CHIROGRAPHY DIGIT IDENTIFICATION METHOD DEPEND ON XILINX USING CNN

B.Mysura Reddy[1], M.Usha Sree[2] , P.Tharun Teja[3]

[1]Assistant Professor, [2,3]Student
Electronics and Communication Engineering
N.B.K.R Institute of Science and Technology, Andhra Pradesh, India

***Abstract*: A CNNs are widely used in the computer vision for different purposes such as image classification and target detection, the total number of parameters and computation of the models is gradually increasing. Therefore, the main objective of this project is CNN processor design with an FPGA based resource multiplexing and power consumption of CNNs.**

In this paper we observe the variation of different algorithms that can classify the hand – written number using various hidden layers, different number of epochs and to make a comparison based on the accuracy of CNN. This is performed using the Modified National Institute of Standards and Technology (MNIST) dataset.

***Keywords*: FPGA, Parellel Processing, Hidden Layers, Epochs, MNIST dataset.**

1.INTRODUCTION

In last few years, some field Programmable gate array (FPGA)based accelerators of the testing phase of CNNs have been proposed. FPGAs are widely used on portable devices. They can be programmed to achieve high parallelism & provide good performance. The power consumption of FPGAs is lower than that of GPUs under the same workload. These reasons make FPGAs suitable for implementing the testing phase of a CNN. They can provide comparable runtime performance of testing to GPUs and achieve lower power consumption, which is critical in portable devices.

In this project, an FPGA based implementation of the testing phase of a CNN for number recognition is proposed. These projects mainly design to focuses on improving power efficiency & providing high performance.

2.CONVOLUTION NEURAL NETWORK

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer
- Fully-connected layer

The convolutional layer is the first layer of a convolutional network. while convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edge. As the image progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

In figure 1, the input of the CNN is an RGB image. The image is convolved by a filter, which also has three channels. Each channel of the image is convolved by the corresponding channel of the filter. The convolution results of all three channels are summed up to produce a new image, this image is called a feature map since it contains some features of the original image. Then the size of this feature map is reduced to 4×4. The values of the new feature map are used as the input of the fully connected layer. The following sub-sections describe each CNN layer in detail.

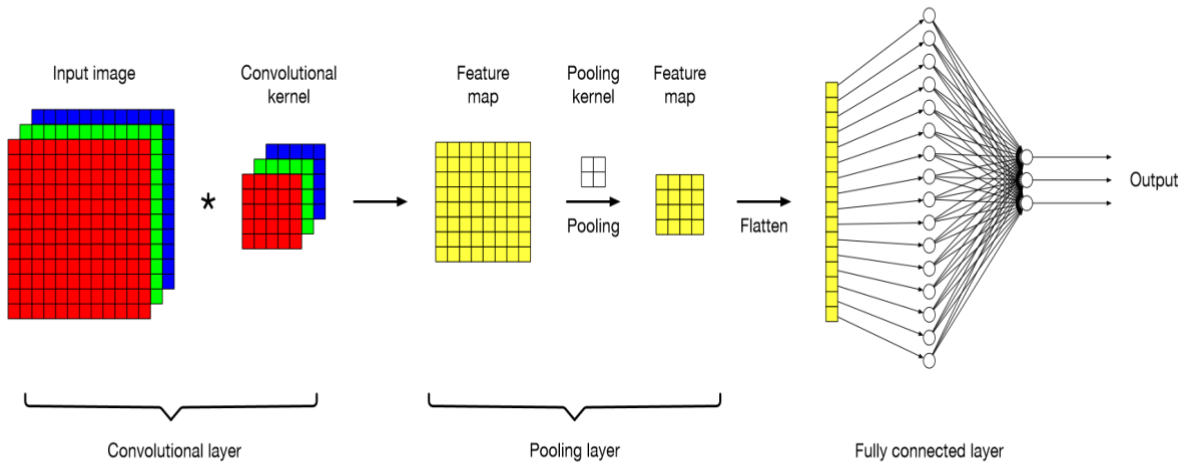

Figure 1: CNN with three layers

2.1:DESIGN OF CNN

The testing phase of the CNN needs to be deployed in the FPGA, the CNN architecture is designed to meet the accuracy of CNN prediction while keeping the amount of data and computation as small as possible. The CNN design of this system has the following features:

- Minimized number of filters per convolutional layer and fully connected layers without degrading classification performance, and use of small convolutional kernels of 3 × 3;
- Bias parameters are not used. Since the parameters need to be converted from floating point to fixed point into the FPGA for processing, the bias parameters are not used to avoid the accumulation of errors in the fixedpoint parameterization, thus reducing the impact on recognition accuracy;
- Using maximum pooling, GlobalAvgPooling is replaced with
- GlobalMaxPooling; it is easier to implement maximum pooling operations in hardware compared to average pooling;
- Use the simple activation function ReLU, as other activations, including division, power and other functions, are difficult to implement in hardware;
- Minimized number of heterogeneous layers to allow FPGAs to implement resource reuse and parallel-processing techniques for large amounts of data.

The CNN of this system is built using Python based on TensorFlow and Keras frameworks, mainly consisting of six convolutional layers, seven activation layers, three pooling layers and one fully connected layer. The CNN removes a large number of fully connected layers and bias terms, and the convolutional layers use all 3 × 3 convolutional kernels. The total number of parameters of the optimized CNN is reduced from 25,000 to about 4676, and the accuracy of the convolutional neural network reaches 97.3 percent on the MNIST dataset, as shown in Figure 2.

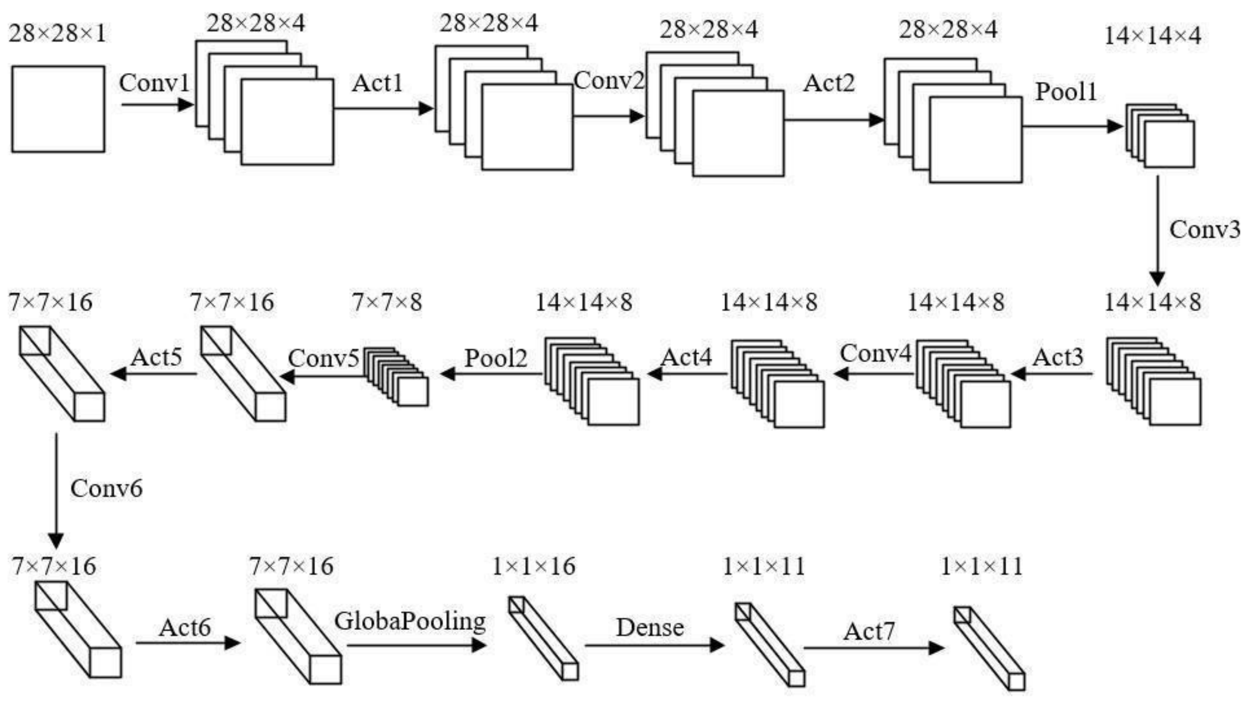

Figure 2: Convolutional neural network model

Among them, the total number of inputs, outputs, convolutional kernel sizes and parameters for each layer of the CNN model is shown in Table 1. To maximize the resource multiplexing architecture of the FPGA hardware resources, the convolutional kernel sizes of the convolutional layers are all 3 × 3, and the fully connected layer uses a small convolutional kernel of 1 × 1.

