FPGA Accelerator Design for License Plate Recognition Based on 1BIT Convolutional Neural Network

Yulu Cai\textsuperscript{1}, Xiunan Lin\textsuperscript{1}, Hui Qian\textsuperscript{1} and Peimin Lu\textsuperscript{2}

\textsuperscript{1} Key Laboratory of IOT Engineering and Applications, Fuzhou University, Fuzhou, China
\textsuperscript{2} Institute of Physics and Information Engineering, Fuzhou University, Fuzhou, China

Abstract. Aiming at the problem that convolutional neural network is difficult to deploy on small embedded devices due to its high complexity and large storage space requirement, this paper propose a convolutional neural network FPGA accelerator architecture based on binarization. Using the gray scale processing, binarization processing, threshold setting to reduce the number of parameters. Designing Parallel structures of convolution kernels, feature maps, and matrix blocks to accelerate. The designed architecture can be deployed on the AX7103 FPGA development platform with limited resources. The experimental results show that the convolutional neural network after parallel acceleration design can achieve a recognition accuracy rate of 98.73\% on the premise of reducing the data bit width from 32 bits to 8 bits, the recognition speed is about 0.21 seconds/time.

Keywords. License plate recognition; convolutional neural network; binarization; FPGA; accelerator design.

1. Introduction

In recent years, deep neural networks have some excellent research results in image recognition [1], computer vision [2], video surveillance [3], etc. Among them, convolutional neural networks (CNNs) have achieved excellent recognition results in pattern recognition, and especially in handwritten digit recognition [4]. CNNs have solved several problems which traditional license plate recognition algorithms can’t solve. CNNs can accurately identify fuzzy images which collected under bad weather or complex traffic conditions, and CNNs have high robustness and generalization ability [5]. However, CNNs demand a large number of datasets for training and a large amount of network parameters. Because of its intensive calculations and resource-intensive features, the current CNNs-based license plate recognition algorithms mainly rely on multiple blocks of GPU to be implemented [6]. However, GPUs have large power consumption, which causes CNNs calculate slowly after it deployed on small portable devices. It cannot meet the needs of practical applications. Field Programmable Gate Array (FPGA), as a reconfigurable hardware structure with a lot of array form logic, which has the characteristics of low power consumption and rich resources [7]. And its flexible structure [8] can matches the algorithms of each layer of CNNs to achieve high parallelization. It can effectively reduce the power consumption of the operation and improve the calculation efficiency. Therefore, using FPGA as a carrier for CNN implementation has become a hot issue in current research [9].

This paper proposes an accelerator design for the license plate recognition CNN based on AX7103 FPGA. It uses gray-scale processing, binarization processing, threshold setting and other pre-processing methods to cut down on the number of datasets. FPGA has the characteristics of transmit data in parallel and process information in real time. It used to realize parallel computing for each layer of deep
neural network. After experiments, the accelerator design proposed in this paper can realize the deployment of license plate recognition based on CNN in portable terminals with limited resources.

2. Convolutional Neural Network

2.1. The Basic Structure of the Convolutional Neural Network

Comparing with other deep neural networks, CNNs have the characteristics of weight sharing and local connection, which can reduce the calculation complexity of the entire network. The main structure of CNNs includes convolutional layer, pooling layer and activation function [10]. The input layer can directly input two-dimensional image data; the convolution layer is used to extract local feature information. It contains multiple feature planes, and each feature plane has multiple neurons. The input of the neuron is related to the previous layer, which is called the “receptive field”, and the convolutional layer extracts the information from the “local receptive field” [11]. The pooling layer is generally used for sampling to avoid dimensional disaster problems caused by multiple convolution operations. The activation function is used to transform the linear characteristics of the network into non-linearity.

