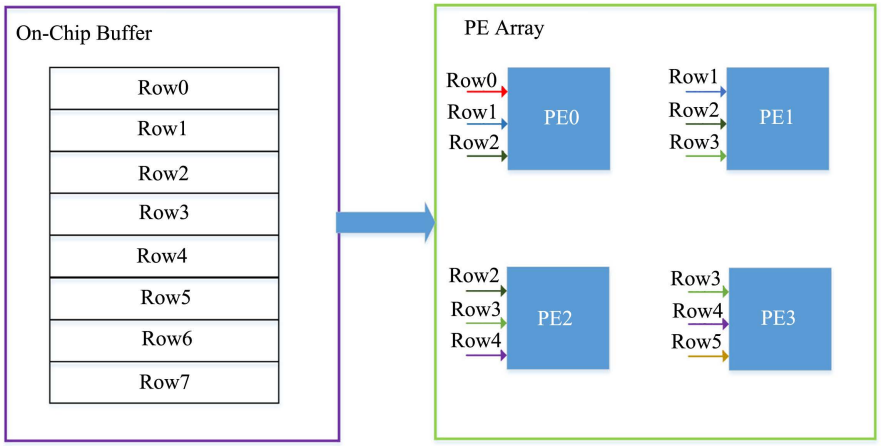



**Figure 5.** Row data reuse strategy.

between on-chip memory and external memory consumes great amount of energy, which is orders of magnitude greater than energy consumed by MAC operations [5]. Thus, data reuse techniques are frequently utilized to solve these problems [6]. For a certain layer, two ideal reuse situations exist: store all input feature map data on-chip; store all weight parameters on-chip.

In VGG-16 network model, Proportion of weight parameter and image data greatly varies among different convolutional layers. As **Figure 6** presents, dominant parameter changes from image data to weight parameter as layers go deeper.

According to the two ideal reuse strategies mentioned above, all weight parameters or input image data only need a one-time load. For implementation platform with sufficient on-chip memory resource, implement such strategies can be quite ideal. However, in most cases, especially for VGG-16 network model with great amount of parameters, on-chip memory resources are insufficient. Therefore, as **Figure 7** shows, the proposed work allocates limited on-chip memory resources into image buffer, weight buffer, temporary buffer and special function buffer.

Based on the property of VGG-16 network model mentioned above, by allocating the temporary buffer to image buffer or weight buffer, the proposed work flexibly selects whether to load input image data or weight parameter as much as possible. Specifically, for shallower layers where input image data occupies much greater amount, weight parameter will be loaded on chip for the best possibility; while for deep layers where amount of weight parameters is much greater, strategy is to load input image data on chip. Off-chip data access can be greatly lessened with this strategy combining the data reuse strategy proposed above. There surely exist circumstances that none of these ideal reuse situations can be implemented. For such instances, parameter with less proportion will be chosen and split to load on-chip. This inevitably leads to data reload and extra energy consumption.

With the implementation of special function buffer, special functional operations such as Pooling, Average-pooling [13] and Eltwise [14] are conducted

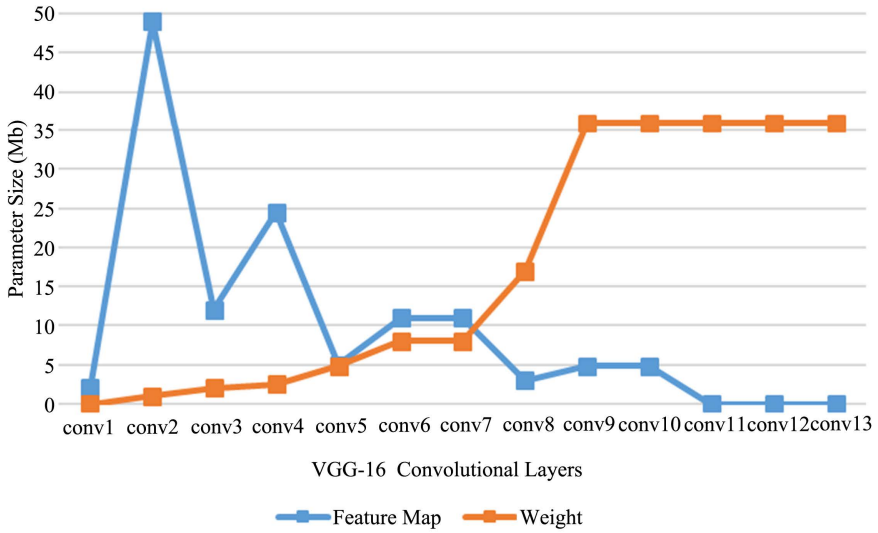



Figure 6. Parameter proportion of different convolutional layers in VGG-16.