	Input	Output	The Size of the Convolution Kernel	The Number of Convolution Kernels	Total Number of Parameters
Conv 1	28 × 28 × 1	28 × 28 × 4	3 × 3	1 × 4	36
Act 2	28 × 28 × 4	28 × 28 × 4	/	/	/
Conv 2	28 × 28 × 4	28 × 28 × 4	3 × 3	4 × 4	144
Act 2	28 × 28 × 4	28 × 28 × 4	/	/	/
Pool 1	28 × 28 × 4	14 × 14 × 4	/	/	/
Conv 3	14 × 14 × 4	14 × 14 × 8	3 × 3	4 × 8	288
Act 3	14 × 14 × 8	14 × 14 × 8	/	/	/
Conv 4	14 × 14 × 8	14 × 14 × 8	3 × 3	8 × 8	574
Act 4	14 × 14 × 8	14 × 14 × 8	/	/	/
Pool 2	14 × 14 × 8	7 × 7 × 8	/	/	/
Conv 5	7 × 7 × 8	7 × 7 × 16	3 × 3	8 × 16	1152
Act 5	7 × 7 × 16	7 × 7 × 16	/	/	/
Conv 6	7 × 7 × 16	7 × 7 × 16	3 × 3	16 × 16	2304
Act 6	7 × 7 × 16	7 × 7 × 16	/	/	/
Global pool	7 × 7 × 16	1 × 1 × 16	/	/	/
Dense	1 × 1 × 16	1 × 1 × 11	1 × 1	16 × 11	172
Act 7	1 × 1 × 11	1 × 1 × 11	/	/	/
Total	/	/	/	/	4676

Table 1. Statistics of each parameter in the model

3.Hardware implementation of CNN on FPGA

The hardware part mainly consists of a Xilinx's A7 series XC7A35T chip as the main control chip, DDR3 chip as the cache buffer unit, OV5642 camera as the video image real time acquisition module, and LCD display as the graph display module and prediction result display module.

The overall block diagram of the system is shown as follows in Figure 3: first, the camera acquires image data, and the FPGA stitches the input image data into 16bit and outputs it to the frame cache controller; then, the image data is stored in the two addresses of DDR3 sequentially using a ping-pong operation, and the frame cache controller reads out the image data from DDR3 through AXI bus; finally, one way of the image data is displayed on the monitor in real time; the other way of the image data is sent to the image pre-processing module, and the processed 28 × 28 image is input to the convolutional neural network module for predictive classification, and the successful recognition result is displayed on the right side of the LCD in real time.