2.1.1. Convolution Layer. The convolution operation is equivalent to using a convolution kernel to slide on the entire image sequentially in a certain step to extract the feature values of each part, which can be expressed by (1): \( N_{li} \) is the number of convolution kernel in the \((l-i)\)-th layer, \( k_{ij} \) is a trainable convolution kernel, \( i,j \) represent the two dimensions of the convolution, \( b_j \) represents the bias in the \( j\)-th feature map and in the \( l\) layer, \( k \) represents the weight, and \( f(\cdot) \) represents the activation function.

\[
x'_j = f\left(\sum_{i=1}^{N_{li}} x^{l-1}_{ij} \ast k_{ij} + b_j\right)
\]  \( (1) \)

The operation of the convolutional layer is shown in figure 1. We set the input image size to be \(5 \times 5\), and generate a \(3 \times 3\) feature map after the \(3 \times 3\) convolution kernel. The number in each small box in the figure indicates the size of the pixel value in the image, and the value of the calculated feature map is the result of the product of the input image and its corresponding convolution kernel pixel value. As shown in the feature map of figure 1, the pixel value of the first row and the first column is calculated as: \(1 \times 1 + 1 \times 1 + 0 \times 1 + 1 \times 0 + 0 \times 1 + 1 \times 0 + 1 \times 0 + 0 \times 1 + 0 \times 0 = 3\). Similarly, the output results of the other positions of the feature map can be calculated.

Convolution layer has two characteristics: local receptive and weight sharing. It solved the problems of difficult training and long training time caused by the excessive amount of weight parameters in the application of traditional neural networks used in the image field.

2.1.2. Pooling Layer. Normal pooling layers include maximum pooling layer and average pooling layer. The pooling layers often follows the convolutional layer, which is used for sampling. It could cut down on the training parameters extracted by the convolutional layers. And it could also avoid overfitting caused by too many parameters. Figure 2 shows a schematic diagram of the calculation of the maximum pooling layer, which can be expressed by equation (2):

\[
x'_j = \text{pool}(f\left(\sum_{i=1}^{N} x^{l-1}_{ij} \ast k_{ij} + b_j\right))
\]  \( (2) \)

2.1.3. The Activation Function. The activation function is often located behind the convolutional layer. It is used to introduce non-linear characteristic functions into the network structure. Commonly used activation functions include linear rectification functions (ReLU), hyperbolic tangent functions (Tanh) and Sigmoid functions. Experiments show that the ReLu activation function has a better convergence
rate. The ReLu can avoid the phenomenon of gradient explosion and disappearance of gradient. And alleviates the problem of overfitting. Therefore, the ReLu activation function is widely used in CNNs.

![Convolutional operation process](image)

**Figure 1.** Convolutional operation process.

![Pooling layer calculation diagram](image)

**Figure 2.** Pooling layer calculation diagram.

2.1.4. *The Fully Connected Layer.* The fully connected layer comprehensively maps all the learned features to the labeled space, and plays a role of feature weighting in the network. The fully connected layer can be regarded as a convolution layer with the size of the convolution kernel is $1 \times 1$, which is expressed as (3), where $w$ is the weight, $b$ is the bias, and $l$ is the number of layers.

$$x^l = f(w^{l-1}x^{l-1} + b^l)$$  \hspace{1cm} (3)

2.1.5. *CNN for License Plate Recognition.* In this paper, the CNN is a 5-layers network consisted of 2 convolutional layers, 2 pooling layers, and fully connected layer, as shown in figure 3. The activation function of the convolution layer is the ReLu. The size of the convolution kernel of the convolution layer is $3 \times 3$, the convolution channel is 8, the step size is 1. It is padding with 0 in the edges to maintain the output dimension of the convolution layer unchanged. The pooling layers are the maximum pooling layers, with the size is $2 \times 2$. The fully connected layer output is $10 \times 1$.

