FPGA Hardware Acceleration Design for Deep Learning

Haochen Shi *

College of Information Sciences and Technology, Northwest University, Xi'an Shaanxi, 710000 China

* Corresponding Author Email: hs19868@essex.ac.uk.com

Abstract. A type of artificial neural network called a convolutional neural network (CNN) can learn characteristics from a huge amount of data and performs very well in the field of large-scale image processing. CNN simulates the behavior of a biological optic nerve. In recent years, with the development of deep neural network algorithms and hardware technology, the current "CPU+GPU" model servers cannot meet the neural network structure in various fields, so a large amount of deep CNN accelerators based on the FPGA platform have gradually emerged. FPGA is beginning to be used in the fields of image recognition and natural language processing because of its programmability, high performance, high stability, high security, and low power consumption. Though FPGA has proven to have better performance, there is still room for optimization at the design level. Yolov3, as a classical algorithm, still consumes a lot of time and computational resources in actual operations. To address this problem, this experiment partially optimizes the Yolov3 algorithm by introducing the CBAM attention mechanism in the Yolov3 model and pruning the embedded system with different proportions using the Network slimming method. Finally, it is verified on a TX2 embedded device developed by Nvidia using the COCO dataset. The experiment finds that the precision, mAP, and the number of parameters of the optimized Yolov3 algorithm under different optimization strategies. It is shown that the Yolov3 algorithm still has more optimization strategies that can reduce the time required for computation and the memory occupied more effectively without any degradation in accuracy.

Keywords: Deep Learning; Convolutional Neural Networks; FPGA; Hardware Acceleration; Parallel Computing.

1. Introduction

CPU and GPU as traditional computing platforms suffer from low computational efficiency and high-power consumption, which cannot meet the storage and computational needs of embedded devices. As a result, the field of image processing and algorithm acceleration uses a lot of semi-custom AI circuits [1]. FPGA development platforms are favored for their programmability, high performance, and low power consumption, and are suitable as dedicated hardware platforms for neural network acceleration [2]. There is a large scope for advancement in both algorithms and hardware to implement neural network models for computation and storage on mobile and embedded devices.

There are two main methods for improving deep learning in terms of algorithms: model compression and designing lightweight backbone networks. The main improvement methods for model compression include knowledge distillation, pruning, quantization, etc [3]. For example, Han et al. proposed iterative pruning of the network to obtain a more streamlined model, reducing the amount of AlexNet parameters by a factor of nine with no loss of accuracy [4]. Lightweight algorithms such as SqueezeNet, MobileNets [5], Xception [6], and other lightweight algorithms have been used to design lightweight backbone networks, greatly reducing the amount of computation required.

At the hardware level, CPUs struggle to support large-scale convolutional operations, so processors using parallel architectures are better able to implement the functionality. FPGAs with more resources are better able to customize reconfigurable convolutional IP [7]. For example, Zhang et al. propose to efficiently utilize bandwidth by increasing the burst transfer length, Shen et al [8]. propose to design multiple accelerators for different hierarchies, Li et al [9]. propose parallel

architectures that expand the output channels as well as the convolutional cores in three dimensions, etc. al [10]. To avoid the problem that CPUs cannot fully exploit the internal parallelism of convolutional neural networks, FPGAs are better at mapping algorithms to the hardware level for acceleration, allowing the individual hardware modules to execute in parallel. the FPGA's flow structure and hardware input-output connections can run convolutional neural network algorithms more efficiently.

In combination with the characteristics of its hardware architecture, there are currently two main approaches to FPGA development: Register Transfer Level (RTL) description and High-Level Synthesis (HLS) description. The advantages of RLS-level development include high stability, high resource utilization, and high performance, but there is a high barrier to development due to its greater difficulty and cost. Therefore, FPGAs prefer to use HLS-level development.

Image detection and recognition direction is a typical application scenario for deep neural networks. Most companies use FPGAs as the hardware platform for image recognition to reduce the price and power consumption of the product and improve stability and speed. The application scenarios of FPGA products based on image recognition are becoming more and more refined and specific in terms of usage scenarios, and more research directions and hotspots are urgently needed.