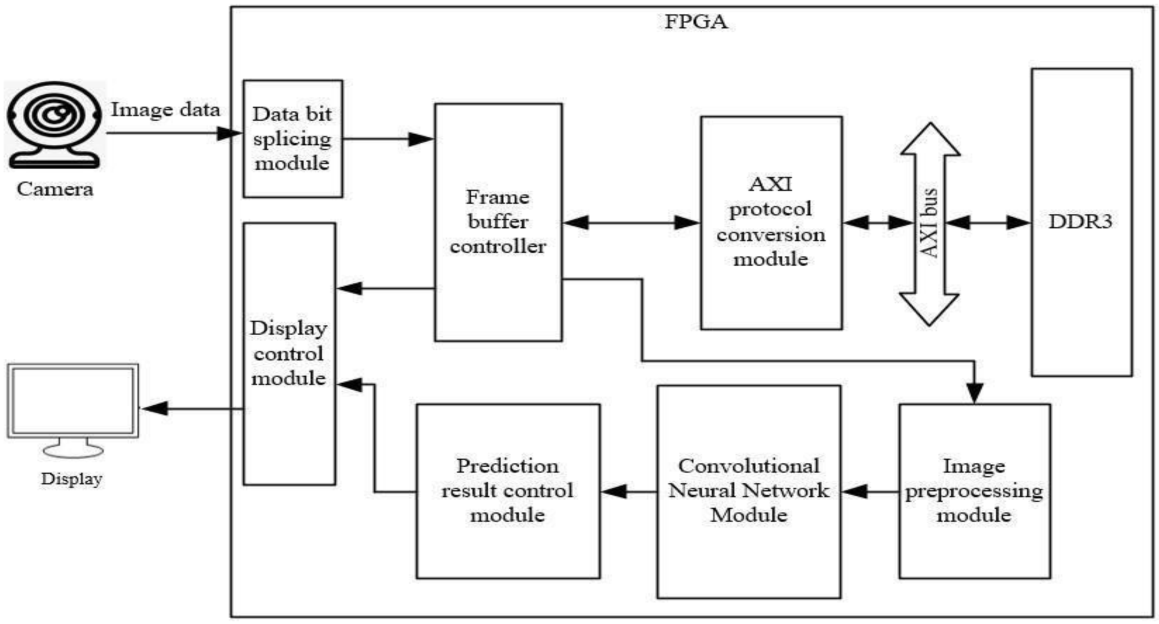
FPGA
Camera
Image data
Data bit splicing module
Frame buffer controller
AXI protocol conversion module
AXI bus
DDR3
Display control module
Display
Prediction result control module
Convolutional Neural Network Module
Image preprocessing module

Figure 3. Overall block diagram of the system.

4.FPGA IMPLEMENTATION

4.1 IMAGE PRE PROCESSING AND RESIZING

After acquiring the image, thresholding is applied and then converted into a binary image. The image is then dilated to reduce the artifacts caused by noise and then connected component labelling is applied to extract the characters from the image. The extracted images are cropped from the source image and then resized into a 24×24 image and then padded into a 28×28 image. Histogram normalization is also applied to help against illumination intensity variations.

