Handwriting recognition system based on FPGA

To cite this article: Yiying Huang et al 2019 J. Phys.: Conf. Ser. 1176 032026

View the article online for updates and enhancements.
Handwriting recognition system based on FPGA

Yiying Huang1, Jianmin Wang2,∗ and Shu Li2
1School of Science, University of Science and Technology, Harbin, China
2School of Science, University of Science and Technology, Harbin, China

*Corresponding author e-mail: wjmfuzzy@126.com

Abstract. An FPGA-based handwriting recognition system is proposed in this paper, which uses convolutional neural network algorithm to realize the recognition of MNIST digital sets. Hardware design techniques including image cropping, convolution, activation functions, pooling, pipeline processing, and parallel processing of multiple convolution kernels are described. Using the characteristics of parallel computing of hardware circuits, the processing speed of the MNIST digital set detection system is accelerated. The proposed MNIST detection architecture is designed using the Verilog HDL and implemented in the Altera DE2 FPGA development board.

1. Introduction
Image recognition in machine vision has always been an active area of research. With the advent of the 5G era, image recognition has a wide range of potential applications in the Internet of Things, human machine interfaces, smart home, and biomedical imaging analysis.

In the convolutional neural network structure implemented by software, the main bottleneck restricting the speed of the neural network is that the convolution process consumes a large amount of computing time and consumes a large amount of resources. The hardware structure parallel processing of data features can accelerate the calculation process to achieve real-time recognition of images. The implementation of convolutional neural networks through FPGAs can take advantage of the fast and easy-to-modify features of FPGAs for application-side requirements.

Neural networks are divided into training processes and recognition processes. In this project, the convolutional neural network training process is realized by matlab, and the recognition process is realized by FPGA. The system uses the MNIST database as a training set and test set.

2. Convolutional neural network
Convolutional neural networks are the basis of this system design. As a kind of feedforward neural network, the convolutional neural network realizes the extraction of picture features through convolution operations. The pooling operation performs secondary feature extraction on the image, thereby achieving the effect of simplifying the number of neural network parameters.

The image recognition area is a fixed position in the picture, a fixed-size rectangular frame (28*28), and the system only detects the information in the image recognition. The image recognition area can be changed to other sizes as needed.

The convolutional neural network uses a cost function to characterize the degree of fit of the network. The smaller the cost function, the better the fit of the neural network. Thus, the problem of finding a neural network structure with a good degree of fit is transformed into a parameter that makes
the cost function reach a minimum value. In order to make the cost function as small as possible, the stochastic gradient descent method and the error backpropagation algorithm are used to change the weight and bias of the neural network, so as to obtain a neural network with good fitness[2].

Convolutional neural networks contain input layers, hidden layers, and output layers. The hidden layer is further divided into a convolution layer, a rectification layer, a pooling layer, and a fully connected layer.

3. Implement

3.1. System Overview

The neural network in the design is used to identify the MNIST handwritten data set, which has 8 convolution kernels. The input data is the pixel value of a specific area in the 640*480 image, and the size is 28*28. The output is displayed on the 7-segment display on the DE2 development board. The structure of the neural network uses eight 7*7 size convolution kernels, one convolution layer, one pooling layer, and one full connectivity layer. The training process of the neural network is completed by matlab. The weights and offsets of the training neural network are then passed to the FPGA project by the scripting language.

The system architecture is shown in Figure 1. The system first decodes the image data collected by the camera and then stores it in the SDRAM. Convert pixel values to grayscale values. The gray value is then binarized. Each time the 7*7 rectangular frame size image data is extracted and convolved with 8 convolution kernels, each time one pixel is moved, the rectangular frame moves from left to right and from top to bottom Z shape. After convolution, a 22*22 convolutional layer is obtained, and then the convolutional layer is divided into a number of 2*2 rectangular regions. A maximum value is obtained for each 2*2 region and an 11*11 pooling layer (2*2 maximum pooling) is generated at the position of the convolution layer according to the rectangular region. Then, each data of the pooling layer (8*11*11) corresponding to the eight convolution kernels is input, multiplied by the weight and offset to obtain the fully connected layer.

Figure 1. System framework.
3.2. Handwriting recognition framework based on FPGA

3.2.1. Image data acquisition. The mixed video signal transmitted by the camera is decoded, and the video decoding chip 7181b is driven by the I2C protocol to convert the analog signal into a digital signal, and the image data is deinterleaved and transmitted to the next module.