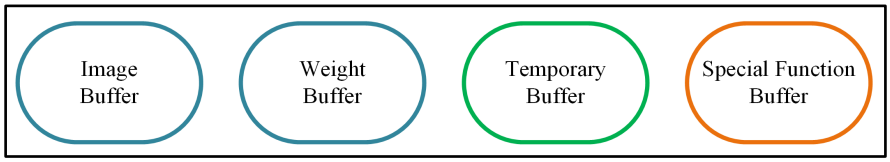



Figure 7. On-chip buffer allocation.

within on-chip buffer. This design helps the proposed accelerator solution realizes complete DNN acceleration except for convolution acceleration.

### 3.4. Optimizing Technique Using Sparse Prosperity

Based on previous researches, proportion of zero value in input feature map rises as the layers go deeper in VGG-16 network model. Figure 8 explains this property in detail. The introduction of ReLU activation leads to this sparse prosperity [13] [14]. Hence, a novel optimization strategy is introduced based on the sparsity, and both convolutional layers and FC layers are taken into consideration.

In the proposed work, for the convolutional layers, when input image data of the multiplier is "0", multiplication operation is skipped. Utilizing such strategy helps reduce great amount of multiplication operations, and further leads to significant reduction of on-chip energy cost.

Except for the convolutional layers, sparseness also occupies a great portion in FC layers. What's more, weight parameters take great proportion in FC layers. As Figure 9 shows, weight parameter proportion can be up to 89.9%. Thus, loading weight parameter of FC layers leads to long loading time and high energy consumption. In most common cases, to compress amount of parameters, FC layers are usually pruned. In the proposed work, through combining the sparse prosperity of VGG-16, a novel optimization strategy is introduced for the FC layers.