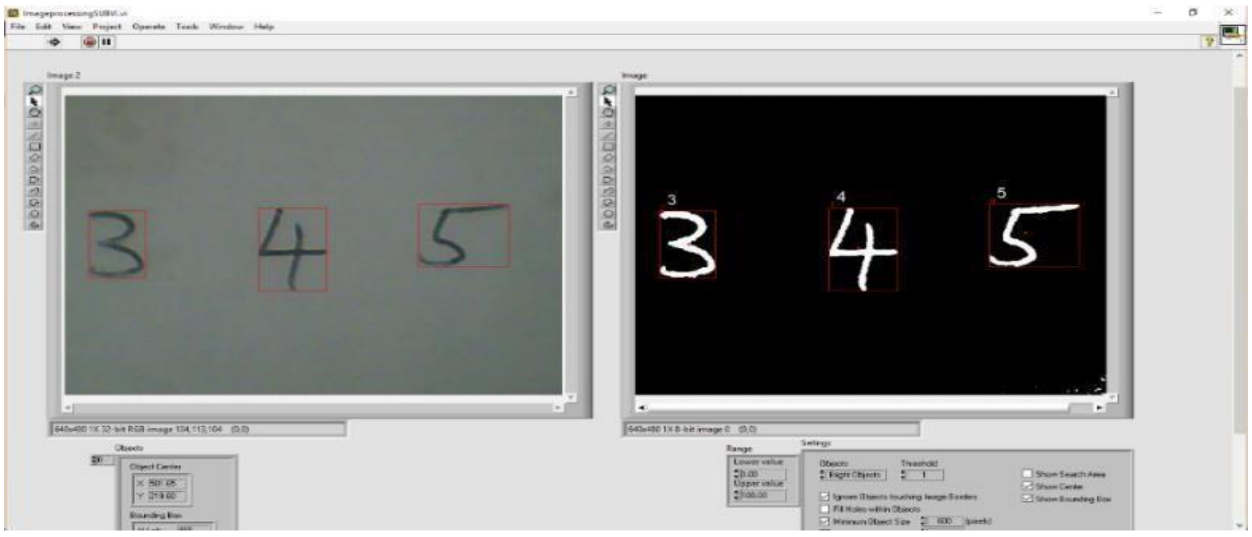

Figure 4: Image Segmentation and Pre processing

4.2: CONVOLUTIONAL LAYER

To maximize the resource reuse architecture, the convolutional layers designed in this system are all 3 × 3 convolutional kernels, which avoids the design of corresponding structures for convolutional kernels of different sizes and makes the designed computational modules universal. In view of the fact that the same layer architecture will occupy a lot of extra hardware resources, this system adopts a resource reuse architecture to reuse the same convolutional layer unit, which greatly reduces the consumption of hardware resources. The structure diagram before and after resource multiplexing is shown in Figure 5.

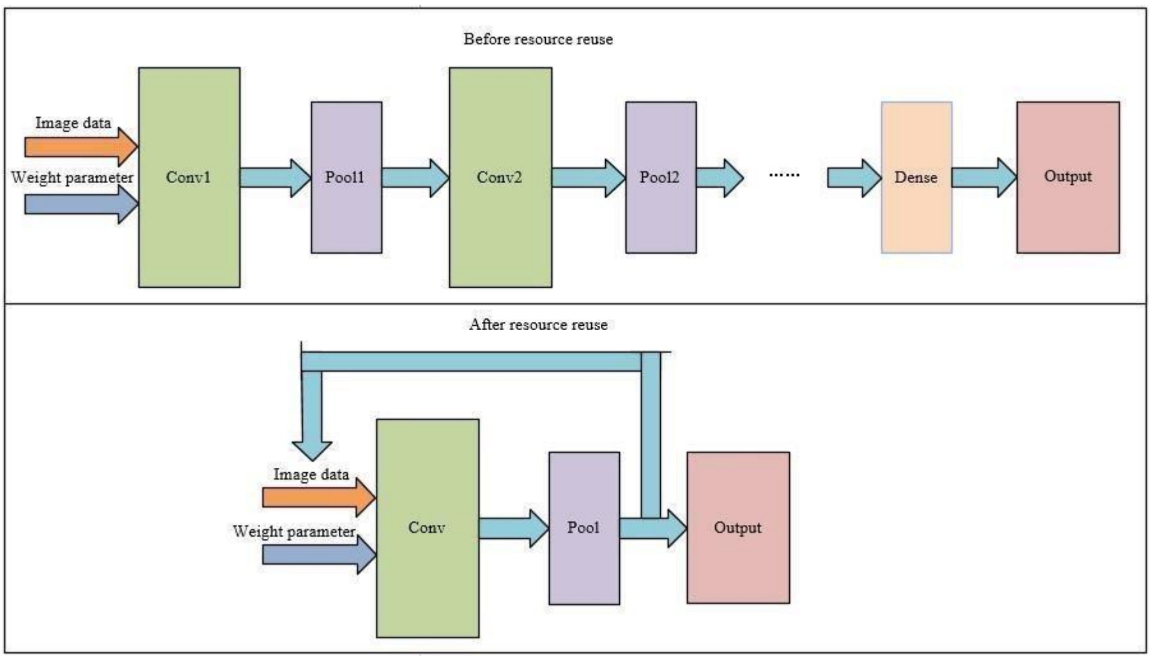

Figure 5: Structure diagram before and after resource multiplexing

In the hardware implementation, for the Zero Padding layer, this system is implemented by the edge detection method, which reduces the clock cycles and the consumption of storage resources for this layer. For the activation layer, except for the last layer, which is a Soft max function, the ReLU activation function is used, which not only saves hardware resources, but also effectively removes negative points and is easy to implement in hardware. The formula of the ReLU activation function is as follows:

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

4.3: POOLING LAYER

A convolutional layer is usually followed by a pooling layer that reduces the spatial dimension of feature maps. A pooling layer works similarly to a convolution layer. The difference is that convolution kernels are replaced by pooling kernels. Max Pooling and Average Pooling are two commonly used pooling algorithms.