![Figure 2. 28*28 buffer structure block diagram.](image1)

![Figure 3. Multiplication addition tree.](image2)

3.2.2. Image preprocessing. Since a specific area in the image is to be identified, the size of the recognition area is 28*28, so a 28*28 buffer is constructed and a corresponding data control module is constructed. The 28*28 buffer structure block diagram is shown in Figure 2. The control module accepts the coordinate values of the X and Y of the picture and the pixel values of the corresponding positions, and determines whether it is within the identification area, if the pixel is within the identification area, shifts one pixel value data to the left of the buffer, when the first line is filled At this time, the data in the buffer is moved up as a whole, and the action is repeated until the buffer is filled. The control module outputs a buffer_rdy signal indicating that the 28*28 buffer has been filled.

3.2.3. Convolution. The data in the 28*28 buffer is convoluted, and there are 8 convolution kernels with a size of 7*7. The convolution operation consists of two steps of multiplication and addition, so it is necessary to construct a corresponding multiplier and adder, which are combined to become a multiplier_adder trees.

In order to realize the convolution operation, a 7*7 convolution window buffer is constructed to store the gray value to be convolved in the local receptive field, the convolution kernel data and the gray value of the corresponding position of the 7*7 size picture. The product is multiplied to obtain 49 product results, and the 49 results are summed to complete a convolution. The local receptive field moves from left to right and moves from top to bottom Z-shaped pixels. For a 28*28 image, 22*22 convolution results can be obtained. For multiplication operations, at least 49 multipliers need to be constructed.

The multiplier_adder trees is constructed as shown in Figure 3, and the 49 product results are used as input to the adder. The addition tree uses a two-input adder and a cascade structure, the output of each stage is equal to the input of the next stage, and finally the sum of all product results is obtained. Therefore the input must be an integer power of two. Since there are only 49 convolution results as input, at least 2 6th power multipliers are required. Construct 64 multipliers, output the result as input to the addition tree, and the 15 additional multipliers take zero as input. The convolution result is output after 6 cycles.

![Figure 3. Multiplication addition tree.](image3)
The input of the design of the multiplier structure must be $2^n$ power, $7 \times 7 = 49$ is between 32 and 64, so it is necessary to construct a 64 input adder, which is more than the data of 49 inputs. 0 as input. The entire addition tree needs an array of 127 data to store.

To reduce the delay, we construct four identical multiplier_adder trees and four $7 \times 7$ convolution window buffers, corresponding to a $2 \times 2$ pooling layer as Figure 4 shown. The weight inputs of the four multiplier_adder trees correspond to the same convolution kernel. The gray data input end corresponds to the gray value in the local receptive field.

![Figure 4. Four local receptive fields in the recognition window.](image)

![Figure 5. Rectification and pooling](image)

The rectification process corresponds to a neural network activation function. The system uses the ReLU activation function, so you only need to convert the convolution result less than zero to zero, and the convolution result greater than zero or equal to zero is equal to itself. As shown in Figure 5, the positive and negative discrimination of the convolution result can be completed by judging the sign bit of the convolution result.

3.2.4. **Pooling.** The pooling operation takes a maximum value of four pixel values in the upper and lower rows of the convolutional layer, and a buffer of $2 \times 2$ size stores four pixel values. The pooling process involves four rows of data in the upper and lower rows, and their addresses are not continuous. Therefore, four multiplier_adder trees corresponding to four local receptive fields are constructed to convolute the gray values in the local receptive field. The four convolution operations have the same convolution kernel, and the input of the gray data corresponds to the gray value in the four local receptive fields. The relative positions of the four local receptive fields are as shown in FIG. 4. Each time the four local receptive fields move as a whole Z-shaped, when moving two pixels at a time when moving laterally, from the rightmost side to the leftmost side, a total of 11 pooling results are output. Then the four partial feelings return to the leftmost position as a whole, and then move down two pixels. Then move 2 pixels from left to right each time, and output 11 other pooling results, and so on. Finally, a pooled layer of $11 \times 11$ is obtained. For 8 convolution kernels, a pooled result of $8 \times 11 \times 11$ is finally obtained. The pooled data corresponding to the eight convolution kernels are stored in eight RAMs.

3.2.5. **Fully connected layer.** Each pooled result ($11 \times 11 \times 8$ total) is connected to each neuron of the fully connected layer. The pooled result is multiplied by the corresponding weight to derive the result of each output neuron.

3.2.6. **Show result.** The result of each output neuron (10 in total) is connected to the comparison circuit to obtain a maximum value, and the number represented by the corresponding neuron is displayed on the 7-segment digital tube.
3.3. FPGA implementation

The recognition system is built using Matlab to train the weights and offsets of the resulting neural network. These weight and deviation training data are trained by the MNIST database. The trained parameters are then passed to the FPGA project file. When building a handwritten digit recognition system through FPAG, the hardware circuit is used to process data in parallel, and the circuit structure is designed in a pipelined manner.