The input of the CNN used for license plate recognition is the digital gray-scale images of the printed body. The size of the images are $32 \times 32$, and the value range of each pixel is (0-1). The size of the convolution kernel of the first convolution layer is $3 \times 3$, the step size is 1. Causing that the boundary is padding with 0 elements to meet the input size, the size of the output feature map of the first layer is also $32 \times 32$. There are 8 convolution kernels which has $32 \times 32 \times 8 = 8192$ neurons, $(3 \times 3 + 1) \times 8 = 80$ weights and $(3 \times 3 + 1) \times 8 \times 1292 = 81920$ connection lines. The second layer is the maximum pooling layer with step size is 2. The output feature map size is $16 \times 16$. The third layer of the CNN is a convolution layer with the size of the convolution kernel is $3 \times 3$. Padding with 0 elements, so the output feature map is $16 \times 16$, the number of neurons are $16 \times 16 \times 8 = 2048$, the number of weights are $(3 \times 3 + 1) \times 8 = 80$, and the number of connections are $(3 \times 3 + 1) \times 20480$. The fourth layer is the maximum
pooling layer, and the image size after this layer becomes $8 \times 8$. The last layer is the fully connected layer, which is also an output layer. There are 10 neurons in this layer. Therefore, the fully connected layer has $(8 \times 8 \times 8 + 1) \times 10 = 5130$ weights. All of the parameters are shown in Table 1.

![Convolutional neural network topology](image)

**Figure 3.** Convolutional neural network topology.

**Table 1.** Parameters of license plate recognition convolutional nerves.

| Layer table | Feature map size | Parameter number |
|-------------|------------------|-----------------|
| Conv 1      | $32 \times 32 \times 8$ | $(3 \times 3 + 1) \times 8 = 80$ |
| Pool 1      | $16 \times 16 \times 8$ | - |
| Conv 2      | $16 \times 16 \times 8$ | $(3 \times 3 + 1) \times 16 = 160$ |
| Pool 2      | $8 \times 8 \times 8$ | - |
| Full connected | 10             | $(8 \times 8 \times 8 + 1) \times 10 = 5130$ |

The activation function used in the license plate recognition CNN is the ReLu activation function, and the expression is shown in equation (4). $m$ is the width of the input feature map; $n$ is the height of the input feature map.

$$ReLU = \begin{cases} 
0 & C_i(m,n) < 0 \\
\max(0, C_i(m,n)) & C_i(m,n) \geq 0 
\end{cases} \quad (4)$$

3. Assembly Convolutional Neural Network FPGA Hardware Acceleration Architecture

The CNN accelerator is composed of five fully pipelined processing units including a first-level convolutional layer computing unit, a first-level pooling unit, a second level computing unit, a second-level pooling unit and a fully connected layer computing unit. Because of the operations of each layer of the CNN is relatively independent, it can be calculated using parallel computing. Since 8 convolution kernels need a large number of multipliers and adders to calculate at the same time, in view of the fixed
convolution kernel parameters, a lookup table is used to perform the multiplication and addition of fixed coefficients. Finally, the results of the 8 input channels are accumulated using an addition tree. The design of the pooling layer needs to open a buffer of a certain size, using the shift register to shift the last bit of data from the buffer, while the new data will enter the buffer at the same time. The fully connected layer is the multiplication of the two matrices. The pipeline can be used to design the parallel calculation of weights and blocks, and then the comparator outputs the maximum of the 10 numbers in the fully connected layer. In the following subsections, we discuss the specific implementation schemes in detail.

3.1. Convolutional Layer Structure Design

\( \{K_i\} \) is the multiple convolution kernel sets in the convolution. \( C_j \) is the output of the \( j \)-th output in the first convolution layer. \( I_j \) is the \( j \)-th feature map, \( N \) is the number of convolution kernels in \( \{K_i\} \).

For the first convolution layer, \( i = 8, j = 5 \), the number of convolution layer parameter is fixed. For a \( 3 \times 3 \) pair of convolution kernels, parallel computing under the normal circumstances requires 9 multipliers and 8 adders to complete. The 8 convolution kernels are calculated simultaneously, which require 72 multipliers and 64 adders. However, in the case that the input is only 0 or 1, and the size of the convolution kernel is \( 3 \times 3 \), the 9-bit input data consisted of 0 and 1 actually completes a convolution operation, which are \( 2^{3 \times 3} = 512 \) kinds of data. The output results of 8 convolution kernels are respectively 92, 89, 104, 51, 94, 100, 59, 157. It’s faster to use a lookup table in this case where the input and output are both limited. Figure 4 shows the hardware accelerator for the first convolutional layer.

