The 9th IEEE International Symposium on Next-Generation Electronics

Traffic Sign Recognition Based on ZYNQ

Juan Liu
School of Physics and Electronic Science
Changsha University of Science and Technology
Hunan Provincial Key Laboratory of Flexible Electronic Materials Genome Engineering
Changsha, China
2420526551@qq.com

Zhigang Zhang
School of Physics and Electronic Science
Changsha University of Science and Technology
Hunan Provincial Key Laboratory of Flexible Electronic Materials Genome Engineering
Changsha, China
zzgxbox360@126.com

Longhao Zheng
School of Physics and Electronic Science
Changsha University of Science and Technology
Hunan Provincial Key Laboratory of Flexible Electronic Materials Genome Engineering
Changsha, China
longhaozheng@qq.com

Yongjun Wen
School of Physics and Electronic Science
Changsha University of Science and Technology
Hunan Provincial Key Laboratory of Flexible Electronic Materials Genome Engineering
Changsha, China
micowen@csust.edu.cn

Feng Bin
School of Physics and Electronic Science
Changsha University of Science and Technology
Hunan Provincial Key Laboratory of Flexible Electronic Materials Genome Engineering
Changsha, China
173704725@qq.com

Lijun Tang
School of Physics and Electronic Science
Changsha University of Science and Technology
Hunan Provincial Key Laboratory of Flexible Electronic Materials Genome Engineering
Changsha, China
tanglj@csust.edu.cn

2021 9th International Symposium on Next Generation Electronics (ISNE) | 978-1-7281-6258-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISNE48910.2021.9493630

***Abstract*—In the traffic field, the traffic sign detection and recognition system is an important part of ADAS. In this paper, a research on traffic sign recognition is carried out, and a traffic recognition sign recognition network is established with the GTSRB data set as the research object. A network computing accelerator is designed with "ARM+FPGA" framework; the functional modules of each part of the network are reasonably divided; and high-level synthesis tools are used to optimize the design of loop unwinding and loop flow to improve throughput and improve recognition speed. The experimental results show that the network has a high accuracy rate in each classification. After optimization, the processing speed of the system under the 100MHz clock is increased by 5.59 times. This method provides an effective reference for the identification of other specific signs.**

Keywords—CNN, Traffic sign recognition, ZYNQ

I. INTRODUCTION

Advanced Driver Assistance System(ADAS) can assist the driver to reduce the safety hazards caused by human cognitive deviation. The traffic sign detection and recognition system is an important part of ADAS. The accuracy and recognition speed of traffic sign classification and recognition is one of the important standards to measure ADAS.

Traditional traffic sign recognition methods include detection algorithms based on template matching and threshold segmentation[1] edge detection algorithms, color space-based detection algorithms, and color coding-based detection algorithms[2]. With the rapid development of computer hardware, various feature extraction algorithms and deep neural networks have been widely used, such as Kd tree classification[3], convolutional neural network[4], region-based convolutional neural network (Faster R-CNN)[5], Capsule Convolutional Network[6], GB-DBM[7], SegU-Net[8], which have greatly improved the recognition rate of traffic signs. Although deep learning algorithms have high recognition rate and flexibility, they have high requirements for computing power, demanding hardware requirements, and difficult applications.

To this end, this paper takes traffic sign recognition as the research object, builds a traffic sign neural network model, and uses HLS tools for hardware acceleration to provide an effective reference for realizing a high-accuracy and fast-speed traffic sign recognition system.

II. NETWORK BUILDING

A. Data Set Production

Ten types of traffic signs are randomly selected from the German Traffic Sign Recognition Benchmark(GTSRB) data set: Speed limit 20km/h, Speed limit 60km/h, Speed limit 120km/h, No passing, Stop, Warning, turn right ahead, turn left ahead, road narrows on the right, slippery road. In order to effectively identify the label, the data set needs to be pre-processed such as data enhancement and label generation.

In order to improve the effect of network classification, the Fancy PCA[9] method is used to expand the data set. First, perform principal component is analyzed on the R, G, and B of the training data set to obtain the corresponding eigenvectors and eigenvalues, and a set of random values are calculated according to the eigenvectors and eigenvalues, as shown in equation (1).

$$[p_1, p_2, p_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3] \tag{1}$$

It is added as a disturbance to the original pixel value, which obeys a Gaussian distribution with 0 as the mean and 0.1 as the standard deviation. After each round of training, they will be randomly selected again and re-added to the original pixel as a disturbance.

In order to adapt to the caffe framework, it is necessary to generate an lmdb data format suitable for single-label data for training images. First, python is used to convert the pictures obtained from open source into txt files, and then shell is used to generate lmdb type data and average files.

