IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Review of Computer Engineering Studies
Vol. 9, No. 4, December, 2022, pp. 136-140
Journal homepage: http://iieta.org/journals/rces

PYNQ Framework Based Object Recognition Implementation Using Convolution Neural Network (CNN)

Jie Wang, Li Xu*

School of Electronics and Information Engineering, Ningbo University of Technology, Ningbo 315211, China

Corresponding Author Email: xuli@nbut.edu.cn

https://doi.org/10.18280/rces.090402

Received: 16 October 2022
Accepted: 2 December 2022

Keywords:
reconfigurable architecture, convolution neural network, accelerator

ABSTRACT

In this paper, we presented a PYNQ framework Based object recognition implementation using Convolution Neural Network (CNN) in Xilinx FPGA. A hardware-software co-design framework is used to implement the CNN for object recognition. Training CNN model on PC, and implementation object recognition under embedded system on PYNQ-Z1 FPGA. Compare to a single ARM processor core on FPGA, we achieve 43.2 times speedup ratio for object recognition implementation. The performance demonstrates that this model can be highly improved by exploring the hardware resources of the FPGA.

1. INTRODUCTION

Object recognition is a research hotspot in the field of computer vision, aiming to use some technique or method to obtain the location of the target in the image sequence. And determine the category to which the target belongs. Object recognition has high engineering application value in the fields of intelligent transportation, medical devices, security surveillance, etc. With the rapid development of target detection technology, many object recognition algorithms have been proposed by scholars, among which the target detection algorithm implemented using Convolution Neural Network (CNN) in deep learning has better detection performance compared with the traditional target detection algorithm. However, the computational process of CNN generally requires hundreds of millions of multiplication and accumulation operations, and the extremely large amount of computation demands high resources, bandwidth and computational speed of the hardware platform for CNN implementation.

To implement CNN-based object recognition algorithms, many embedded systems and supporting software development tools have emerged. And these systems are designed for faster computation, smaller device size, and lower operating power. CPUs, which use instruction sets for computing, are suitable for running software programs, but are not good at multiplying and accumulating operations on a large scale. GPUs are good at multiplying and accumulating operations and image processing, and are suitable for training and inference calculations of CNN. But GPUs' high power consumption makes them impossible to implement in low-power embedded platforms. Compared with CPU and GPU, ASIC has the advantages of both high computing rate and low power consumption, and has a small size, but the customization cost is too high and the development flexibility is poor. FPGA is a high-speed parallel general-purpose device with high flexibility, low cost, high integration, low power consumption, moderate development cycle and wide application. FPGA uses data parallel connection to perform various parallel calculations, and is therefore well suited for implementing accelerated computation of CNN.

In this paper, we aim to use the PYNQ framework of Xilinx FPGA to accomplish object recognition implementation. We take full advantage of the reconfigurable and low power consumption of FPGA to quickly implement complex data parallelism and large-scale multiplication and accumulation operations in CNN based the PYNQ framework.

The contributions of this paper are:

(1) Hardware-software co-design scheme to implement object recognition on Xilinx FPGAs via ARM and FPGAs.

(2) Fast implementation of CNN acceleration based on the PYNQ framework.

This paper is organized as follows: Chapter 2 provides related work, Chapter 3 represents a PYNQ framework based CNN Accelerator. Testing and analysis is presented out in Chapter 4. Finally, gives a conclusion.

2. RELATED WORK

With the wide application of CNN in image processing and the acceleration of FPGA reconfigurable architecture, more and more people adopt FPGA to implement CNN accelerator. Yahia designed a PYNQ-YOLO-Net based on PYNQ framework and YOLO framework, and implemented a mask detection application using this net. The experimental results is that compared with the software alone, the recognition time is reduced to 1/3 of the original time. While the computational throughput is more than 30 times of the original and the recognition rate is guaranteed to be more than 90% [1]. Wang et al. proposed a fast CNN generation model based on PYNQ framework. Compared with ARM alone running CNN, the computational throughput increased by 20 times and energy consumption reduced by 28 times. Those papers demonstrated that PYNQ framework can quickly achieve CNN accelerator [2]. Hou designed a license plate recognition system based on the PYNQ framework, and finally achieved a license plate recognition rate of 97.89% with limited CNN network

parameters [3]. Maclellan implements radio modulation classification by spectral information using Matlab and PYNQ [4].