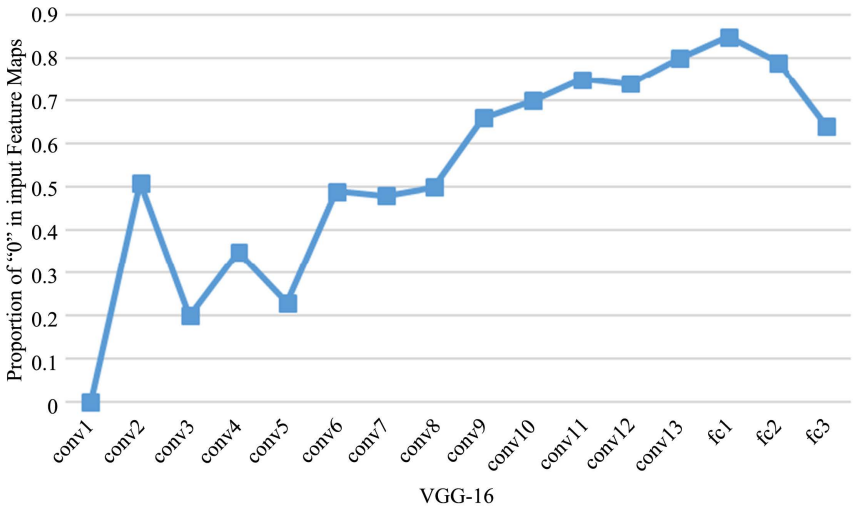



**Figure 8.** Sparseness of the input feature map of each layer in VGG-16.

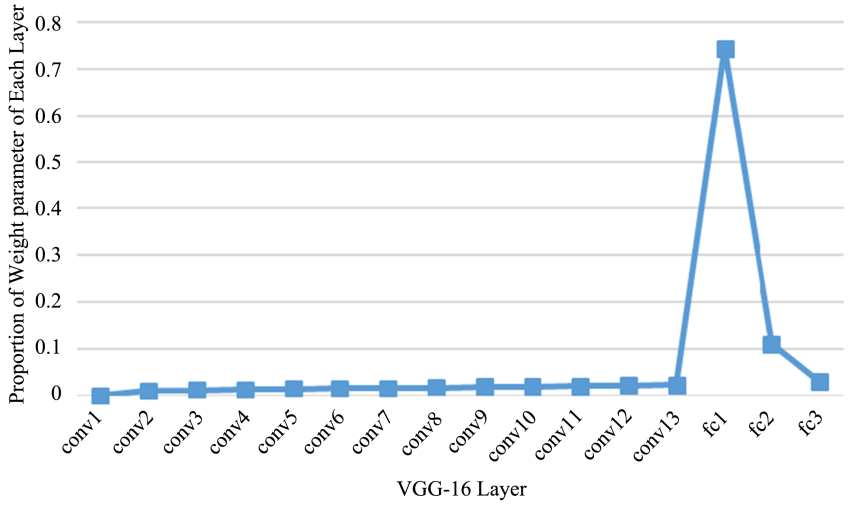



**Figure 9.** Weight parameter proportion of each layer in VGG-16.

For FC layers, Equation (1) can be utilized for the calculation.

$$\left[R_0,\cdots,R_{m-1}\right]=I_0*\left[W_{0,0}\cdots W_{0,m-1}\right]+\cdots+I_{n-1}*\left[W_{n-1,0}\cdots W_{n-1,m-1}\right] \quad (1)$$

With Equation (1), one row of weight will be loaded to be multiplied by $I_i$ and get corresponding result $R_i$ in the implementation. To utilize the sparseness of VGG-16, for "0" valued input, corresponding row of weight parameter will not be loaded. Data access can be significantly reduced with this method.

Thus, a comprehensive optimizing technique using sparse prosperity of VGG-16 is introduced and implemented. Amount of data transmission between on-chip buffer and external memory can be greatly lessened.

## 4. Experimental Results and Analysis

To verify the performance of the proposed resource-saving DNN accelerator, the accelerator is implemented on Xilinx Zynq ZC706 platform.

As shown in Figure 10, AXI bus based on the chosen FPGA platform is employed for data exchange among programming logic (PL) and processing system (PS). When the system starts operation, PS sends the address of configuration instruction stored in DDR to the accelerator, and follows with a start signal. Utilizing DMA engine, the accelerator located in PL side then reads configurations and sets all hardware modules into corresponding status. And the accelerator then gets into working status by following the instructions. When the work progress comes to an end, a finished signal is sent from PL side to PS Side. In the last step, PS side reads the inference result from specific DDR address. As mentioned above, image data stored in the image buffer is organized as two-dimensional mode, thus the proposed work utilize a transfer buffer to change the one-dimensional data stored in DDR to two-dimensional data. Accordingly, the image buffer, weight buffer, and special function buffer are equipped with double-buffers operating in ping-pong mode, to overlap data transmission time with calculating time.

Details of accelerator architecture are shown in Table 1. And Table 2 introduces detailed parameter for the PE array to deal with different filter sizes in VGG-16.

Table 1. Accelerator parameter details.

| Parameter | Value |
|---|---|
| Number of PE in PE array | 16 |
| Number of image buffer banks | 18 |
| Number of temporary buffer banks | 18 |
| Number of special function buffer banks | 16 |

