Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 3, No. 1, 2023

Design and Implementation of Convolutional Neural Network Accelerator Based on FPGA

Jixun Cheng*

School of Physics & Electronic Information Engineering, Henan Polytechnic University, Jiaozuo 454000, Henan, China
* **Corresponding author Email:** 849061591@qq.com

Abstract: Convolutional neural network, as a kind of feed-forward neural network, has been widely used in image recognition, speech processing and other fields in recent years. In this paper, an FPGA-based CNN gas pedal is designed to solve the problem of slow running and high-power consumption of CNN on resource-constrained hardware. The design invokes multi-stage pipeline parallel processing technology to accelerate convolutional operations; quantifies network parameters from 32-bit floating-point to 8-bit fixed-point while guaranteeing CNN accuracy, and uses data multiplexing to reduce resource consumption. Experimental results show that the design is 10 times faster than the intel i7-8700 and consumes only 1% of the power of the RTX 2060 at 50MHz.

Keywords: Field Programmable Gate Array; Convolutional Neural Network; Parallelization; Accelerator.

1. Introduction

With the progress of time and technology, Deep Learning (Deep Learning) technology has been developed and widely used in different application scenarios [1]. And convolutional neural networks, as a common deep learning structure with two features of local perception and parameter sharing, are structurally closer to actual biological neural networks, and have achieved good results not only in the application of computer vision fields such as image classification [2][3], target recognition, and video analysis [4][5], but also in natural language processing [6], speech recognition [7], and other fields as well breakthroughs in natural language processing [6], speech recognition [7], and other fields, which have greatly advanced the development of artificial intelligence.

However, in the continuous pursuit of better performance, the number of parameters and computation of algorithmic models are also increasing rapidly, making the traditional general-purpose processors more and more strained for network models. More and more researchers are designing structures for hardware acceleration using other platforms to address the structural characteristics of convolutional neural networks.

Currently, there are three types of platforms for convolutional neural network acceleration: Graphics Processing Unit (GPU), Application Specific Integrated Circuit (ASIC) [8], and Field Programmable Logic Array (FPGA). GPUs have powerful parallel computing capabilities for large-scale, same-type operations [9], which are well suited for CNN operations, but their power consumption is too high ASICs, as specialized integrated circuits, can be customized with dedicated acceleration circuits for convolutional neural network models to achieve high throughput and energy efficiency [10]. However, the design process has many parts, takes long time, and is expensive to design. Finally, FPGAs, with low power consumption, short development cycle, and high flexibility, can have higher computational performance and energy consumption ratio than CPUs and GPUs, and are more reconfigurable and less expensive to develop than ASICs, while FPGAs are more suitable for hardware acceleration research as a functional simulation and verification platform before the flow of AI chips [11].

To address this, this paper designs a fast and low-power convolutional neural network gas pedal based on FPGAs by conducting an in-depth study of CNN algorithms to discover the potential parallelism of convolutional neural network networks, and taking into full consideration the advantages of rich logic resources and flexible design of FPGAs themselves.

2. CNN

Convolutional neural network is a very important research branch in the field of neural networks, which is a very classical forward propagation neural network [12], including input layer, transmission layer and hidden layer. The input layer is the input data of the whole neural network; the output layer generally adopts a fully connected approach, in which the extracted features are analyzed according to different weight values to draw conclusions; and the hidden layer consists of three main parts: convolution, pooling and activation [13]. The convolution layer is used to convolve the original image with different convolution kernels to extract the features from the original image. The pooling layer, also known as the down sampling layer, is mainly used to reduce the dimensionality of the feature map and compute the statistical features within each pooling window, compressing them into one value to achieve the effect of reducing the amount of data, preventing overfitting, and speeding up the operation. Commonly used pooling methods are maximum pooling and average pooling. The activation layer is to introduce a nonlinear factor to the network structure to solve the problem of insufficient expression and classification ability of linear models.