Many other research works focused on the PYNQ framework conjunction with OpenCV [5] framework and YOLO [6] framework. In addition to the use of PYNQ framework, CNN accelerator has also been implemented using FPGA alone [7-9]. The overall indication is that FPGA as well as the PYNQ framework are very useful to implement CNN algorithms for accelerating and completing object recognition implementation.

3. PYNQ FRAMEWORK BASED CNN ACCELERATOR

3.1 CNN

Due to the improvement of deep learning theory, CNN have been developed rapidly, and a number of new algorithms have emerged in speech recognition [10], face recognition [11], reinforcement learning [12], and natural language processing [13]. In CNN algorithms, neurons in the same convolutional layer are not associated with each other, while there are computational relationships between neurons in neighboring convolutional layers. Five main parts-data input layer, convolutional layer, pooling layer, fully connected layer and classification output layer, constitute the CNN. The data input layer represents the input of the original image data, and the input image features are collected through the convolutional layer and the pooling layer, and the collected image features are input to the fully connected layer and the classification output layer to complete the classification and recognition of the input image. The structure of the CNN is shown in Figure 1.

(1) Convolutional layer

The core idea of convolutional layers is to simulate neurons in the human brain, but unlike information transfer in the brain, in CNN, information is passed from layer to layer to each other, from the convolutional layer to the pooling layer, then from the pooling layer to the convolutional layer, and finally arrives in the classifier, which outputs the information of recognition result.

The main role of the convolutional layer is to extract features from the input feature map, and the parameters of the optimized convolutional kernel and BIAS can be obtained by BP (back propagation algorithm). Generally, a single convolutional layer extracts simple feature parameters, while a multi-layer convolutional layer can obtain more complex feature information by iterative methods.

The convolution is mathematically divided into the convolution of continuous and discrete functions, defined as follows.

$$(f*g)(n)=\int_{-\infty}^{+\infty} f(\tau)g(n-\tau)d\tau$$
$$(f*g)(n)=\sum_{\tau=-\infty}^{\infty} f(\tau)g(n-\tau) \quad (1)$$

The convolution formula in CNN is defined as follows.

$$f(n,i,j)=\sum_{m=0}^{xn-1}\left(\sum_{v=0}^{kh-1}\left(\sum_{u=0}^{kw-1} w(n,m,u,v)*x(m,u+i*sw,v+j*sh)\right)\right) \quad (2)$$

where, kw and kh stands for width and height, sw and sh represent the step lengths in the horizontal and vertical directions. $f(n, i, j)$ represents the n output of the first image at the point (i, j), $w(n, m, u, v)$ represent w weights.

(2) Pooling layer

The features collected through the convolutional layer can, in principle, be directly passed into the fully connected layer for network computation. However, the computational overhead of this approach is high, and the training and testing time of the CNN can be seriously increased, thus affecting the real-time performance of the algorithm. The goal of designing the pooling layer is to reduce the computational effort of the convolutional network by reducing the parameters of the feature map output from the convolutional layer. Therefore, the pooling layer is usually added after the convolutional layer to downsample the feature maps. Since a picture contains a lot of redundant information, the pooling layer can remove the redundant information to extract the important features. Pooling layer can also prevent overfitting to a certain extent. Overall, pooling layer effectively reduces the amount of data in the feature map after convolution calculation and improves the computation speed. As shown in Figure 2, the maximum pooling method and the average pooling method are the more common pooling methods.

(3) Fully connected layer

In a CNN structure, after multiple convolutional and pooling layers, one or more fully connected layers are usually connected. Each neuron in a fully connected layer is connected to all the neurons in its previous layer. Fusion of the local feature information collected in the convolutional or pooling layers is the main function of the fully connected layers. In order to reduce the network computation time of the CNN, the activation function of each neuron in the fully connected layer is usually a Rectified linear unit (ReLU) function. The output feature information of the last fully connected layer is fed to the classifier for data classification.

(4) Activation function

The activation function used Sigmoid function is a function used for binary classification which maps the input real numbers between (0, 1). The formula of Sigmoid function is defined as follows.