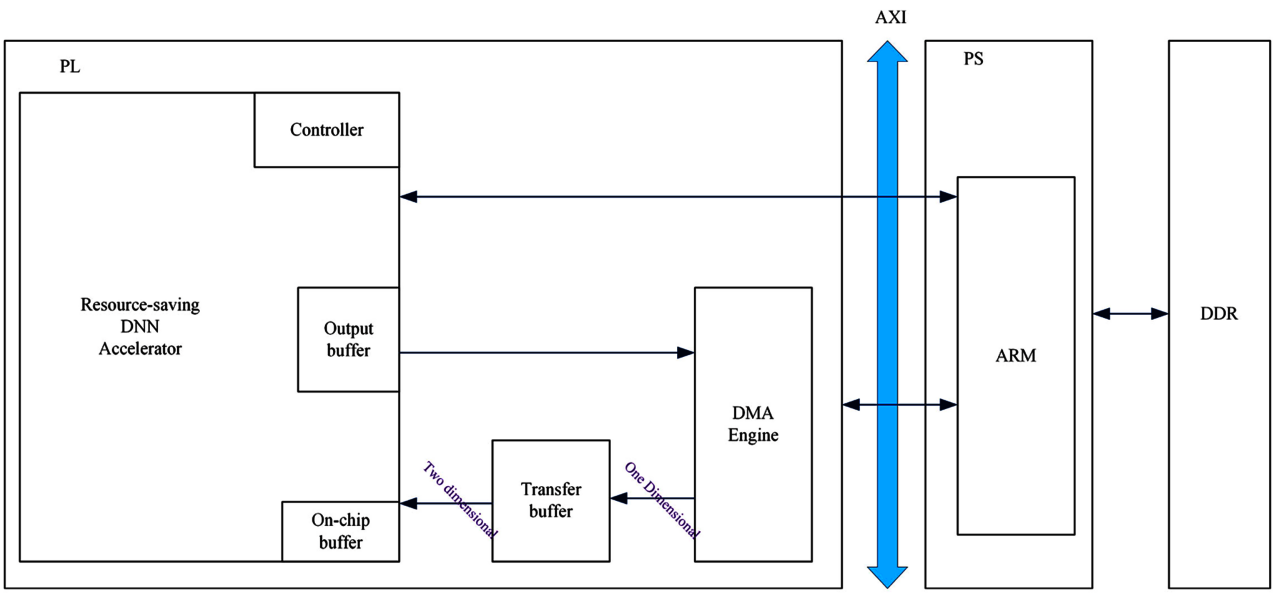



Figure 10. Accelerator implementation.

To evaluate the performance of the proposed architecture, performance density is employed to acquire relatively fair comparison between different architectures based on different FPGA platforms. For hardware accelerator implementations, throughput density (TD) works as performance density and is expressed as shown in Equation (2)

$$\text{TD} = \text{Throughput}/\text{clock} * \text{Multiplier Number} \tag{2}$$

In Equation (2), multiplier number is usually decided by the number of used DSP, while in some cases, one DSP can be taken as two multipliers. And the throughput can be calculated using Equation (3)

$$\text{Throughput} = \text{Total operation number}/\text{Run time} \tag{3}$$

Compared to other state-of-the-art FPGA-based accelerators, performance of proposed architecture on VGG-16 is shown in Table 3.

As the results above shows, compared to existing state-of-the-art accelerators, the proposed DNN accelerator solution achieves competitive performance. Moreover, the proposed work only consumes about one quarter of computation resource comparing to other researches. Good flexibility is also exhibited in the comparison due to its ability to deal with different kernel sizes. Thus, the proposed work contains great scalability for further research to cope with different network models. Specifically, work of Meloni's [9] shows good flexibility, however, the proposed work achieved better PD with significantly less calculation

**Table 2.** PE array parameter details.

| Kernel Size | Stride | PE array parameter | | | SRAM Banks |
|---|---|---|---|---|---|
| | | Feature Map Number | Channel Number | Image Rows | |
| 1 × 1 | 1 | 8 | 8 | 2 | 16 |
| 3 × 3 | 1 | 1 | 1 | 18 | 18 |
| | | 1 | 2 | 10 | 10 |
| | 2 | 1 | 2 | 17 | 17 |

**Table 3.** Performance comparison.

| | | Qiu [8] | Meloni [9] | Venieris [10] | Proposed Work |
|---|---|---|---|---|---|
| Platform | | Xilinx Zynq Z-7045 | Xilinx Zynq Z-7045 | Xilinx Zynq Z-7045 | Xilinx Zynq Z-7045 |
| Clock (Mhz) | | 150 | 140 | 125 | 140 |
| Precision | | 16-bit fixed | 16-bit fixed | 16-bit fixed | 16-bit fixed |
| DSP | | 780 | 864 | 855 | 172 |
| Throughput (GOP/s) | Conv | 187.89 | 169.7 | 155.81 | 35.43 |
| | Full | 136.97 | 122.58 | - | 28.31 |
| Throughput Density | Conv | $1.61 \times 10^{-3}$ | $1.40 \times 10^{-3}$ | $1.45 \times 10^{-3}$ | $1.44 \times 10^{-3}$ |
| | Full | $1.17 \times 10^{-3}$ | $1.01 \times 10^{-3}$ | - | $1.15 \times 10^{-3}$ |
| Flexibility | | Low | High | Low | High |

resources consumption. For the other two researches with limited flexibility: Works of Qiu [8] and Venieri [10], they achieve high performance due to their specific design for VGG-16 model, their flexibility is limited as re-designing of code needs to be done when dealing with different kernel sizes. Especially for Qiu's architecture, it exhibits state-of-the-art performance. Comparing to these high performance architectures, the proposed work still has competitive performance on PD, and consume significantly less computation resource.

