A Hardware Accelerator Based on Neural Network for Object Detection

Tong Zhao¹, Lufeng Qiao¹*, Qinghua Chen¹, Qingsong Zhang¹ and Na Li¹

¹Institute of Communication Engineering, Army Engineering University of PLA, Nanjing Jiangsu 210007, China.
*Corresponding author’s e-mail: 13357837783@189.cn

Abstract: At present, deep learning algorithms such as neural networks are widely used in all aspects of artificial intelligence. The computing performance of the CPU is low, and the power consumption of the GPU is large. This topic uses FPGA to study and implement the target detection algorithm. The proposed related technology can meet the accuracy requirements of the unmanned system for target detection, as well as the real-time requirements of the video stream. The frame rate of this design is 23.1FPS, and the mAP value is basically the same as that of CPU and GPU. The delay is 47.2% of Intel CPU, and the energy efficiency is 26% of Titan X GPU.

1. Introduction

In the field of image recognition and target detection, neural network technology breaks through the traditional technology method and can extract image features more efficiently, thus achieving higher accuracy. For example, Alexnet [1] relies on the powerful global feature representation capabilities of convolutional neural networks, and various network training techniques (data expansion, Dropout, local response normalization, nonlinear ReLU, etc.), and won the first in the ILSVRC 2012 image recognition competition. An increase of precision nearly ten percentage points compared to the traditional method. Not only that, with the advent of Alexnet, various deep network models (such as VGGNet [2], GoogLeNet [3], ResNet [4], etc.) emerge in an endless stream and are widely used in the field of image processing, significantly improving the precision of image recognition and detection. With the continuous development of neural networks, and the increasing performance of network, the structure scale is also expanding, which makes the overall calculation of the neural network greatly increased, and the storage space required for the weight and intermediate results rises linearly. The problem seriously restricts the application of neural network technology to hardware platforms with limited resources such as storage and computing. Therefore, if the FPGA can be used to implement the target detection algorithm, the hardware volume can be greatly reduced, and the implementation speed is fast, the flexibility is high, and the power consumption is low.

2. System overall architecture

This paper optimizes the SSD [5] target detection algorithm and implements it in FPGA. We replace the basic network in the typical SSD target detection algorithm with the MnasNet[6] network, and add two convolutional layers as the output layer of the prediction frame. At the same time, the fully connected layer in the network is removed, and the category is directly prediction. The corresponding chart is shown in Figure 1.
3. FPGA implementation of target detection algorithm

3.1 Initial screening of prediction frames

For each of priori box, a 3x3 Depthwise convolution is performed to get the value. The whole target detection is to find the category and confidence value corresponding to each prediction box. The design compares the confidence values one by one and selects the maximum confidence. The corresponding flow chart is shown in Figure 2.

![Flow chart of preliminary screening prediction box](image)

3.2 NMS process

This section is an FPGA implementation of the Non-Maximum Suppression (NMS) algorithm. The flow chart of the entire NMS is shown in Figure 3. After extracting the top-k in the prediction box, NMS is performed again, but since the design simplifies the calculation method, the extracted top-k is not sorted in descending order, so before the IOU is performed, it is necessary to maximum value of each class in top-k founding. It can be seen that the design divides the NMS into three parts. The first part is to extract the maximum value of each class, the second part is to write the final result into the result FIFO, and the third part is to further filter the remaining value.

The first part of NMS is mainly to screen top-k, which can select the maximum value of each class. So we initialize each class to 0 firstly, then select the maximum value of each class from top-k in the previous step, or there may be some classes, then the confidence value of these classes is always 0. In
In this design, the values are taken from FIFO one by one, determine which type of prediction box the value belongs to, and then update the value. In addition to the maximum value, the remaining values are written to the cache FIFO for the next step. So that you only need to traverse it to find the maximum of all categories.

In the process of using the FPGA implementation, it is found that the class is determined first, and then the data is compared and updated. It takes two clock cycles, so it is necessary to read one value every other clock cycle.

The data $a_2$, $b_2$, $c_2$, and $d_2$ in the setting graph are respectively the maximum values of the four types of prediction frames. It can be seen that it takes two clock cycles to read the data and compare it with the reference value, so the interval clock cycle is taken. The way of reading simplifies the way of implemented. The reference value of each participating comparison needs to be pre-calculated, which is the value with the highest confidence in the category currently participating in the comparison, so that the maximum value of all categories can be selected in one pass.

The main function of the second part is to write the result FIFO. After the maximum value of each class is selected in the first step, these maximum values are the final output, and these values are written to the result FIFO for storage for external software to read.