$$f(x)=\frac{1}{1+e^{-x}} \quad (3)$$

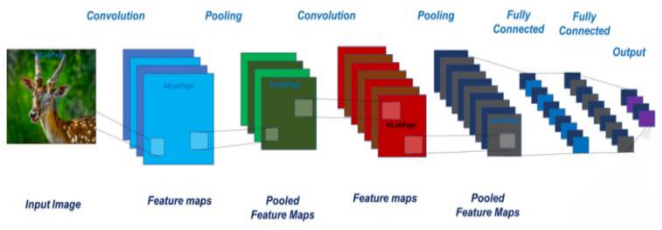

Figure 1. Convolutional neural network structure for object recognition

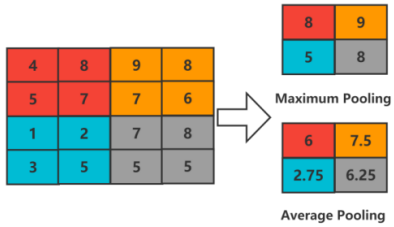

Figure 2. Maximum pooling & average pooling

3.2 PYNQ framework

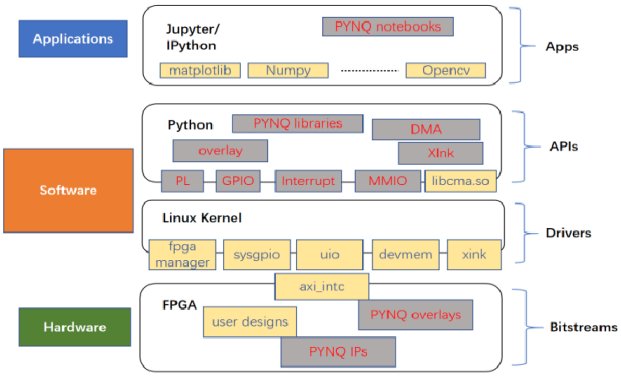

Figure 3. PYNQ framework

The PYNQ architecture is shown in Figure 3. From this figure, we can see that the PYNQ system architecture is divided into three layers: a hardware layer based on FPGA design, a software layer based on ARM-based Linux kernel plus Python, and an application layer based on Jupyter Notebook.

The hardware layer is designed mainly to realize the cooperative interaction between PS and PL, the whole FPGA part of the design is called Overlay also called hardware library, which is ZYNQ's PL (FPGA) design. overlay is mainly used to accelerate software applications, also for multi-user and multi-application to generate different bitstream files, and can be called through software APIs to the logic functions on the FPGA can be dynamically switched. The whole process is based on PYNQ, the C driver software is encapsulated in the PYNQ framework and provides the user with a Python call interface. The user can change the access IP of the overlay by changing the parameters of the Python interface, thus completing the underlying control of the programmable hardware.

3.3 Hardware-Software co-design framework

The CNN hardware-software co-design framework based on the PYNQ framework is shown in Figure 4. The test images are taken through the hardware, software and application layers of the PYNQ architecture and thus the test results. The hardware layer is supported by the PYNQ development framework, which also encapsulates the driver program, regulates the underlying hardware of the DMA, burns bitstream files, and solves tcl files. The software layer mainly completes the configuration process of encapsulating the computational tasks of each layer and provides a parameterized class call interface for designers. The application layer can automatically deploy the already trained Tensorflow models and designed to run on configurable CNN accelerator. During the development of the hardware architecture-driven control program, the program writer needs to complete the configuration of the entire DMA channel transfer using a high-level programming language based hardware driver. As showed as Figure 4, a large amount of computation is embedded in the computation of the network layer based on the PL part. Therefore, the basic model of CNN needs to be built by PC and passed to the PYNQ framework for more deep learning as well as practical operation, in order to meet the design requirements of hardware architecture and configuration. The ultimate goal is to make it easier and faster for users to design CNN accelerator based on the PYNQ framework for object recognition implementation.

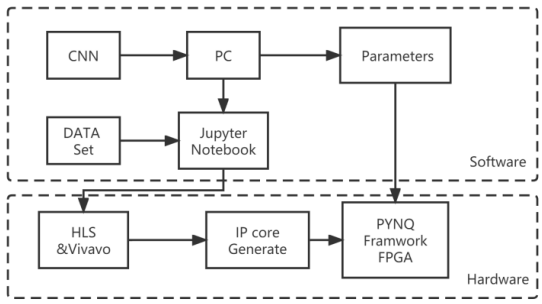

Figure 4. Hardware-Software co-design framework

4. TESTING AND VALIDATION

(1) Dataset

The test in this paper uses CIFAR-10 dataset, CIFAR-10 is a small volume dataset used to recognize common objects, which includes 10 common objects, such as cat, dog, deer, frog, horse, ship, truck, airplan, birds and automobile. The images are RGB color type, size 32*32, and the dataset includes 10,000 test images and 50,000 training images. the images included in CIFAR-10 are shown in the Figure 5. The CIFAR-10 dataset is from the real world, and real world objects have the characteristics of large noise and objects with different proportions and features, which can make recognition more difficult and lead to direct linear models does not perform well on the dataset. In this paper, 10,000 images are used for training and 50,000 images are used for testing.

(2) CNN model

The CNN model for this test consists of convolutional layer, pooling layer and fully-connected layer. As shown in Figure 6, two convolutional layers and two pooling layers and two fully connected layers are used. a color map with 32* 32 resolution of the input image of the CIFAR-10 dataset is added to the first convolutional layer of the model with the RELU activation function followed by the maximum pooling layer. The second convolutional layer is also processed through the RELU activation function maximum pooling layer. The two convolutional layers and the pooling layer are followed by a fully connected layer and finally the output layer. The output layer outputs 10 numbers, which are used to represent the classification scores in each of the 10 objects in CIFAR-10 datasets. And the highest value being the final recognition result.

(3) FPGA Platform

This paper uses the PYNQ-Z1 embedded development board as the system test platform, which is a heterogeneous multi-core platform based on the Xilinx ZYNQ-7000, including an ARM processor and a FPGA. The ARM and FPGA are associated high-speed communication interfaces. In addition to inheriting the powerful processing performance of the traditional ZYNQ platform, the PYNQ-Z1 is also compatible with Arduino interfaces and standard Raspberry Pi interfaces, making the PYNQ-Z1 highly scalable and open source. Figure 7 shows the physical diagram of the PYNQ-Z1 development board. We use Xilinx Vivado tools for the hardware part and ARM-based Ubuntu 18.04 for the software

part. At the same time, the PC is used to extract the weight and parameters of the convolutional layers and fully connected layers and download them to the PNYQ Framework for processing. Finally, the implementation of CNN accelerator is completed to improve the practicality and real-time performance of object recognition.

(4) Results and analysis

The object recognition system is mounted on the PYNQ-Z1 development board, and the hardware and software environments of the board need to be properly configured before testing and validating. The Python programming language is used to implement the control operations. CIFAR-10 dataset was used to test the performance of the system, verify the design and compare with the only PYNQ-Z1 ARM processing for CNN acceleration, the test results are shown in Table 1.

It can be seen that the acceleration ratio of the CNN accelerator based on the PYNQ framework for object recognition is about 43.2 times compared to the only ARM for CNN acceleration. The average power consumption of the CNN accelerator during the experimental test is only 1.921 W. The system dynamic power consumption of the accelerator is only 1.794 W, and the static power consumption of the accelerator is 0.153W. ARM processor in the PYNQ-Z1 embedded development board accounts for 91% of the dynamic power consumption of the whole system, but when system running the power of the FPGA part of the accelerator circuit is lower.

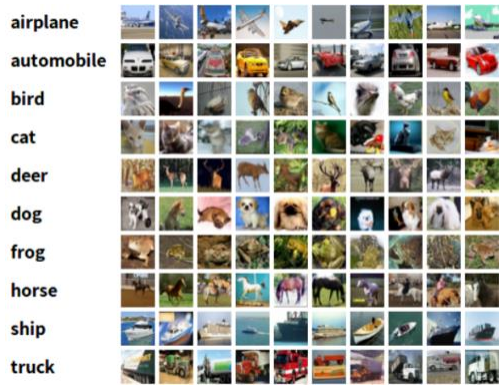

Figure 5. CIFAR-10 image examples

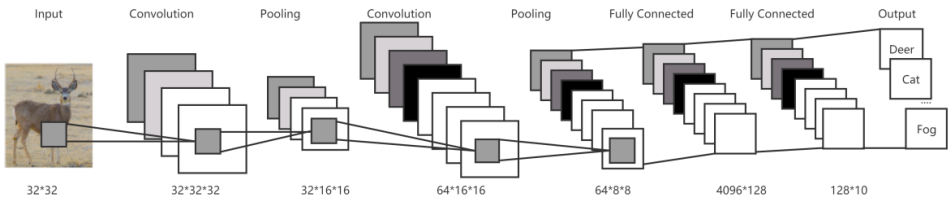

Figure 6. CNN model

Figure 7. PYNQ-Z1 FPGA Platform

Table 1. Results of the test

Platform	Frequency	Time per 1 image(ms)
PYNQ-Z1 only ARM	650Mhz	1.34
PYNQ-Z1 with FPGA	650Mhz (ARM)+30Mhz (FPGA)	0.031

5. CONCLUSION

In this paper, we present PYNQ framework Based object recognition implementation using CNN. A hardware-software co-design framework is designed for the hardware and software of the CNN accelerator based on the PYNQ framework. The experimental results demonstrates that the computational speed of CNN accelerator designed and implemented in this paper reached 35.39 fps when testing the CIFAR-10 dataset. The computational speed is 43.2 times faster than the only ARM processor on the PYNQ-Z1 with CNN program. The recognition ratio of the hardware implementation and software implementation remains basically the same, both achieving more than 98%. The average power consumption of the CNN accelerator based on PYNQ framework in this paper for object recognition implementation is only 1.921w, which is very suitable for the rapid deployment and use of low-power embedded devices.