\[ \text{Figure 4. Hardware accelerator for the first convolutional layer.} \]

For the second convolution layer, the hardware accelerator of the convolution unit used in this paper is shown in figure 5. The convolution calculation unit is mainly composed of input buffer, output buffer, data buffer, weight buffer, multiplier and adder. \( K \) is the size of the convolution kernel, \( M \) is the size of the input feature map. The input data and the weight data enter the convolution calculation unit by line through the shift register. When the first input data reaches the end of the data buffer, the multiplier starts to work. In the CNN used in this paper, the input feature map of the second convolution layer is the output of the first convolution map with size is 16×16, and there are 8 feature maps in total, so \( M = 18, K = 3, i = 8, j = 8 \). The data buffer can store 39 8-bit numbers. The last bit of data will be shifted out from the high buffer and new data will be entered from the low buffer, which could form data flow. The weight buffer can store 9 8-bit numbers. When the 39 numbers are ready, the first convolution operation start to work. The multiplication operation of 9 numbers is completed in one cycle, and then it is sent to the pipeline addition tree for accumulation to obtain the final convolution result. Since the second convolution layer has 8 input feature maps, and there is no dependency between different convolution kernel, the 8 feature maps can be convolved at the same time, and the convolution calculation results are added to obtain an output result of the channel.
3.2. Activation Function Design
According to equation (4), the ReLu activation function can be known that only the most significant sign bit needs to be determined. When the highest bit is 0, the output remains unchanged; When the highest bit is 1, the output becomes 0.

3.3. The Pooling Layer Design
The pooling layer chooses the maximum pooling layer, and the step size is 2. The hardware accelerator for the pooling layer consists of input and output buffers, data buffer and comparators. Take the first pooling layer as an example, the input data is 32×32. Its data buffer can store 34 8-bit numbers. When the first input data reaches the end of the data buffer, the comparator starts to work. Because of the pooling step is 2, there will be a calculation result every cycle. The input data of the second layer is 16×16, its data buffer can store 18 16-bit numbers. The remaining operations are the same as those of the first pooling layer. Figure 6 shows the hardware accelerator for the pooling layer.

3.4. The Fully Connected Layer Design
For the fully connected layer, this paper uses the weight parameter matrix block to process. The hardware accelerator of the matrix multiplication part of the fully connected layer is shown in figure 7. The input and output buffers, data registers, and LUTBRAM used for store 8-bit weight parameters of the fully connected layer. In this article, the output of the pooling layer is 8×8×8, after tile it into one-dimensional, it becomes the input of the fully connected layer; the weight parameter matrix of the fully connected layer is 512×10, we divide it into 8 blocks, the size of each block is 64×10, and the corresponding input data is also divided into 8 blocks, each block size is 1×64. The block input of 1×64 is calculated in parallel with the 10 columns of the weight matrix block of 64×10, and each column is calculated in a pipelined manner. Finally, the calculation results of the eight blocks are accumulated through the pipeline addition tree. For the output results of the fully connected layer with size of 1×10, a pairwise comparison is performed through the comparator, and the index of the position where the maximum value corresponds to the characters in the license plate.

4. Experimental Results and Analysis

4.1. Experimental Environment
The CNN accelerator in this paper is deployed on the FPGA of XC7A100T-2FGG484I chip of Xilinx Artix-7 series. The entire structure of the FPGA using the core board and expansion board model. The core board are mainly composed of FPGA, DDR3 and QSPI-FLASH, which are responsible for FPGA high-speed data process and storage. The bandwidth of the entire system is up to 25Gb/s (800M * 32bit,
data bit width is 32-bit). The two DDR3 chips have a capacity of 8Gbit, which meet the demand for high buffers during the data processing. The FPGA selected by AX7103 is XC7A100T chip of XILINX’s ARTIX-7 series. The development board has the characteristics of small size, little resource requirement and easy deployment in the portable terminals.