The experiments are based on the classical Yolov3 model, introducing the CBAM attention mechanism and pruning using Network slimming to design a more efficient FPGA hardware acceleration scheme and compare it with the original Yolov3 model. The hardware-accelerated system is experimented with the COCO dataset and deployed on a TX2 embedded device. In the experiments, the authors pruned channels with smaller incoming weights in the trained CNN, fine-tuning the network to regain accuracy. Moreover, the authors explicitly enhanced the sparsity of the channels in the optimization algorithm, using different pruning thresholds, 40%, 50%, and 60%, to find more efficient acceleration methods to make the construction process smoother. The experimental results show that after deploying the CBAM attention mechanism, the accuracy and the mAP value of the model are increasing, and it can extract the features of the image more accurately. At the same time, pruning with different scales by the Network slimming method is also able to reduce the number of parameters effectively without decreasing the accuracy, which speeds up the model and achieves the expected results.

2. Convolutional Neural Networks

Convolutional neural networks originate from research in neuroscience. Since the 1980s, research on convolutional neural networks has gradually become intensive. CNN has made advancements in artificial intelligence and computer vision research

As a classical supervised learning algorithm, CNN uses a backward approach for training and a feed-forward approach for recognition. Many designers train CNN offline in industrial practice, then use the offline trained CNN to carry out time-sensitive jobs. Therefore, it is the speed of the feed-forward computation that is most important. This paper uses FPGA-based accelerator designs to speed up the feed-forward computation. A classifier and a feature extractor make up a standard CNN. The input image is filtered by the feature extractor into a "feature map" that represents the many features of the image. These features could have angles, lines, corners, and so forth. A vector containing these features is the feature extractor's output.

The convolutional layer is computed by first receiving N feature maps as input to the convolutional layer. Each input feature map is convolved by a K x K convolution kernel to produce one pixel in an output feature map. The span of the shift window is S. A total of M output feature maps will form the set of input feature maps for the next convolutional layer.

CNN is constructed from a great number of neurons performing a dot product operation with weights and requires backpropagation for training and forward propagation for testing. Unlike traditional recognition algorithms, CNN can use images for their input data and no longer require feature extraction and data reconstruction. Under these conditions, the forward propagation of CNN

is equivalent to two-dimensional convolution, which means that CNN is highly deterministic and parallel to facilitate their speed-up. The classical CNN model consists of many different layers on which the feature map is sequentially computed, with the next layer reading the input features from the output of the previous layer, and then executing and outputting them. Commonly used CNN layers include a convolutional layer, a pooling layer, a fully connected layer, a ReLU layer, and a classification layer.

The convolutional layer is the core layer of building a convolutional neural network, which produces most of the computation in the network. Its parameters are composed of a collection of learnable filters. The convolutional layer can effectively reduce the number of parameters and prevent overfitting due to too many parameters. The role of the pooling layer is to gradually reduce the number of parameters in the network by decreasing the spatial size of the data, which makes the computational resources consumed less and also effectively controls overfitting. The main function of the fully connected layer is to integrate the features obtained from the convolutional and pooling layers into one value, which has the advantage of reducing the influence of feature position on the classification result and improving the robustness of the whole network. The ReLU layer can avoid the gradient explosion and gradient disappearance problems, and simplify the computation process.

The BN layer is special in the neural network whose role is to solve the problem of changing data distribution in the intermediate layers during the training process. It can improve the gradient of the network and prevent the gradient from disappearing or exploding, thus speeding up the training. Currently, the BN layer has been widely adopted by most modern convolutional neural networks. It is a standard method to accelerate network convergence and obtain better generalization performance. In this paper, the principle of the BN layer is used in Network slimming. The authors believe that the final result can be kept constant by reducing the value of the scaling factor and the weights of the convolutional layers. The effect of a scaling layer will be completely masked by the BN layer if a scaling layer is inserted before the BN layer. If a scaling layer is inserted after a BN layer, then for each channel there will be two consecutive scaling factors. This results in a CNN model with fewer parameters, small runtime memory, and compactness.

3. Hardware Acceleration

The author presents a design approach to FPGA hardware acceleration, which is presented through three main sections: algorithm optimization, dataset, and experimental environment. In algorithmic optimization, the workings of the Yolov3 algorithm are presented in a major way. Also, the authors study the CBAM attention mechanism and the way Network slimming works. Then, the authors explain the improvements that have been made to the Yolov3 algorithm. Finally, the data set and experimental environment sections are also briefly explained.