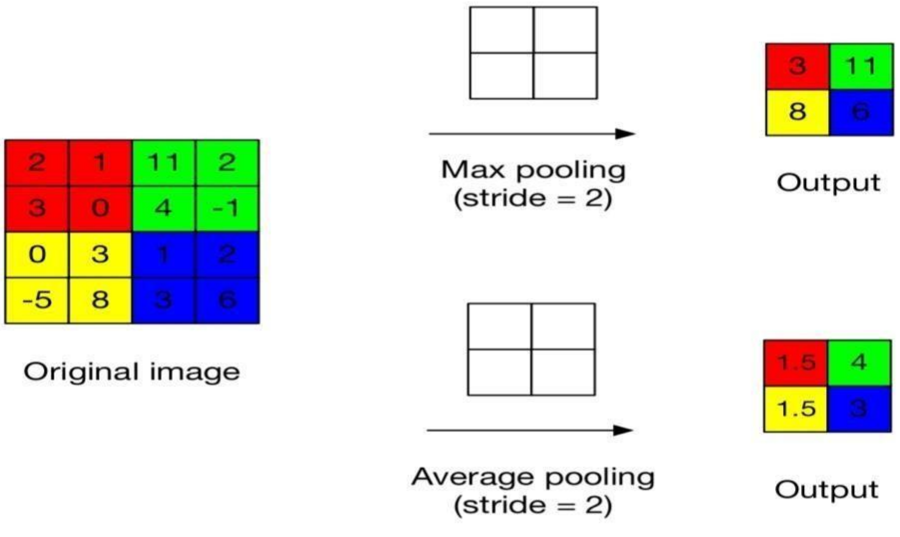

Figure 6: Max pooling and average pooling

4.4: FULLY CONNECTED LAYER

Pooling layers help to solve these problems. The output of a convolutional layer will be different even if only one pixel of the input changes. Let us assume that a CNN consists of only convolutional layers and fully connected layers. If there are many images that are only slightly different from each other in the training set, the network will have a very high chance of overfitting since the network has to take all pixels into consideration. With a pooling layer, the resolution of the image is reduced, which means that some trivial features are removed. Those images that have little difference from each other may end up generating the same feature map. Therefore, the network is able to tolerate small differences between images without suffering from overfitting.

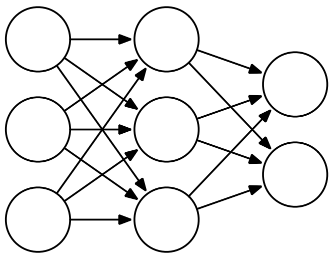

Figure 7 : Fully connected layer

5: TRANING THE NETWORK

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins. In supervised training, both the inputs and the outputs are provided.

The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.