4.2. Overall Architecture Optimization Plan

The small amount of resources in FPGA can’t content the deployment conditions of CNNs. In order to realize the arrangement of the CNN license plate recognition algorithm on FPGA, the following optimizations have been made in this paper:

4.2.1. Overall System Flow. The overall system can process the license plate images collected by the camera OV5640. This system does something like RGB to grayscale, binarization, morphological operations, license plate positioning, and character segmentation. And then the characters are scaled to 32×32 and sent to the convolution. And the recognition result is displayed on the LCD. The video data collected by the camera is processed by the self-mounted RGB image to grayscale image and processed by the binary coprocessor. On the one hand, ‘the video in to the AXI4-Stream’ is used to convert the line and field synchronization to a stream protocol, send to DDR and used for subsequent license plate segmentation. On the other hand, after the coprocessor processing and morphology processing, the position of the license plate is transmitted to CM3, the character is divided according to the proportion, and the divided character image is transmitted to the scaling coprocessor. The scaling coprocessor scales it to the size of 32×32 as input, and after it is processed by the CNN hardware accelerator, the result is displayed on the LCD. The system hardware block diagram is shown in figure 8.

Before the data is read into the CNN, the grayscale processing made the data volume from 24bit (three channels of R, G, and B) to 8bit (single channel), and the binarization processing changed the data volume from 8bit to 1bit. Setting the threshold and clipping process could made the data volume from 8bit to 1bit before entering the network, compressed by 3 times, 8 times and 8 times, respectively. These works provide the condition for the deployment of the CNN on the FPGA.

4.2.2. Image Binarization. The license plate images are color images composed of three channels of R, G, and B. The amount of these data is relatively large. Through grayscale processing, the range of pixel values can be changed from 256x3 to 256x1. It could increase the contrast and resolution of the images and reduce the amount of calculation. Equation (5) is used to calculate the image gray value H:

\[ H = 0.229R + 0.589G + 0.114B \]  

(5)
In actual design, we need to do something to avoid floating point operations. Therefore, we enlarged the entire formula 28 times, and then do the shift calculation. The calculated result is shifted to the right by 8 bits, and we only need to take the highest 8 digits. So the above formula could be rewritten as:

\[ H = (R \times 76 + G \times 150 + B \times 30) \gg 8 \]  \hspace{1cm} (6)

The binarization process could adjust the threshold value by the local threshold method to separate the license plate character information from the background information [10]. As shown in (7), setting the binarization threshold to 50, the value of pixels which below 50 set to 255, and the value of pixels which above 50 set to 0.

\[
P_x = \begin{cases} 
255 & (P_x > 50) \\
0 & (P_x \leq 50)
\end{cases} \hspace{1cm} (7)
\]

4.2.3. Character Scaling. Before the license plate recognition, according to the characteristics that the gap between every character on the license plate are the same, and each character accounts for 13% of the length of the license plate [11]. First determine the license plate frame coordinates, and then use the proportion to divide the characters. The size of the segmented character image is 128×64, and it needs to be scaled to 32×32 in order to match the input size of the CNN. The character scaling process reduces the amount of data by 8 times. The specific scaling process is shown in equation (8), where \(x_i, y_i\) represents the coordinates of any pixel on the compressed image.

\[
\text{resize}[x_i, y_i] = 0.25 \times \text{unresize}[2x_i + 1, (4y_i + 1)] + 0.25 \times \text{unresize}[2x_i + 1, (4y_i + 1)] + 0.25 \times \text{unresize}[2x_i + 1, (4y_i + 1)] + 0.25 \times \text{unresize}[2x_i + 1, (4y_i + 1)] \hspace{1cm} (8)
\]

4.3. Comparison with Other Algorithms

Existing license plate recognition algorithms mainly include pattern matching methods, reinforcement learning algorithms, machine license-based license plate recognition algorithms, and neural network recognition algorithms. Through the comparative analysis of experimental results, the improved recognition algorithm of convolutional neural network realized in this paper has better recognition results in recognition accuracy.
4.4. Result and Analysis
In this paper, the proposed CNN accelerator is deployed on the Artix-7 development board. The Artix-7 series uses a miniaturized package and a unified Virtex series architecture, which provides the highest performance power consumption structure. It is suitable for the data acquisition, transmission and processing system of this project. After experiments, the method in this paper can implement the license plate recognition deployment on FPGA. The recognition accuracy is 98.73%, and the character recognition time is 0.21 s. The hardware deployment is shown in figure 9.