3.1 Algorithm Optimization

Yolov3 is a popular deep learning algorithm for image recognition and classification in recent years. Yolov3 algorithm is fast and efficient in detecting within the accuracy range, but there is still some room for improvement and optimization. Pruning of the algorithm is an effective way to improve efficiency. The ranking metric of contribution can be the average of the regularization of the weight parameters L1 and L2 of the neurons, the average output value of the activation function, and the number of times it is not zero on the validation dataset or other metrics. There will be some loss in the accuracy of the model if pruning these neurons with a low contribution. Therefore, more training is usually required to ensure that the pruned model still has some performance. If too many neurons are pruned at once, the model may become too corrupted and the performance becomes worse, but it is still an effective means to reduce the complexity of the network and reduce overfitting.

Convolutional Block Attention Module (CBAM) is a new approach to data processing in machine learning proposed by Sanghyun Woo et al. in [11]. It focuses on an attention mechanism in a network

architecture. It is a lightweight attention module that can perform attention operations on features in the channel and spatial dimensions, which is presented in Figure 1.

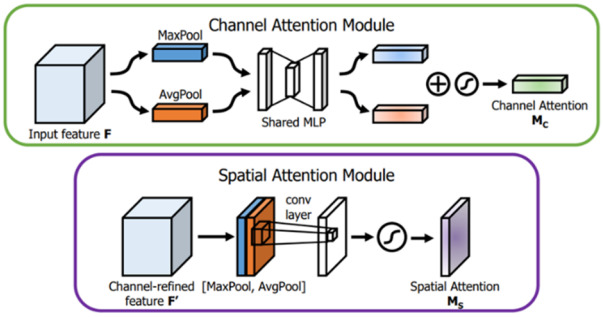

Fig 1. Principle diagram of CBAM

This attention mechanism is mainly used to find local information about the image and to refine the feature map adaptively, it can find the more deserving feature map in the middle of the model, and make the model pay more attention to this area. Sanghyun Woo implemented CBAM in the ResNet50 model and found that the activation maps using the CBAM module were more accurate. Therefore, the CBAM module was added to yolov3 for training in this paper.

Network slimming is an effective pruning strategy, which was proposed by Zhuang et al. in [12]. It uses the scaling factor γ in the BN layer to measure the importance of channels in the training process, and the unimportant channels are removed to achieve the effect of compressing the model size and improving the computational speed. As shown in Figure 2, the left half is the model under training, and the middle column is the scaling factor γ in the BN layer, when γ is small, the corresponding channel will be reduced to obtain the model shown in the right half.

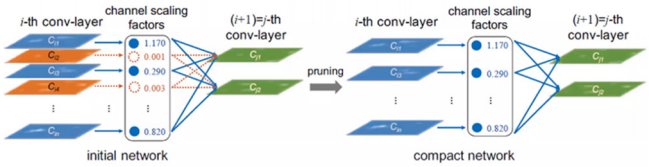

Fig 2. Principle diagram of Network slimming

Zhuang et al. found that in several models of VGGNet, DenseNet-40, and ResNet-164, the parameter compression rates and FLOP compression rates were more pronounced until after pruning exceeded 80% when the accuracy of the models dropped more significantly. Therefore, it can be said that Network slimming is an effective way to reduce the number of parameters in the model.

In this paper, the author performed a similar pruning optimization of the Yolov3 algorithm. The author obtained the overall covariance of the sparse model after sparse regularization training and subsequently trim the channels with scale factors close to zero. The author pruned the channels using different γ as threshold values in all layers. The author chose 40%, 50%, and 60% of the threshold values of γ and performed different pruning operations to obtain more compact networks with fewer parameters, shorter computation time, and less runtime memory.

3.2 Dataset

This experiment used the classical COCO dataset to verify the accuracy of FPGA for image recognition. The COCO dataset includes 80 common item categories and is a large-scale dataset for image detection, semantic segmentation, and image caption generation. It has more than 330K images containing 1.5 million targets for 91 material categories, each image contains five image utterance descriptions and 250,000 pedestrians with key point annotations.