978-1-7281-6258-4/21/$31.00 ©2021 IEEE

B. Network Architecture

A traffic sign recognition network is built. The network includes an input layer (INPUT), two convolutional layers (C1, C3), two pooling layers (S2, S4), a fully connected layer (F6), and an output layer (OUTPUT). The input layer inputs preprocessed 3-channel 32×32 pixel pictures. The convolutional layer C1 uses a 5×5 convolution kernel to perform feature extraction on the feature image, and obtains a 16-channel 28×28 pixel feature. The pooling layer S2 uses 2×2 maximum pooling to perform down-sampling processing on the feature image to obtain 16-channel 14×14 pixel features. C3 also uses a 5×5 convolution kernel to perform feature extraction on the feature image to obtain a 32-channel 10×10 pixel feature. S4 also performs 2×2 maximum pooling to down-sampling the feature image to obtain a 32-channel 5×5 pixel feature. The fully connected layer F5 has 84 neurons. Rule is used as the activation function, softmax is selected as the classifier, and the output layer outputs 10 judgment results. The network model structure is shown in Figure 1, and the specific parameters are shown in Table 1.

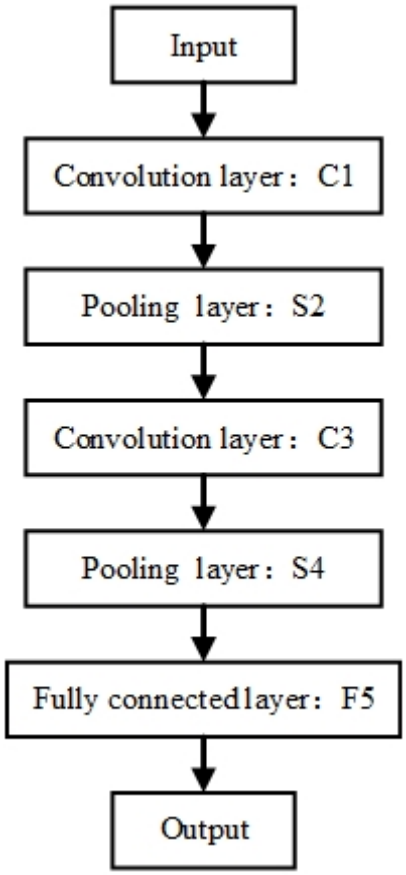

Fig. 1. Schematic diagram of model structure.

TABLE I. PARAMETERS CONFIGURATION

Layer	Feature map_size	Feature map_num	Kernel_size	stride
Input	32×32	3		1
C1	28×28	16	5×5	2
S2	14×14	16	2×2	1
C3	10×10	32	5×5	2
S4	5×5	32	2×2	1
F5		84 neurons		
Output		10 neurons		

III. ACCELERATION METHOD

A. Hardware Architecture

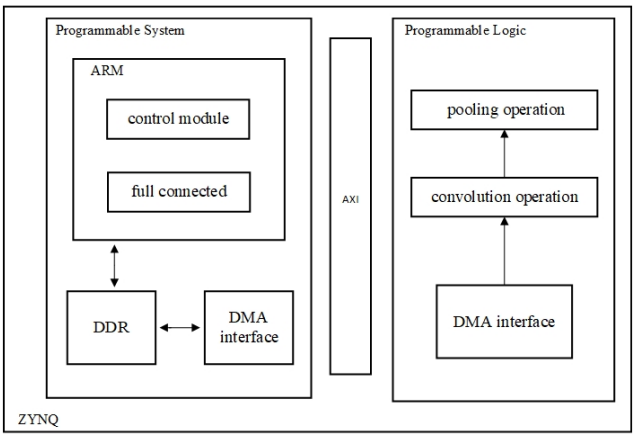

Fig. 2. System structure.

"ARM+FPGA" hardware framework is used, as shown in Figure 2. FPGA completes logic operations, ARM realizes logic control and performs complex floating-point operations, and the communication between FPGA and ARM is stable and fast through the AXI protocol. The combination of software and hardware greatly improves the flexibility of system design.

ARM is responsible for the overall process control of the convolutional neural network algorithm. First, the input features are stored in the DDR. After the storage is completed, the instruction is sent to the FPGA to start the convolution operation, and the convolution operation result is transmitted to the pooling operation. After the pooling operation, the FPGA will transmit the results to the ARM end for fully connected operation, and the final classification result is output from the ARM end. The control information and data transmission between ARM and FPGA is realized through DMA interface using AXI protocol.