The features of each layer of the CNN are obtained from the local but not the global region of the previous layer by the same convolutional kernel excitation with shared weights, as shown in Equation (1) [14], which on the one hand reduces the number of weights making the network easy to optimize, and on the other hand reduces the complexity of the model and reduces the risk of overfitting. This advantage is even

more obvious when the input is an image, as the image can be directly used as the input to the network, avoiding the complex feature extraction and data reconstruction process in traditional recognition algorithms, which has a great advantage in the processing of two-dimensional images [15].

$$x_j^l = f\left(\sum_{i=l}^{m} x_i^{l-1} \otimes k_{ij} + b_j\right) \quad \text{j=1, 2,…,n} \qquad (1)$$

The neural network model used in this paper is the modified LeNet-5 model, and its model structure is shown in Figure 1.

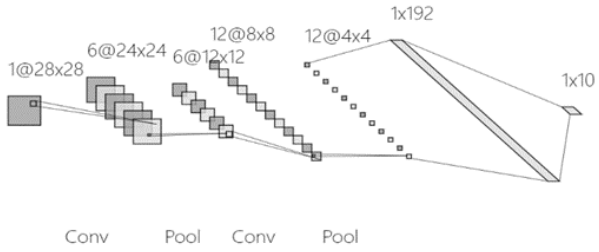

Figure 1. Improved LeNet-5

The model has two convolutional layers and two pooling layers, with a convolutional kernel size of 5X5. The results of the second pooling layer are expanded into a (192, 1) column vector, and 10 results are obtained after a fully connected layer, whose maximum value is the final result of the corresponding handwritten digit prediction.

In this paper, the model is built and trained on jupyter notebook based on the pytorch deep learning framework.

In this model, the input of the first convolutional layer is 28 X 28 pixels after grayscale processing and normalization, the size of convolutional kernel is 5 X 5, the number of convolutional kernels is 6, step size = 1, padding = 0, the activation function is ReLU activation function, and the size of output of each channel is 24 X 24.

The input of the first pooling layer is 24 X 24, and the output is 12 X 12 after maximum pooling of size 2 X 2 and step size 2.

The input of the second convolutional layer is six 12 X 12 feature maps with 5 X 5 convolutional kernels, 12 convolutional kernels with a step size of 1 and padding = 0. The activation function is chosen as the ReLU function, and the size of the output of each channel of this layer is 8 X 8.

The size of the input of the second pooling layer is 8 X 8, and the size of the output is 4 X 4 after a maximum pooling of size 2 X 2 and a step size of 2.

A total of 12 4 X 4 feature maps are obtained after the second pooling layer, and a (192, 1) column vector is obtained after expansion.

A total of 10 neurons in the fully connected layer are computed with the above (192, 1) column vector for full connection, and finally 10 computational results are obtained.

3. Design options

3.1. Overall structure

The parallelism of the convolution part in this design is 6, i.e., 6 convolution modules, activation modules and pooling modules are instantiated. The simplified LeNet model has 6 convolutional kernels in the first layer and 12 convolutional kernels in the second layer. If the 6 convolutional modules are used to obtain a complete convolutional result (N X N) at the same time, then the entire network needs 13 convolutional cycles to recognize a picture.

The first convolutional cycle: input the pixel data of the image, the calculation result of each convolutional module plus the bias of the corresponding convolutional kernel enters the pooling module through the activation module, and finally obtains 6 groups (144 per group) of data, i.e. 6 feature maps of size 12 X 12 in the first layer, which are recorded as F1 to F6. At this time, the calculation of the first convolutional layer and the pooling layer is completed.

The 2nd convolutional cycle: F1 is selected as the input of the 6 convolutional modules, and finally 6 groups of 64 data are obtained, which are recorded as F1_01 to F1_06.

The 3rd convolution cycle: F1 is still selected as the input of the 6 convolution modules, and 6 sets of data (64 in each set) are obtained, which are recorded as F1_07 to F1_12.

The 4th convolution cycle: F2 is selected as the input of the 6 convolution modules, and 6 sets of data (64 each) are obtained, which are recorded as F2_01 to F2_06.

The 5th convolution cycle: F2 is still selected as the input of the 6 convolution modules, and 6 sets of data (64 each) are finally obtained, which are recorded as F2_07 to F2_12.