3.3 Experimental Environment

The TX2 embedded device developed by Nvidia was used in this experiment. It has 6 CPU cores, consisting of a dual-core Denver2 processor and a quad-core ARM Cortex-A57, and is equipped with 256 CUDA cores. Its 8G of running memory and 32G of actual memory are capable of fast calculations. The CPU cores have five power modes, enabling developers to take full advantage of the tensor cores and Deep Learning Accelerator (DLA) units in the GPU.

4. Experimental Results

This experiment was conducted on a TX2 embedded device running with the COCO dataset, deploying the CBAM attention mechanism and using Network slimming as the pruning strategy, using thresholds of 40%, 50%, and 60% γ. The results of the experiments are illustrated in Table 1.

Table 1. Experimental results

	precision	***mAP***	***Parameters***
Initial Yolov3	88.4%	36.4	8.21M
Yolov3+CBAM	90.3%	37.8	8.27M
γ=40%+CBAM	93.8%	42.1	4.28M
γ=50%+CBAM	94.4%	43.5	4.06M
γ=60%+CBAM	94.0%	43.3	3.93M

The table shows a significant increase in model accuracy after the deployment of the CBAM attention mechanism, with a 1.2% improvement in mAP. After pruning with three different γ values, the number of parameters was reduced while keeping the accuracy largely unchanged.

5. Conclusion

In this experiment, the model of yolov3 was optimized and pruned. Also, the attention mechanism of CBAM was added and the target detection was performed in TX2 embedded device using the COCO dataset, which effectively improved the target detection of the model. In addition, the Network slimming method was used to optimize the pruning of the Yolov3 model and found that the threshold value γ is between 40% and 60%, which can reduce the number of parameters as much as possible without degrading the accuracy, effectively shorten the computation time and reduce the memory occupied by the operation, which has a greater advantage in the design of the embedded terminal and has a strong value for further research and extensive research. It has a strong value for in-depth research and wide application prospects.

References

[1] J Y GUO, C M SHAO, J WANG, et al. Programming and development environment for FPGA graph computing: Overview and exploration[J]//Computer Research and Development, 2020, 57(6): 1164-1178. (In Chinese)

[2] C C XU. Research on FPGA-based Graph Computing Accelerator System [D]//Hefei: University of Science and Technology of China,2018. (In Chinese)

[3] Q Q GAO, Y ZHAO, G LI et al. a knowledge distillation based super-resolution convolutional neural network compression technique [J]// Computer Applications, 2019 (10). (In Chinese)

[4] S HAN, J POOL, et al. Learning Both Weights and Connections for Efficient Neural Networks[C] //28th International Conference on Neural Information Processing Systems, Montreal: NIPS, 2015: 1137-1141.

[5] A G HOWARD, M ZHU, B CHEN, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [J / OL]. [2021-05-10]. https://arxiv.org/abs/1704.04861.

[6] F CHOLLET. Xception: Deep Learning with Depthwise Separable Convolutions[C] // IEEE Conference on Computer Vision and Pattern Recognition. Honolulu: IEEE, 2017: 1251-1258.

[7] L YU, R W LI, Y WANG, et al. Overview: Open-Source Processors for SoC-FPGAs [J] // Journal of Electronics, 2018, 46(4): 992-1004. (In Chinese)

[8] Z CHEN, Z FANG, P ZHOU, et al. Caffeine: Towards Uniformed Representation and Acceleration for Deep Convolutional Neural Networks[C] //. International Conference on Computer-aided Design. Austin: IEEE, 2016: 1-8.

[9] Y SHEN, M FERDMAN, P MILDER. Maximizing CNN Accelerator Efficiency through Resource Partitioning [C] //44th Annual International Symposium on Computer Architecture. Toronto: IEEE, 2017: 535-547.

[10] H LI, X FAN, J LI, et al. A High-Performance FPGA-based Accelerator for Large-scale Convolutional Neural Networks [C]//26th International Conference on Field Programmable Logic and Applications. Lausanne: IEEE,2016: 1-9.

[11] S WOO, J PARK, J Y LEE, et al. CBAM: Convolutional Block Attention Module[C]// European Conference on Computer Vision. Springer, Cham, 2018.

[12] L ZHUANG, J Li, Z SHEN, et al. Learning Efficient Convolutional Networks through Network Slimming [C]// 2017 IEEE International Conference on Computer Vision (ICCV). IEEE, 2017.