The main function of the third part is to filter the residual value. According to the calculation of the first two steps, the maximum value of each type of confidence is obtained, and then the data is read from the FIFO, and the IOU calculation is performed. If the result is greater than the threshold value of 0.5, it will be read. The predicted frame is culled, otherwise it is rewritten into the FIFO for the next round of screening. In order to save storage resources in this design, only one FIFO is used to store the intermediate result in the whole NMS process. Because the FIFO is repeatedly written, it is necessary to set a counter to indicate whether the data of each round is completed.

![Flow chart of the NMS](image-url)
4. Results analysis

The goal of this design is to process the video stream in real time, that is, the frame rate must reach 24 FPS, and the accuracy close to the original SSD target detection algorithm, and the power consumption of the system should be reduced as much as possible. This topic uses the Xilinx Virtex7 xc7vx690t hardware platform to map the entire design to the FPGA, mainly comparing the target detection schemes from the three aspects of mAP, FPS and energy efficiency, including the implementation of the GPU platform, the software implementation of the CPU and the rest. The FPGA platform is implemented and compared on the COCO and VOC2007 data sets.

4.1 VOC2007 data set results analysis

This design performs target detection application on the VOC2007 dataset, and the obtained results are compared with the results of different GPU platforms. The results are shown in Table 1.

| work           | Hardware platform | Basic network | Number of box | mAP  | FPS | J/img |
|----------------|-------------------|---------------|---------------|------|-----|-------|
| SSD300[5]      | Nvidia Titan X    | VGG16         | 8732          | 74.3 | 46  | 2.8   |
| SSD300[7]      | NVIDIA V100       | MobileNet     | 2493          | 68   | 37  | 2.3   |
| Our work       | Virtex7 xc7vx690t | MnasNet       | 25194         | 71.3 | 23.1| 0.73  |

In the data set of VOC2007, the mAP of this design reached 71.3, although it is lower than 74.3 of VGG16, but higher than 68 of Nvidia V100, it has basically met the accuracy requirement. In terms of energy efficiency, the design is 0.73J/img, which is only 26% of Titan X and 31.7% of V100. The energy consumption is very low, and FPS=23.1 can basically meet the real-time processing requirements of video stream.

4.2 COCO dataset results analysis

This design performs target detection application on the COCO data set, and the obtained results are compared with different GPU platform results. The results are shown in Table 2.

| work           | Hardware platform | Basic network | mAP  | FPS | J/img |
|----------------|-------------------|---------------|------|-----|-------|
| SSD300[5]      | Nvidia Titan X    | VGG16         | 23.2 | 45  | 2.8   |
| SSD300[8]      | Nvidia Jetson Nano| MobileNet     | 21   | 35  | 2.4   |
| Our work       | Virtex7 xc7vx690t | MnasNet       | 22.8 | 23.1| 0.73  |

This design can achieve 22.8 mAP on the COCO dataset, and the energy efficiency is only 0.73 J/img.

In summary, the accuracy of the same algorithm of the design is basically the same as that of the GPU, but the power consumption of the single picture is very low, and can basically satisfy the processing of the video stream.

5. Conclusion

In this paper, the FPGA implement the target detection algorithm. This design improves the typical SSD target detection algorithm, and adds the feature map of the prediction frame screening, and replaces the original convolution with Depthwise convolution, which reduces the amount of computation. The filtering process is also optimized so that the entire system can be mapped to the FPGA more efficiently. Compared with the GPU’s similar algorithm, mAP is slightly lower than the original SSD, but the unit energy consumption is only about 30% of the GPU. Compared with the CPU, the mAP of this design is higher than the CPU on the COCO and VOC2007 data sets, and the delay is only 47.2% and 22.8% of the CPU. In comparison with the FPGA design of DAC-SDC2018, the frame rate is 23.1 FPS, which can basically meet the computing needs of video streams.
References

[1] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[2] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[3] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[4] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[5] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.

[6] Tan M, Chen B, Pang R, et al. Mnasnet: Platform-aware neural architecture search for mobile[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 2820-2828.

[7] Lee Y, Hwang J, Lee S, et al. An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2019: 0-0.

[8] Wang L, Chen Z, Liu Y, et al. A Unified Optimization Approach for CNN Model Inference on Integrated GPUs[J]. arXiv preprint arXiv:1907.02154, 2019.

[9] Liau H, Yamini N, Wong Y L. Fire SSD: Wide fire modules based single shot detector on edge device[J]. arXiv preprint arXiv:1806.05363, 2018.

[10] Xu X, Zhang X, Yu B, et al. DAC-SDC low power object detection challenge for UAV applications[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019.