## 5. Conclusion

This paper introduces a novel solution for DNN model acceleration with resource-saving method. A novel reusable PE structure is designed to deal with limitation of computational and memory resource in real-life applications, and achieves competitive performance with only one quarter of DSP consumption compared to existing work. In addition to the PE, on-chip buffer allocation is specially designed and significantly lessens data access between on-chip buffer and off-chip memory. Experimental results suggest such strategy and design help further enhance performance of the accelerator. Moreover, through combining sparsity of input feature map and weight proportion property of VGG-16, a novel comprehensive optimization strategy that takes both convolutional layers and FC layers into consideration is introduced. Through implementing the proposed architecture on FPGA platform, and comparing with existing state-of-the-art architectures based on FPGA, the proposed work achieves good performance and significant resource-saving prosperity in the meantime. For future works, due to high flexibility of the proposed architecture, implement different network models with enhanced accelerator architecture based on this work will be done.

## Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

## References

[1] Jiang, X., *et al.* (2020) Attention Scaling for Crowd Counting. 2020 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 13-19 June 2020, 4705-4714. https://doi.org/10.1109/CVPR42600.2020.00476

[2] He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2020) Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **42**, 386-397. https://doi.org/10.1109/TPAMI.2018.2844175

[3] Zhang, Y., Zhou, D., Chen, S., Gao, S. and Ma, Y. (2016) Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. 2016 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 27-30 June 2016, 589-597. https://doi.org/10.1109/CVPR.2016.70

[4] Chen, Y., Yang, T., Emer, J. and Sze, V. (2019) Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, **9**, 292-308.

https://doi.org/10.1109/JETCAS.2019.2910232

[5] Jouppi, N.P., Young, C., *et al.* (2017) In-Datacenter Performance Analysis of a Tensor Processing Unit. *Proceeding of* 2017 *ACM/IEEE* 44*th Annual International Symposium on Computer Architecture*, Toronto, ON, Canada, 24-28 June 2017, 12 p. https://doi.org/10.1145/3079856.3080246

[6] Shin, D., Lee, J., Lee, J., Lee, J. and Yoo, H.-J. (2018) DNPU: An Energy-Efficient Deep-Learning Processor with Heterogeneous Multi-Core Architecture. *IEEE Micro*, **38**, 85-93. https://doi.org/10.1109/MM.2018.053631145

[7] Zhang, C., *et al.* (2015) Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks. *Proceedings of the* 2015 *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, New York, NY, USA, February 2015, 161-170. https://doi.org/10.1145/2684746.2689060

[8] Qiu, J., Wang, J., Yao, S., Guo, K.Y., *et al.* (2016) Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. *Proceedings of the* 2016 *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, New York, NY, USA, February 2016, 26-35. https://doi.org/10.1145/2847263.2847265

[9] Meloni, P., Capotondi, A., *et al.* (2018) NEURAghe: Exploiting CPU-FPGA Synergies for Efficient and Flexible CNN Inference Acceleration on Zynq SoCs. *ACM Transactions on Reconfigurable Technology and Systems*, **11**, 24 p. https://doi.org/10.1145/3284357

[10] Venieris, S.I. and Bouganis, C. (2018) fpgaConvNet: Mapping Regular and Irregular Convolutional Neural Networks on FPGAs. *IEEE Transactions on Neural Networks and Learning Systems*, **30**, 326-342. https://doi.org/10.1109/TNNLS.2018.2844093

[11] Qu, X., Huang, Z.H., Mao, N., Xu, Y. Cai, G. and Fang, Z. (2019) A Grain-Adaptive Computing Structure for FPGA CNN Acceleration. *IEEE* 13*th International Conference on ASIC*, Chongqing, China, 2019, 1-4. https://doi.org/10.1109/ASICON47005.2019.8983480

[12] Simonyan, K. and Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556. https://arxiv.org/abs/1409.1556v6

[13] Howard, A.G., *et al.* (2017) MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861. https://arxiv.org/abs/1704.04861

[14] He, K., Zhang, X., Ren, S. and Sun, J. (2016) Deep Residual Learning for Image Recognition. 2016 *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 27-30 June 2016, 770-778. https://doi.org/10.1109/CVPR.2016.90