The inspection system is divided into four sections.

The first part is the acquisition of gray scale data in the recognition area. This part is composed of an image acquisition module, an image decoding module, an image storage module, an image window control module and a window encapsulation module. The video signal collected by the image acquisition module is processed by the image decoding module to obtain a grayscale image of 640*480 size and transmitted to the image storage module.

The second part is the multiplier-adder trees that is needed to construct the convolution. After obtaining the 28*28 size identification area, the data in the 28*28 buffer is convolved. The window encapsulation module contains four 7*7 convolution window buffers corresponding to the local receptive fields, which are used to store the gray value in the 7*7 rectangular frame to be convoluted, and the rectangular frame is called the convolution window. Since the recognition area size is 28*28 and the convolution window size is 7*7, the convolution window has 22*22 positions in the recognition area, corresponding to a 22*22 size convolution layer.

As shown in Figure 5, due to the use of the ReLU activation function, the linear rectifier module in the corresponding circuit. It is only necessary to convert the convolution result smaller than zero into zero, and the convolution result greater than zero or equal to zero is equal to itself. The corresponding circuit structure is the control bit of the data selector by the sign bit of the convolution result, and the convolution result and 0 are the data inputs respectively.

The third part is the construction of the pooling layer. As shown in Figure 5, due to the 2*2 maximum pooling, a 2*2 pooling buffer needs to be constructed, and each position in the buffer corresponds to the output of one multiplier-adder trees. The pooling is to take the maximum value of the adjacent 2*2 rectangular area data in the convolutional layer, and the data transmitted to the pooling module is transmitted from left to right, from top to bottom Z-type, and the distance of each movement is For two pixels. The pooled result corresponding to the convolution window is finally stored in the feature map storage module. Finally, a pooled layer of 11*11 size is output, and the result is stored in the RAM of the feature map storage module. There are 8*11*11 pooled data for 8 convolution kernels.

The fourth part is the construction of the fully connected layer. There are 10 outputs in the system, so 10 neurons are set in the fully connected layer. 8*11*11 pooled data stored in 8 feature map storage modules are connected to the neurons of each fully connected layer. For each fully connected layer neuron, the 8*11*11 pooled data is multiplied by the corresponding weight plus the offset and the result is summed to obtain the output value of the neuron. The maximum output is obtained by comparing the output of these 10 neurons. The number represented by the neuron that obtains the maximum value is the recognition result of the neural network.

Finally, the ten output results are compared to obtain a maximum value, and the number represented by the corresponding neuron is the recognition result of the system, and then the number is displayed through the 7-segment display digital tube.

4. Simulation results

The recognition area is set to an area of 28*28 in the upper left corner of the picture 640*480. The coordinates of the upper left corner of the picture are (0,0), and the coordinates of the lower right corner are (640,480) to establish the coordinate system, with screen_x representing the abscissa and screen_y representing the ordinate. The simulation results are shown in the Figure 6. Shift_left_tb remains high in the interval of screen_x from 1 to 28. Since the output of shift_left_tb has a clock delay. When screen_x reaches 29, shift_left_tb outputs a high level of the clock, indicating that the first line of data in the identification area of 28*28 has been stored in the 28*28 buffer, and the data in the buffer is moved up one position as a whole.
As Figure 7 shown, when screen_x is equal to 29 and screen_y is equal to 28, buffer_rdy_tb outputs a high level of a clock indicating that the 28*28 buffer has been filled. After 8 clock cycles, pixel_rdy_tb begins to output a high level until all the grayscale values in the 22*22 7*7 convolution window are convolved and rectified into the pooling module.

When 4 data in the 2*2 pooled buffer is valid data, nh_rdy_tb outputs a high level, and is input to the write enable end of the RAM, and the pooled data is stored in the feature map storage module.

The high level of fm_buffer_full_tb indicates that 121 pooling results are stored in the feature map storage module. Due to the parallel result, the pooling results corresponding to the eight convolution kernels are simultaneously stored in the eight corresponding feature map storage modules.

After the full connection layer matrix multiplication control module receives the high level of fm_buffer_full_tb, mult_en_tb starts to set to enable the fully connected layer matrix multiplication module until the pooling result corresponding to all convolution kernels and the full connection layer weight complete the product.

Pr_dy indicates that the matrix multiplication corresponding to all convolution kernels is completed, and the 7-segment digital display is enabled to display the recognition result.

5. Conclusion

This paper proposes 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.

References
[1] Michael A. Nielsen."Neural Networks and Deep Learning”.Determination Press, 2015
[2] Information on http://ufdl.stanford.edu/tutorial