The next loop is similar to the above. So far, 72 groups of data are obtained, and the 12 groups of data obtained after adding (example: F1_01 + F2_01 + F3_01 + F4_01 + F5_01 + F6_01) are the inputs of the activation module in the second convolutional layer, and then 12 groups of data (16 per group) are obtained after pooling module, which are the inputs of the fully-connected layer.

The above is the entire computation process except for the fully connected layer, from which the hardware circuit is designed and accelerated by pipelining method.

The results of the 2nd and 3rd convolution cycles are stored in 12 FIFOs (FIFO_01 to FIFO_12), and in the 4th convolution cycle, the data in the corresponding FIFOs (FIFO_01 to FIFO_06) are summed and written back to the same FIFO. The subsequent convolution cycles are similar, except that in the 12th and 13th convolution cycle, the result of the summation is not written back to the FIFO but flows into the activation module.

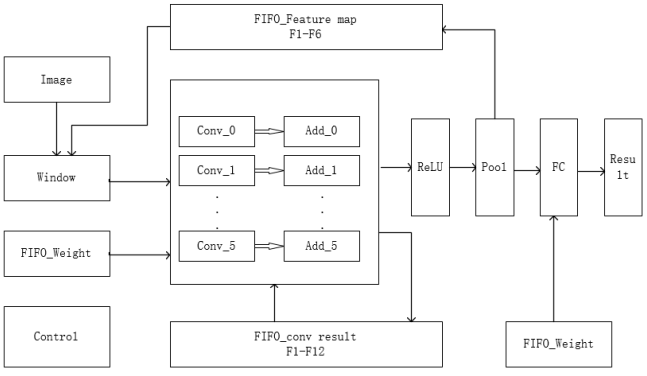

Figure 2. Overall framework diagram

3.2. Model quantification

Neural network model in GPU, CPU do are floating point calculation, considering to reduce the size of the model, improve the speed of calculation, after research will be more appropriate to convert floating point to INT8, its accuracy does not lose too much, and in the FPGA, implementation will save a lot of resources used to improve the parallelism of the design, improve the acceleration effect. In this design, the model is quantized using static quantization after training and hierarchical quantization.

The INT8 quantization is to map the weight parameters of each layer between -127 and 127, and the simple max-max

mapping is to map the negative value of the maximum absolute value of the parameters to -127 and the positive value to 127. This method will greatly lose accuracy in the case of uneven distribution of parameters, so we need to observe the distribution of parameters first and choose the appropriate mapping method.

First extract the parameters from the model, visualize them as histograms, and observe the distribution of the parameters. We can see that the weight parameters of each layer are mostly concentrated in a certain interval, for example, the weight parameters of the second convolutional layer are concentrated between [-0.4,0.4], and there are few parameters outside this interval. Therefore, when quantifying, the parameters outside this interval are rounded off, and -0.4 is mapped to - For the first convolutional layer, the parameters in the interval [-0.6,0.6] are mapped to [-127,127]. For the fully connected layer, the parameters in the interval [-0.4,0.4] are mapped to [-127,127], and the discarded parameters are taken as -127 or 127.

3.3. Convolution module

In this model, all convolution kernels are of size 5 X 5 with step size 1 and padding = 0. Therefore, the convolution module can be reused in both layer 1 and layer 2 convolution loops. Although the input sizes of layer 1 and layer 2 are different, they can be controlled by sampling means without additional circuitry.

In this paper, a sliding window approach is used to implement a 5 X 5 matrix multiplication. The sliding window module is implemented by shift ram, as shown in Figure 3. The serial data flows in from shift_in, shifts right at each clock arrival, and outputs the tap vector when it reaches the end of the line and sequentially goes to the beginning of the next line. Once the entire shift_ram is filled, each subsequent clock causes the window to pan one unit to the right.