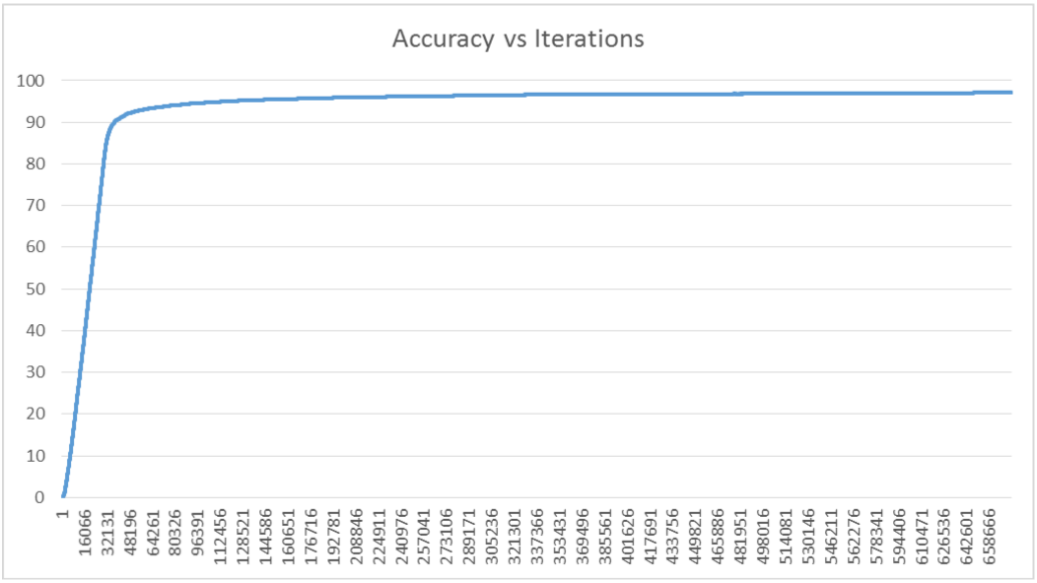

Figure 8: Training result of CNN for MNIST

Hyper Parameters used for training the network are:

- Learning rate: 0.001
- Mini Batch Size: 1
- L2 Regularization strength: 0.001
- Momentum: 0.9

Type of Network used	**Training**	**Validation**
Multi layer Perceptron	95.30%	70.10%
Convolutional Neural Network	98.15%	97.25%

Table 9: Training results of the network

6. RESULT

This paper proposes an FPGA-based resource multiplexing-architecture convolutional neural-network-processor design, which aims to reduce the consumption of hardware resources and power consumption by CNN. The system takes handwritten number recognition CNN as an example to design a convolutional neural network processor with a resource multiplexing architecture. Finally, the prediction accuracy of the processor is 97.25 percent.

7.CONCLUSION

An image detection hardware architecture based on convolutional neural network algorithm. The convolution operation is designed by a pipeline scheme. The system processes 8 convolution kernels in parallel to speed up the processing of the image detection system. Through simulation, each module and the entire system can work normally. The use of hardware parallel processing features and the use of pipelined processing data in the implementation of the FPGA increases system throughput and speeds up the system.

REFERENCE

[1] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. “Optimizing FPGA-based accelerator design for deep convolutional neural networks”. In: Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM. 2015, pp. 161– 170.

[2] Kalin Ovtcharov, Olatunji Ruwase, Joo-Young Kim, Jeremy Fowers, Karin Strauss, and Eric S Chung. “Accelerating deep convolutional neural networks using specialized hardware”. In: Microsoft Research Whitepaper 2.11 (2015), pp. 1–4.

[3] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou,
Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, et al. “Going deeper with embedded FPGA platform for convolutional neural network”. In:
Proceedings of the 2016 ACM/SIGDA International Symposium on FieldProgrammable Gate Arrays. ACM. 2016, pp. 26–35.

[4] Naveen Suda, Vikas Chandra, Ganesh Dasika, Abinash Mohanty, Yufei
Ma, Sarma Vrudhula, Jae-sun Seo, and Yu Cao. “Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks”. In: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM. 2016, pp. 16–25.

[5] Ritchie Zhao, Weinan Song, Wentao Zhang, Tianwei Xing, Jeng-Hau Lin, Mani Srivastava, Rajesh Gupta, and Zhiru Zhang. “Accelerating binarized convolutional neural networks with software-programmable FPGAs”. In: Proceedings of the 2017 ACM/SIGDA International Symposium on FieldProgrammable Gate Arrays. ACM. 2017, pp. 15–24.

[6] Yongmei Zhou and Jingfei Jiang, "An FPGA-based accelerator implementation for deep convolutional neural networks," 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), Harbin, 2015, pp. 829-832.

[7] C. Morales Morales, U. Flores, M. Adam Medina, M. Diaz Salazar, J. Abiel Caballero, D. Criado Cruz and S. Pavoni Oliver, "Digital Artificial Neural Network Implementation on a FPGA for data classification", IEEE Latin America Transactions, vol. 13, no. 10, pp. 3216-3220, 2015.