B. Hardware Implementation

Through the Vivado HLS tool, the C language is used to design the system, and the design results are optimized by adding HLS constraints to increase throughput and increase system operating speed.

The convolution operation is mainly realized by the nesting of six levels of for loops, and the loop sequence is Hout, Wout, CHout, Ky, Kx, Chin. The pooling operation is mainly completed by a five-layer for loop, and the loop sequence is CHin, Hin, Win, Kx, Ky. By default, HLS synthesizes the loop into a sequential execution mode, which wastes the parallel computing function of the FPGA. Loop unrolling can copy the loop body through the loop factor, thereby increasing the parallelism of the loop body and making the system faster. The loop instruction #pragma HLS UNROLL is inserted into the loop body and it will be automatically expanded by the HLS tool. In the same way, the loop body is executed sequentially by default, and all statements in the second iteration of the same loop body will not occur until all the statements in the first iteration are completed, and the same is true for subsequent iterations. But statements that are not mutually exclusive can be executed in parallel to increase system performance. Pipeline processing is used for the innermost loop. When pipelinling constraints

are added, the sub-loops under the loop are expanded to improve throughput.

IV. Results and Discussions

A. Network Training Results

The network training process in this article uses the Windows 10 operating system, the hardware environment is dual-core Intel(R) Xeon(R) Gold 5218R CPU @2.10GHz, and the graphics card is TITAN RTX from NVIDA. After 10,000 trainings, the test results of the network on each category are shown in Table 2.

TABLE II. Training Result Test

Category	Accuracy
Speed limit 20km/h	100%
Speed limit 60km/h	99.33%
Speed limit 120km/h	99.90%
No passing	98.91%
Stop	94.46%
Warning	96.73%
turn right ahead	99.78%
turn left ahead	99.68%
road narrows on the right	98.65%
slippery road	96.64%

B. Hardware Test Results

Under 100M clock, use Vivado to synthesize the number of clock cycles consumed by the system before optimization and the number of clock cycles consumed by the system after optimization by the HLS tool. As shown in Table 3, the optimized result is 5.59 times higher than that before optimization. Under a clock of 100M, the system takes 56.88ms to recognize a picture once.

TABLE III. HLS Optimization Results

	Lantancy
Before optimization	4,688,651
After optimization	838,307

V. Conclusions

For traffic signs, a traffic sign recognition network is constructed, which has a high accuracy rate in each classification. Reasonable function division is carried out on the ZYNQ 7020 platform, and optimization systems such as loop expansion and assembly line structure are adopted to improve the speed of traffic sign recognition. Under 100MHz clock, after HLS optimization, the system FPGA part consumes clock increased by 5.59 times. If the FPGA processor with richer resources is used, the accuracy and processing speed of the system can be further improved.

Acknowledgment

This work is supported by the Hunan Provincial Key Research and Development Project [Grant No. 2018GK2054],the Hunan Water Conservancy Science and Technology Project [Grant No. XSKJ2019081-44],the Open Research Fund of Hunan Provincial Key Laboratory of Flexible Electronic Materials Genome Engineering [Grant No. 202019], the Postgraduate Scientific Research Innovation Project of Hunan Province [Grant No. CX20200896],and the Changsha University of Science and Technology Graduate School-level Innovation Project [Grant No. CX2019SS].

References

[1] H. Akatsuka, S. Imai, “Road signposts recognition system,” Proc. SAE vehicle highway infrastructure: safety compatibility, 189–196, 1987.

[2] L. Priese, J. Klieber, R. Lakmann, et al. “New results on traffic sign recognition,” Intelligent Vehicles'94 Symposium, Proceedings of the. IEEE, 1994:249-254.

[3] F. Zaklouta, B. Stanciulescu, O. Hamdoun, “Traffic sign classification using K-d trees and Random Forests,” International Joint Conference on Neural Networks. IEEE, 2011:2151-2155.

[4] D. Tabernik, D. Skocaj, “Deep Learning for Large-Scale Traffic-Sign Detection and Recognition,” IEEE Transactions on Intelligent Transportation Systems, 2020, 21(4): 1427-1440.

[5] SANTOS D C, SILVA F A D, PEREIRA D R, et al. Real-Time Traffic Sign Detection and Recognition using CNN[J]. Revista IEEE América Latina, 2020, 18(3): 522-529.

[6] H. Y. Guan, Y. T. Yu, D. F. Peng, et al. “A Convolutional Capsule Network for Traffic-Sign Recognition Using Mobile LiDAR Data With Digital Images,” IEEE Geoscience and Remote Sensing Letters, 2020, 17(6): 1067-1071.

[7] H. Y. Guan, W. Q. