Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 5, No. 3, 2023

FPGA Hardware Acceleration Research and Implementation of Deep Learning Algorithms

Yuxuan Hu *

School of Electrical Engineering, Chongqing University, Chongqing 400444, China
*** Corresponding author Email:** 20193263@cqu.edu.cn

Abstract: The convolutional neural network model is an important algorithm for deep learning, and YOLOv3-tiny based on this model has excellent object detection ability. However, the computational power required by the model is still large, and it is difficult to realize the application in the embedded field. This paper proposes a hardware acceleration method for YOLOv3-tiny and implements it on FPGA platform. Firstly, the fixed-point quantitative processing was carried out for the network, and an appropriate fixed-point strategy was designed with the data accuracy as the index. Secondly, the parallel computing design and pipeline optimization principle were carried out, and the FIFO structure was introduced to shorten the running time. Finally, the experiment was carried out on the Xilinx PYNQ-Z2 platform, and the data were compared with the previous related work.

Keywords: YOLOv3-tiny; Hardware Acceleration; FPGA; Convolutional Neural Network.

1. Introduction

With the rise and vigorous development of artificial intelligence today, deep learning algorithms have a wide range of applications in real life. Convolutional neural network (CNN) is an important algorithm in deep learning. Its model structure has a history of more than 20 years. It can learn the rule and feature information of samples from data sets and directly process image data, so it has good effects and is widely used in computer vision fields such as face recognition, target detection and image discrimination. However, with the rapid development of the field of artificial intelligence, the complexity of CNN application scenarios continues to increase, and the requirements for model accuracy are also gradually increasing[1]. Therefore, the model size and computational complexity of CNN are also increasing, which requires the computing power of computing hardware devices. In the face of large-scale convolution calculation, the traditional central Processing Unit (CPU) architecture has poor parallelism and is not suitable for dealing with large-scale matrix operations. The advanced Graphics Processing Unit (GPU) has good parallelism and is suitable for large-scale operations, which is widely used in the training and reasoning of CNN models, but it is expensive and has high power consumption. And the relatively fixed hardware structure is difficult to meet the embedded mobile terminal. Based on various limitations, Field Programmable Logic Gate Array (FPGA), a semi-custom solution with strong computing power, low power consumption, rich logic resources and good parallelism, is very suitable as a hardware platform to accelerate the CNN optimization calculation method [2,3].

At present, in the field of object detection, the YOLO algorithm [4,5] has good generalization ability. When low-precision data is used to represent parameters such as network weights, the neural network can still have good performance and achieve a good balance between accuracy and speed, but it still requires large computing power. Based on the FPGA platform, this paper designs a hardware acceleration scheme for YOLOv3-tiny, which further optimizes the computational efficiency of CNN on FPGA.

2. YOLOv3-tiny Network and Hardware Acceleration Principle

2.1. YOLOv3-tiny Network Model Structure

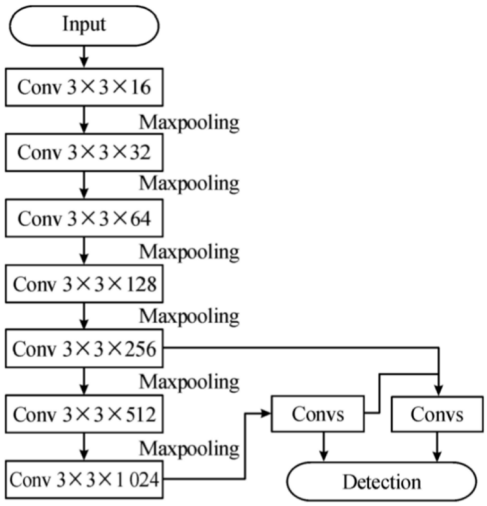

Figure 1. Network structure diagram of YOLOv3-tiny

Yolov3-tiny is a simplified version of YOLOv3 network, which simplifies the number of network layers, reduces the amount of calculation, and obtains faster calculation speed. However, the model recognition accuracy is lower than that of YOLOv3, which is a lightweight neural network model. YOLOv3-tiny model is mainly composed of a backbone network and a detection network, as shown in Figure 1, where the backbone network is composed of 13 convolutional layers and 6 Max pooling layers, and its network structure is shown in Figure 1. The default size of the model feature input image is 416×416, and the feature of the detection target is extracted through the convolution layer, and the convolution kernel size is 3×3. A max-pooling layer is connected after each convolutional layer. In order to detect objects of different sizes, two resolution feature maps of 13×13 and 26×26 are finally extracted, the former is used to extract features of large objects, and the latter is used to detect small objects. The backbone network of YOLOv3-tiny is a typical cylindrical