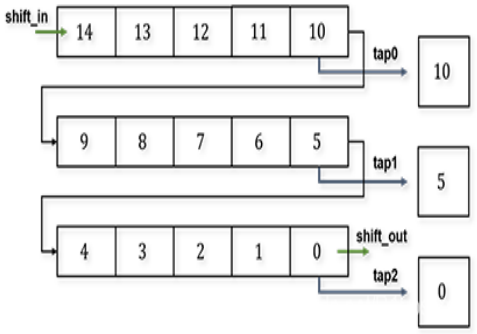

Figure 3. Sliding window

When calculating the first and second convolutional layers, the input size of the sliding window module is different, 28 X 28 for the first layer and 12 X 12 for the second layer, which can be achieved by controlling the address of the output data with the same shift ram.

The size of the convolution kernel is 5 X 5, and each convolution kernel has 25 weight parameters stored in a 5 X 5 array of registers. The output of the sliding window module is passed forward in order to calculate the multiplication and addition result of each column, and the sum of the 5 multiplication and addition results is an output result of the convolution module. The appropriate sampling rules are set according to the convolution step size and the input size of each layer.

3.4. Activation and pooling module

The activation function used in this paper is the ReLU activation function, which is constant when the input is positive and takes 0 when it is negative. the pooling method is 2X2 maximum pooling with a step size of 2, and its implementation is shown in Figure 4.

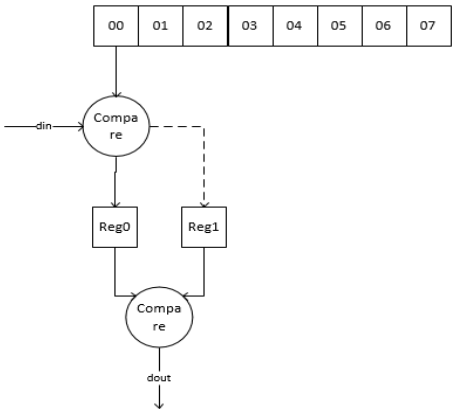

Figure 4. Pooling calculation

The pooling process is similar to a ping-pong operation, where one row of input data is first cached, and when the first data in the second row is sampled, it is compared with the first data in the first row of the cache, and the result is stored in Reg0; when the second data in the second row is sampled, it is compared with the second data in the first row of the cache, and the result is stored in Reg1; when the third data in the second row is sampled, it is compared with the third data in the first row of the cache, and the result is stored in Reg0, and so on. The result of the comparison between the data in Reg0 and Reg1 is the output of the pooling module. The output of the pooling module designed by this method is a maximum pooling result with a step size of 1, and the output needs to be filtered.

3.5. Full-connected module

The number of multiplications and additions in the fully connected layer of the convolutional neural network is large, accounting for almost half of the multiplications and additions in the whole computation process. In order to strengthen the acceleration effect, most of the resources in the FPGA are used for the design of fully connected modules in this design, with a parallelism of 10 and no multiplexing of modules.

The handwritten digit recognition is a 10-category image classification calculation, and there are 10 neurons in the fully-connected layer, so 10 fully-connected modules are designed, corresponding to 10 neurons in the fully-connected layer.

In this design, the parallelism of convolution module, activation module and pooling module is 6. At the 12th convolution cycle, the pooling module outputs 6 valid data at the same time, and these 6 valid data are input to each fully connected module for multiplication and addition. Each fully-connected module performs 192 multiplications, requiring 192 weight parameters, 96 for the 12th convolutional cycle and 96 for the 13th convolutional cycle.

In the 12th convolution cycle, the nth valid data output from the six pooling modules are multiplied and summed with w(0+n), w(16+n), w(32+n), w(48+n), w(54+n), and w(80+n), respectively. The output size of the second pooling layer is 4 X 4, so the input valid signal of each fully connected module

will be pulled up 16 times, and the result obtained each time will be cumulated plus the previous result, and after 16 times, the cumulative result will be output and recorded as result0. At the 13th convolution loop, the nth valid data output by the six pooling modules are compared with w(96+n), w(112+n), w(128+ n), w(144+n), w(160+n), w(176+n), and the final output result1. Result0 and result1 are added together to obtain the final output of the neuron. At this point, the acceleration module completes an acceleration process.