![Figure 9. LCD display results.](image)

The resource occupancy of each module of the identification system on FPGA is shown in table 2. LUT is lookup table, DSP is digital signaling processor, FF is trigger, and BRAM is embedded random memory processor. The resource utilization of each module is shown in table 3.

| Arithmetic Pattern matching [12] Reinforcement learning [13] SVM [14] CNN [15] This algorithm |
|---------------------------------|------------------|------------------|------------------|------------------|
| Accuracy rate (%)              | 91.7             | 98.8             | 95               | 97.5             | 98.7             |

5. Conclusion
This paper uses FPGA’s large-scale logic unit and a large number of digital processing units to design a FPGA-based CNN parallel accelerator. The overall calculation volume and resource occupation rate are reduced. We solved the problem of neural networks is hard to deploy on hardware. And the recognition speed of each character is 0.21 s, and the recognition accuracy of the PC is 98.73%. CNN has high requirements on device processing capabilities and severe restrictions on power consumption. And it cannot be applied to intelligent terminal devices in reality. This design in this paper solves these...
problems. At the same time, this CNN accelerate strategy is also applicable to the deployment of other CNNs on terminal, which has a wide scope of application and application prospects.

References
[1] DiCecco R, Lacey G, Vasiljevic J, et al. 2016 Caffeinated FPGAs: FPGA framework for convolutional neural networks Proceeding of 2016 International Conference on Field-Programmable Technology (FPT) pp 265-268.
[2] Poznanski A and Wolf L 2016 CNN-N-gram for handwriting word recognition Proceedings of the IEEE Conference on Computer Vision and Pattern recognition pp 2305-2314.
[3] Wang J, Chen Y, Hao S, et al. 2019 Deep learning for sensor-based activity recognition: A survey Pattern Recognition Letters 119 3-11.
[4] Sharif Razavian A, Azizpour H, Sullivan J, et al. 2014 CNN features off-the-shelf: An astounding baseline for recognition Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops pp 806-813.
[5] Li H, Fan X, Jiao L, et al. 2016 A high performance FPGA-based accelerator for large-scale convolutional neural networks 26th IEEE International Conference on Field Programmable Logic and Applications (FPL) pp 1-9.
[6] Zhang C, Li P, Sun G, et al. 2015 Optimizing FPGA-based accelerator design for deep convolutional neural networks Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (ACM) pp 161-170.
[7] Zhang C, Wu D, Sun J, et al. 2016 Energy-efficient CNN implementation on a deeply pipelined FPGA cluster Proceedings of the 2016 International Symposium on Low Power Electronics and Design (ACM) pp 326-331.
[8] Abdelouahab K, Pelcat M, Serot J, et al. 2018 Accelerating CNN inference on FPGAs: A survey (arXiv preprint) arXiv: 1806.01683.
[9] Draper B A, Beveridge J R, Bohm A P W, et al. 2003 Accelerated image processing on FPGAs IEEE Transactions on Image Processing 12 (12) 1543-1551
[10] Jiheng F E I 2019 Character Information Recognition Method Based on Image Processing (U.S. Patent Application 10/311) p 322.
[11] Pan G 2019 Research and Application of Key Technology of License Plate Recognition Based on Convolutional Neural Network (Beijing: Beijing University of Posts and Telecommunications).
[12] Li C 2018 Design of license plate recognition system based on deep learning Computer Technology and Development (06) 85-89.
[13] Crow B and Burry A M 2016 A deep reinforcement learning approach to character segmentation of license plate images IEEE International Conference on Machine Vision Applications pp 539-542.
[14] Li L, Zu G, Gu P, et al. 2018 Research on license plate recognition technology based on SVM Journal of Guizhou University (Natural Science Edition) (05) 48-54.
[15] Zhao H, Liu J, Jiang L, et al. 2019 Two-line license plate segmentation algorithm based on convolutional neural network Journal of Computer Aided Design and Graphics 31 (8) 1320-1329.