ACKNOWLEDGMENT

This research was supported by Zhejiang Provincial Natural Science Foundation of China, Grant No.: LY19F020008; Zhejiang students' technology and innovation program (XinMiao program), Grant No.: 2022R428A002.

REFERENCES

[1] Said, Y. (2020). Pynq-YOLO-net: An embedded quantized convolutional neural network for face mask detection in COVID-19 pandemic era. International Journal of Advanced Computer Science and Applications, 11(9): 100-106. https://dx.doi.org/10.14569/IJACSA.2020.0110912

[2] Wang, E., Davis, J.J., Cheung, P.Y. (2018). A PYNQ-based framework for rapid CNN prototyping. In 2018 IEEE 26th Annual International Symposium on Field-

Programmable Custom Computing Machines (FCCM), pp. 223-223. https://doi.org/10.1109/FCCM.2018.00057

[3] Hou, X., Fu, M., Wu, X., Huang, Z., Sun, S. (2018). Vehicle license plate recognition system based on deep learning deployed to PYNQ. In 2018 18th International Symposium on Communications and Information Technologies (ISCIT), IEEE, Bangkok, Thailand, pp. 79-84. https://doi.org/10.1109/ISCIT.2018.8587934

[4] Maclellan, A., McLaughlin, L., Crockett, L., Stewart, R. (2019). FPGA accelerated deep learning radio modulation classification using MATLAB system objects & PYNQ. In 2019 29th International Conference on Field Programmable Logic and Applications (FPL), Barcelona, Spain, IEEE, pp. 246-247. https://doi.org/10.1109/FPL.2019.00045

[5] Wang, L.Z., Wang, J.Y., Xu, Y.J. (2022). Design of contactless intelligent epidemic prevention system based on PYNQ. Engineering Letters, 30(2): 218-231.

[6] Bao, C., Xie, T., Feng, W., Chang, L., Yu, C. (2020). A power-efficient optimizing framework FPGA accelerator based on winograd for yolo. IEEE Access, 8: 94307-94317. https://doi.org/10.1109/ACCESS.2020.2995330

[7] Wu, D., Zhang, Y., Jia, X., Tian, L., Li, T., Sui, L., Shan, Y. (2019). A high-performance CNN processor based on FPGA for MobileNets. In 2019 29th International Conference on Field Programmable Logic and Applications (FPL), Barcelona, Spain IEEE, pp. 136-143. https://doi.org/10.1109/FPL.2019.00030

[8] Bai, L., Zhao, Y., Huang, X. (2018). A CNN accelerator on FPGA using depthwise separable convolution. IEEE Transactions on Circuits and Systems II: Express Briefs, 65(10): 1415-1419. https://doi.org/10.1109/TCSII.2018.2865896

[9] Kala, S., Jose, B.R., Mathew, J., Nalesh, S. (2019). High-performance CNN accelerator on FPGA using unified winograd-GEMM architecture. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(12): 2816-2828. https://doi.org/10.1109/TVLSI.2019.2941250

[10] Syed, S.A., Rashid, M., Hussain, S., Zahid, H. (2021). Comparative analysis of CNN and RNN for voice pathology detection. BioMed Research International, 2021. https://doi.org/10.1155/2021/6635964

[11] Chavda, A., Dsouza, J., Badgujar, S., Damani, A. (2021). Multi-stage CNN architecture for face mask detection. In 2021 6th International Conference for Convergence in Technology (i2ct), IEEE, pp. 1-8. https://doi.org/10.1109/I2CT51068.2021.9418207

[12] Gu, B., Sung, Y. (2021). Enhanced reinforcement learning method combining one-hot encoding-based vectors for CNN-based alternative high-level decisions. Applied Sciences, 11(3): 1291. https://doi.org/10.3390/app11031291

[13] Widiastuti, N.I. (2019). Convolution neural network for text mining and natural language processing. In IOP Conference Series: Materials Science and Engineering, IOP Publishing, 662(5): 52010. https://doi.org/10.1088/1757-899X/662/5/052010