CNN network structure, and the calculation of convolution and pooling is unified, which is convenient for hardware acceleration. Detection is the last layer of the network output, which can output features W×H× (4+1+C) ×B, where W represents the width of the network predicted border, H represents the height of the network predicted border. The output image is divided into W×H grids, and each grid has (4+1+C) ×B channels, where C is the category judgment information and B is the number of bounding boxes.

2.2. Principle of Fixed-point Number Optimization

YOLOv3-tiny's original weight parameter is represented by 32bit floating-point numbers [6], and the data is redundant in parameter accuracy. As a numerical expression with fixed decimal point position, the fixed-point number is composed of sign bits, integer bits and decimal places, which is more flexible and different from the fixed value of floating-point numbers, and converts 32bit floating-point numbers into fixed-point numbers with lower digits. It has advantages in storage and calculation, and the utilization efficiency of computing resources is higher[7]. At the same time, CNN has a strong tolerance for low precision weight values. When the weight value is low precision, it has little impact on the detection accuracy. Therefore, it is a better trade-off choice to choose the appropriate number of fixed points for optimization, and sacrifice a small model accuracy to greatly improve the calculation and storage efficiency.

The total bit width of the fixed-point number is W, the bit width of the integer part is I, and the quantization factor is (W-i-1). S_float(32), int(W), and I_float(32) are used to represent the original floating-point data, the fixed-point quantized data, and the fixed-point unquantized data, respectively. The relationship between the three is shown in Equation (1) :

$$\text{int(W)} = \text{S\_float(32)} \times 2^{\text{W-I-1}} \quad (1)$$

$$=\text{I\_float(32)} = \text{int(W)}/2^{\text{W-I-1}} \quad (2)$$

When the total bit width and the integer part bit width are fixed, the above equation can be used for data quantization and inverse quantization, where the quantization factor (W-I-1) represents the accuracy, and the larger the quantization factor, the higher the accuracy of the data. In order to measure the accuracy of the fixed-point data loss, the fixed-point sample data is usually inverse quantized and compared with the original sample data. The overall quantization accuracy loss can be expressed by the following Equation (2)

$$\text{loss} = \frac{\sum_{n=0}^{N}(\text{S\_float}_n(32) - \text{I\_float}_n(32))^2}{\sum_{n=0}^{N}\text{S\_float}_n(32)^2} \times 100\% \quad (3)$$

Where N denotes the total number of data to be quantized. When the loss of accuracy is within an acceptable range, the quantization work can be completed, and the fixed-point data can be used for calculation.

2.3. Principles of Parallel Computing and Pipeline Optimization

Parallel computing design is the key of hardware acceleration module design[8,9]. In order to improve the performance of the hardware acceleration module, in the acceleration of convolutional neural network, it is necessary to make the basic computing units calculate at the same time as possible, while taking into account the resource consumption of FPGA, the work efficiency of the hardware acceleration module and the data throughput, and finally obtain the result of the trade-off between data. The dimensions of parallel computing are diverse, including inter-convolution kernel parallelism, inter-feature map parallelism, intra-feature map parallelism, and intra-convolution kernel parallelism. The size of the convolution kernel in the YOLOv3-tiny backbone network is 3×3, and 9 multiplications and 8 addition operations are needed in the calculation process. If the serial execution is performed, it needs to loop 9 times. It greatly reduces the computing time, but only using unroll consumes a lot of hardware resources. Therefore, in order to get a balance between computing time and hardware resources, we also introduce pipeline optimization.

Pipelined structure is widely used in hardware design. By optimizing the serial execution of the function through the pipeline, it can be parallel in different stages and improve the throughput, as shown in Figure 2. By comparison, it can be seen that for 3 tasks, the serial design takes 9 clock cycles, the parallel design takes 3 clock cycles, and the pipelined design takes 5 clock cycles. Compared to parallel computing, which requires three times the hardware resources, pipelining only requires extra memory units and registers.

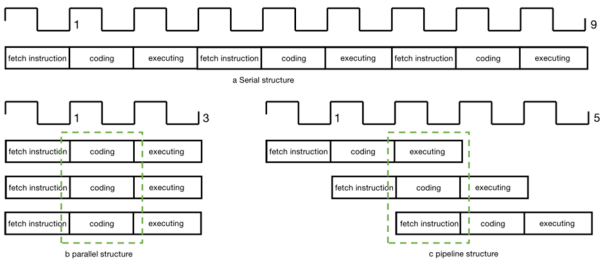

Figure 2. Serial, parallel and pipeline structure

3. Hardware Acceleration Design for YOLOv3-tiny

3.1. Fixed-point Quantization

The original weight parameter of YOLOv3-tiny is 32-bit floating point number, so this paper tests the accuracy loss of weight under different bit width representation by fixed-point quantization and reverse quantization, as shown in Figure 3. It can be seen that from the use of 8-bit fixed-point number, with the decrease of bit width, the accuracy loss is gradually obvious. Considering the unity of the fixed-point and the ability of the fixed-point number to cover the data range of the intermediate layer, the decline of data accuracy caused by the 16-bit fixed-point number is small, so this paper uses the 16-bit fixed-point number as the overall fixed-point number of the accelerator.

Figure 3. Serial, parallel and pipeline structure

3.2. Parallel Computing Design

The convolution kernel in YOLOv3-tiny has two sizes of 3×3 and 1×1, which will affect the calculation speed in parallel computing. Therefore, the parallel computing design

can be optimized for this pain point. The number of input and output channels is mostly multiple of 16 and the value is relatively large, and the output channel is larger than the input channel. In order to make full use of the hardware resources, the input parallel factor is selected as 8, and the output parallel factor is 16, which can obtain better computing speed, as little as possible occupy the hardware resources, and get a better balance between efficiency and resource utilization.

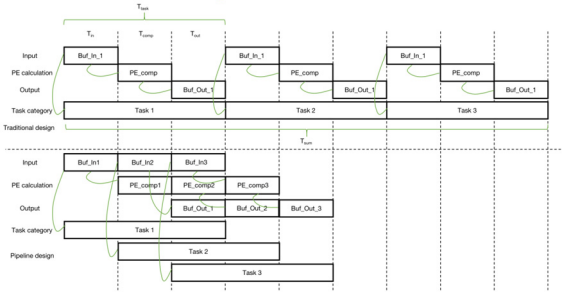

Figure 4. Temporal comparison of pipelined structure and traditional design

In the process of multi-channel parallel execution, the operation of write is more frequent than that of read. Because the number of output channels is often larger than that of input channels, the output port is prone to data blockage. Therefore, this paper uses AXI4-Stream protocol and introduces First-in First-out (FIFO) structure to optimize the pipeline structure. When the Loop Initial Interval(II) is 1, two clock cycles are needed to obtain the input data of the eight channels. When the new eight input channels arrive, the 16 output channels generated by the previous input are cleared, to avoid this, the input stream must pause for three additional cycles to complete a full input-output operation, so a total of five cycles are required. When added to the FIFO structure, the output data is sent one by one, so that the input and output data of the accelerator can be pipelined at the task level. Figure 4 takes three tasks as examples to carry out the timing comparison between the pipelined structure adopted in this design and the traditional design.

In Figure 4, the time T_{task} to complete an operation at the task level includes the time T_{in} to read data from input register, the time T_{out} to calculate in PE and the time T_{cmp} to read calculation results in PE, so the total time to complete three tasks under the traditional structure is

$$T_{sum} = 3 \times (T_{in} + T_{out} + T_{cmp}) \quad (4)$$

After introducing the FIFO structure

$$T_{FIFO\_sum} = T_{in} + 3\min(T_{in}, T_{out}, T_{cmp}) + T_{out} < T_{sum} \quad (5)$$

Greatly reduces the running time.

4. Experiment and Comparison

4.1. Experimental Design

The design mainly uses Vivado HLS 2017.4 to design the IP of YOLOv3-tiny accelerator, and high level synthesis (HLS) can convert C /C ++ language into RTL language required by FPGA. Then the accelerator design of YOLOv3-tiny was completed through the software development kit SDK of Xilinx company. The experimental verification platform is Xilinx PYNQ-Z2. The development board uses FPGA+ARM architecture, and the main chip is ZYNQ XC7Z020-1CLG400C chip, including 512MB DDR and 220 DSP.

4.2. Experimental Results and Analysis

In this paper, the comprehensive performance of the system design is evaluated and compared with the previous related work with exactly the same design function, as shown in Table 1. Through comparison, it can be seen that the design has a good acceleration effect and a better performance to power consumption ratio.

Table 1. Comprehensive comparison of system design performance

Performance	Reference [10]	Reference [11]	Reference [4]	This paper
Test platform	VC707	Ultrascale+	GTX Tian X	XC7Z020
Network model	YOLOv2-tiny	YOLO-tiny	Sim-YOLOv2	YOLOv3-tiny
PL clock rate	200 MHz	-	1GHz	100 MHz
BRAM	1026	-	-	185
DSP	168	-	-	160
LUT-FF	60k-86k	-	-	26k-46k
Input image size	416×416	416×416	416×416	416×416
Performance per Watt/ (GOPS/W)	25.41	12	8.89	9.35
Power Dissipation/W	18.29	6	170	3.8

5. Conclusion

This paper studies the acceleration of deep learning algorithms based on FPGA. Taking YOLOv3-tiny as an example, after in-depth analysis of the related characteristics of CNN model, a FPGA-based YOLOv3-tiny hardware acceleration structure is proposed. In this paper, fixed-point quantization is used to reduce the amount of network calculation and data storage for YOLOv3-tiny network parameters. The data is quantized from 32-bit floating point to 16-bit fixed-point number, which reduces the amount of data storage and calculation, and the network detection accuracy is basically unchanged. FIFO structure is used to parallel the input and output channels, which improves the network acceleration performance and reduces the consumption of hardware resources. Through the verification results on the X ilinx platform, it can be concluded that the hardware acceleration structure for YOLOv3-tiny convolutional neural network designed based on FPGA in this paper has a good balance between reducing power consumption and reducing hardware resource consumption.

The comprehensive performance is better than that of the current research schemes, and the computing resources and storage resources are relatively small, which is of great significance to realize YOLO acceleration in the case of limited resources of mobile terminals.

References

[1] Zhang, C., Li, P., Sun, G.Y., Guan, Y.J., Xiao, B.J., Cong, J. (2015) Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. International Symposium on Field-Programmable Gate Arrays (FPGA), 161-170.

[2] Sun, F., Wang, C., Gong, L., Xu, C., Zhou, X. (2017) A High-Performance Accelerator for Large-Scale Convolutional Neural Networks. 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), 1-9.

[3] Venieris, S.I., Bouganis, C.S. (2016) FPGAConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs. IEEE International Symposium on Field-Programmable Custom Computing Machines, London, UK, 40-47.

[4] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016) You only look once: unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779-788.

[5] Ren, S., He, K., Girshick, R., Sun, J. (2017) Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis & Machine Intelligence, 39(6): 1137-1149.

[6] Bi, F., Yang, J. (2019) Target Detection System Design and FPGA Implementation Based on YOLO v2 Algorithm. 2019 3rd International Conference on Imaging, Signal Processing and Communication (ICISPC).

[7] Wai, Y.J., Yussof, Z.B.M., Salim, S.I.B., Chuan, L.K. (2018) Fixed point implementation of tiny-yolo-v2 using opencl on fpga. International Journal of Advanced Computer Science & Applications, 9(10):506-512.

[8] Lu, Z.J. (2013) Research on the parallel structure of convolutional neural network based on FPGA. Thesis of Harbin Engineering University.

[9] Nakahara, H., Yonekawa, H., Fujii, T., & Sato, S. (2018) A Lightweight YOLOv2: A Binarized CNN with A Parallel Support Vector Regression for an FPGA. Field Programmable Gate Arrays. ACM, 31-40.

[10] Nguyen, D.T., Nguyen, T.N., Kim, H., Lee, H.J. (2019) A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1-13.

[11] Preuer, T.B., Gambardella, G., Fraser, N., Blott, M. (2018) Inference of Quantized Neural Networks on Heterogeneous All-Programmable Devices. Design, Automation & Test in Europe Conference & Exhibition, 833-838.