4. Experiments and analysis of results

The training process of the CNN network is done on the PC, and the obtained training weights are fixed-point for backup on the PC. The FPGA circuit implementation platform is the HAN Pilot Platform development platform, and the device used is the 10AS066K3F40E2SG of the Arria 10 series, and the development environment used is Quartus Prime 18.1. The internal resource utilization of the FPGA is shown in Table 1. It can be seen that the overall design takes less resources from the development board, so the gas pedal can be used in FPGAs with fewer resources. If higher computational performance is required, the DSP utilization can go to the appropriate level.

Table 1. Resource usage

Category	Resource usage	Occupancy rate
Logic Utilization	56995	23%
RAM Blocks	30	1%
DSP Blocks	90	5%

The digital recognition library is chosen to use the MNIST dataset, this dataset has 60,000 images in the training set and 10,000 images in the test set, each image is a 28 x 28 grayscale image, the images are fixed point and then imported into the FPGA together with the weights to complete the recognition.

The current maximum operating frequency of this design is up to 130MHz, and the experiment was run at 50MHz, and the results are shown in Table 2. The design recognizes 10,000 images at 50MHz with 96.1% recognition accuracy, which meets the error requirements, and its computing speed is 10 times faster than the intel i7-8700 and power consumption is only 1% of the RTX 2060.

Table 2. Comparison of this design with CPU and GPU

Category	Calculating time/us	Power Consumption/w	Accuracy rate/%
CPU	879	95	98.70
GPU	230	160	98.57
Design	85	1.463	96.10

5. Conclusion

This design makes full use of the high parallel processing capability and low power consumption of FPGAs, and completes the overall design of the FPGA-based convolutional neural network gas pedal. By using FPGA on-chip memory resources, the parallelism of the circuit is improved, and the pipeline structure improves the computation speed and data throughput. The experiments show that the design has good application prospect with high recognition accuracy, fast operation speed, and only 1.463w power consumption.

The design is currently hardware accelerated only for specific convolutional neural networks, and in the next step, more possibilities for its generality can be explored.

References

[1] Mao YH, Gui XL, Li QIAN, He XS. Research on deep learning application techniques [J]. Computer Application Research, 2016, 33(11): 3201-3205.

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

[3] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 1–9.

[4] Xu Z, Yang Y, Hauptmann A G. A discriminative CNN video representation for event detection [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 1798–1807.

[5] Karpathy A, Toderici G, Shetty S, et al. Large-scale video classification with convolutional neural networks [C]. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2014: 1725–1732.

[6] Dos Santos C, Gatti M. Deep convolutional neural networks for sentiment analysis of short texts [C]. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. 2014: 69–78.

[7] Abdel-Hamid O, Mohamed A-r, Jiang H, et al. Convolutional neural networks for speech recognition [J]. IEEE/ACM Transactions on audio, speech, and language processing, 2014, 22 (10): 1533–1545.

[8] Jouppi N P, Young C, Patil N, et al. In-Datacenter Performance Analysis of a Tensor Processing Unit [J]. 2017.

[9] Patterson, David A, Hennessy, John L, Goldberg, David. Computer architecture: a quantitative approach[M]// Computer Architecture: A Quantitative Approach. Morgan Kaufmann Publishers Inc. 2008.

[10] Zhao BY. Research on the design and implementation of hardware accelerator based on convolutional neural network[D]. Harbin Institute of Technology,2018.

[11] Ye J, Hu Y, Li X. Hardware Trojan in FPGA CNN Accelerator[C]//2018 IEEE 27th Asian Test Symposium (ATS). IEEE, 2018: 68-73.

[12] Gong J, Zhao S, He h, Deng N. FPGA-based quantized CNN acceleration system design[J].Computer Engineering ,2022,48(03):170-174+196.

[13] Ahmad S, Sadiq M, Aiman E. FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review. [J]. IEEE Access, 2019, 7: 7823-7859.

[14] MIYAMA M. FPGA Implementation of 3-Bit Quantized Multi-Task CNN for Contour Detection and Disparity Estimation[J]. IEICE Transactions on Information and Systems,2022, E105.D(2).

[15] Chen Y, Tushar K, Joel S, et al. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. [J]. J. Solid-State Circuits, 2017, 52(1): 127-138.