Research and Design of Open Convolutional Neural Network Based on FPGA

Huining Lin¹,², Yu Wang¹,², Jian Zhou¹,², *, Shengqing Pei¹,², Zhiqiang Geng¹, Zhuang Chen¹, and Zhihan Guo³

¹College of Information Engineering, Wuhan University of Technology
²Hubei Key Laboratory of Broadband Wireless Communication and Sensor Networks, Wuhan University of Technology
³School of Energy and Power Engineering

*Corresponding author e-mail: jianzhou@whut.edu.cn

Abstract. In this paper, using the fluidity and parallelism of FPGA, we design a set of memory scheduling mechanism to implement an open convolutional neural network. The self-built vehicle database and the cropped and zoomed Daimler Pedestrian Detection Benchmark dataset are each trained under different network structures, and their recognition rates reached 97.8% and 98.6%. An open feedforward network is implemented on the FPGA platform, and experiments show that for small convolutional networks with different structures, the FPGA platform can increase its recognition speed when resources permit.

Keywords: Convolutional neural network; FPGA; Object detect; Memory scheduling.

1. Introduction

Convolutional Neural Network (CNN) performs well in tasks such as image classification, semantic segmentation, object detection, and tracking, so they have become the focus of attention. This technology has also been widely used in industry, such as autonomous driving, video surveillance, voice recognition, etc. [1, 2, 3]. However, it needs to consume a lot of computing and storage resources during training and application, it is usually selected as a Graphics Processing Unit (GPU) as the platform for implementation [4]. GPU has the characteristics of high power consumption, it is difficult to play a role in the embedded environment, so more and more researchers began to study the use of Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC) to realize CNN.

CNN's feedforward network is very suitable for FPGA implementation. The parallel computing of FPGA can greatly accelerate the process of convolution calculation. Nowadays, there are many researches dedicated to CNN acceleration using FPGA, mainly divided into two development tools: Hardware Description Language (HDL) [5] and High Level Synthesis (HLS) [6, 7]. For HDL development, researchers often need to choose between power consumption and reconfigurability of the network. Many studies have achieved good results in power consumption and acceleration, but the network structure is actually fixed. Once the network is retrained, corresponding code modifications are required on the FPGA.
This article uses the parallelism of CNN network structure convolution calculation and the reusability of convolution kernel to design a set of memory scheduling mechanism to improve the calculation efficiency for the problem of insufficient internal RAM space; Load the network model in the form of a file to improve reconfigurability. Finally, the vehicle and the pedestrian are used as the detect objects to conduct experiments.

2. Design of detect system
This article is based on the structure of LeNet-5 and adjusted on the basis of its original network structure. LeNet-5 has seven layers, including 2 convolutional layers, 2 pooling layers, and 3 fully connected layers. It was proposed by Yann LeCun in 1998 for handwritten digit recognition [8]. We conduct corresponding training on Matlab for the different layers of the network, the size of the convolution kernel, and whether it is pooled, and identify the targets as pedestrians and vehicles. The trained network will be saved as a file in the form of parameters and imported into FPGA.

In the case of considering only forward propagation, this paper takes advantage of the parallel computing characteristics of convolutional neural networks and considers the premise that the network model is imported through parameters, and implements CNN on the FPGA platform. The overall system structure is shown in Fig 1.

![Figure 1. Identification process.](image)

3. Design of open convolution neural network

3.1. Convolution calculation
In this paper, the convolution calculation module includes 5 × 5 and 3 × 3. The following introduction takes 3 × 3 as an example. Since the convolution operation module has the characteristics of multiple inputs and multiple outputs [9], the characteristics of FPGA pipelines can be used to implement the convolution operation. As shown in Fig 2, using 2 shift registers can complete a 3 × 3 convolution operation every clock. Before starting the convolution calculation, the first shift register is buffered, then the convolution operation starts at the second clock after the first image pixel is output from the second shift register.
Fig 3 is a timing diagram of the convolution operation process. The detection window size is 32 * 32, the depth of the shift register is 32. After the pixel data enters the first register, it moves out of the second register after 64 clocks. Controlled by the signal of shift_en. At the same time, within the first 9 clocks, the corresponding convolution kernel weight data will be read into the corresponding weight register. After that, the conv_en signal is set high and the convolution operation starts.

![Figure 2. 3*3 convolution process.](image)

3.2. Pooling

Pooling operation is usually performed following convolution calculation, which is used to reduce the feature dimension and the number of parameters [10]. Common pooling methods are maximum pooling and average pooling. In this paper, average pooling is used. As shown in Fig 4, in the implementation

![Figure 3. Timing of 3*3 convolution.](image)

![Figure 4. Pooling process.](image)
of the FPGA platform, the shift register method is used to average the data in the 2 * 2 area. After adding the 4 data, the result is shifted to the right by two digits to obtain the average value.

4. Memory scheduling of FPGA

Most CNN systems implemented in HDL are aimed at a fixed network structure, and the storage unit needed and the time of various operations to be performed in the network can be planned in advance. Using the pipeline structure, the operating efficiency of the entire network can be improved within the scope of resources. But this means that the network structure must be determined first, and the training part of the host computer must get a definite result. To make the FPGA side compatible with networks with different structures, the overall status of the network can be obtained by loading the network model parameters. The format of each layer of parameters in the network is shown in Fig 5.

Before the calculation of each layer, all information of the layer must be read from the ROM, including the current layer number, the size and number of convolution kernels, whether to pool, etc. The specific process is shown in Fig 6, where the number of input and output feature maps of the current layer is set to 6 for explanation. F1 ~ F6 are 6 RAM storage units, which store the calculation results of the previous layer, that is, the feature map group to be read out for convolution operation, F7 ~ F12 are the storage unit groups of the current layer calculation results, C1 ~ C6 Represents 6 convolution kernels. Proceed as follows:

Step1: copy the input original picture to F1 ~ F6;
Step2: Starting from i = 1 (i = 1, 2, ..., 6), perform a convolution operation on the data in F1 ~ F6 and Ci to obtain a total of 6 calculations for Tj (j = 1, 2, ..., 6) result;
Step3: If pooling is required, the result Tj obtained in the second step is pooled, otherwise step 4 is entered;
Step4: accumulate the 6 result graphs obtained according to the corresponding pixel positions and write them into Fi + 6;
Step5: judge whether to calculate to the last convolution kernel C6, if yes, then end, otherwise proceed to step 2.

| Input image size | Output image size | Total level | Current level | Maps quantity | Kernel quantity | Kernel size | Whether to pool | Weights of kernel |
|------------------|------------------|-------------|---------------|---------------|----------------|-------------|----------------|-------------------|
|                  |                  |             |               |               |                |             |                |                   |

Figure 5. Format of parameters.
Using the memory scheduling mechanism to realize the "ping-pong storage" of the feature map is essentially the reuse of the RAM storing the feature map. In terms of timing, the calculation of different feature maps between layers and in each layer is a serial relationship. When performing convolution operations, the operations of multiple inputs and a single convolution kernel are performed in parallel. As shown in fig 7, Level_1_en is the operation control enablement of the first layer of the network, Feature_11 to Feature_16 represent the input of the first layer, and Core_11 and Core_12 represent the first 2 of the 6 convolution kernels to be calculated at the current layer. In the calculation of the first layer, it can be seen that when calculating the first convolution kernel Core_11, the convolution operation of the 6 input images is parallel, and the correlation operation of the second convolution kernel is only performed after its completion. Finally, after the operation of this layer is completely finished, the next layer is entered.

Fig.8 shows the partially implemented network structure.
5. Experiment

CNN is trained on computer by using Matlab. The vehicle sample set is a self-built sample set, the data size is 32*32 pictures, the image value range is 0 to 255, the training sample set contains 40,000 images, and the test set contain 25000 images; the pedestrian sample set uses Daimler Pedestrian Detection in the 48*96 size part of Benchmark [11], the training set contains 22304 images, and the test set contains 9800 images. To facilitate training, we cut and rescale sample set 32*32 size. Fig 9 shows some samples. Pedestrians and vehicles can each achieve a recognition accuracy of 97.8% and 98.6% respectively on their respective test sets.

Table 1. Run time of pedestrian detection

| Network structure | CPU(i7-7700HQ) /ms | FPGA(XILINX XC6SLX150) /ms |
|-------------------|--------------------|--------------------------|
| Layer=3, core size=5 | 19.1               | 6.018                    |
| Layer=4, core size=3 | 21.8               | 9.015                    |
| Layer=5, core size=5 | 34.3               | 12.331                   |
Table 2. Run time of car detection

| Network structure | CPU(i7-7700HQ)/ms | FPGA(XILINX XC6SLX150) /ms |
|-------------------|-------------------|--------------------------|
| Layer=3, core size=5 | 18.7             | 6.121                    |
| Layer=4, core size=3 | 22.1             | 8.987                    |
| Layer=5, core size=5 | 34.2             | 12.132                   |

6. Conclusion
This paper designs and implements an open convolutional neural network model based on PFGA. The self-built vehicle data set and the Daimler Pedestrian Detection Benchmark data set after cropping and scaling are used to train different network structures, and the network parameter file will be read and run by itself. Compared with the computer, it is subject to the FPGA's own resources and program optimization. Currently, it can only run small convolutional networks with 3 to 6 layers and no more than 12 single-layer convolution cores. The design of the algorithm has room for further optimization.

Acknowledgments
This work was financially supported by 2020 National College Student Innovation and Entrepreneurship Training Program, Wuhan University of Technology Undergraduate Project of Independent Innovation Research Fund 2020, 2020 Wuhan University of Technology Independent Innovation Research Fund Postgraduate Free Exploration Project

References
[1] J. van der Geer, J.A.J. Hanraads, R.A. Lupton, The art of writing a scientific article, J. Sci. Commun. 163 (2000) 51-59.
[2] Luo, Yuhao;Yin, Dong;Wang, An;Wu, Wentao.Pedestrian tracking in surveillance video based on modified CNN[J].Multimedia tools and applications,2018,Vol.77(18): 24041-24058.
[3] Passricha, V.;Aggarwal, R.K..A Hybrid of Deep CNN and Bidirectional LSTM for Automatic Speech Recognition[J].Journal of Intelligent Systems,2019.
[4] Pei, Z.;Li, C.;Qin, X.;Chen, X.;Wei, G..Iteration time prediction for CNN in Multi-GPU Platform: Modeling and analysis(Article) [J].IEEE Access,2019,Vol.7: 64788-64797.
[5] Enrico Reggiani;Marco Rabozzi;Anna Maria Nestorov;Alberto Scolari;Luca Stornaiuolo;Marco Santambrogio.Pareto Optimal Design Space Exploration for Accelerated CNN on FPGA [A].2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) [C], 2019.
[6] Osuna-Coutiño JAJ;Martinez-Carranza J.High Level 3D Structure Extraction from a Single Image using a CNN-Based Approach. [J].Sensors (Basel, Switzerland), 2019, Vol.19(3).
[7] Sang-Soo Park;Kyeong-Bin Park;Ki-Seok Chung. Implementation of a CNN accelerator on an Embedded SoC Platform using SDS0C [A].ICDSP 2018:Proceedings of the 2nd International Conference on Digital Signal Processing [C], 2018.
[8] Lecun, Y.;Bottou, L.;Bengio, Y.;Haffner, P..Gradient-based learning applied to document recognition [J].Proceedings of the IEEE, 1998, Vol.86 (11): 2278-2324.
[9] LIU Hanying,ZHANG Yaotian,ZHANG Yuxi, et al,SAR target recognition and FPGA implementation based on deep learning.Computer Engineering and Applications, 2017, 53(S2): 371-375.
[10] Khatami, Amin;Babaie, Morteza;Tizhoosh, H.R.;Khosravi, Abbas;Nguyen, Thanh;Nahavandi, Saeid.A sequential search-space shrinking using CNN transfer learning and a Radon projection pool for medical image retrieval. [J]. Expert Systems with Applications, 2018, Vol.100.
[11] Fabian Flohr;Dariu Gavrla;T. Burghardt;D. Damen;W. Mayol-Cuevas;M. Mirmehdi.PedCut: an iterative framework for pedestrian segmentation combining shape models and multiple data cues [J]. British Machine Vision Conference 2013, 2013.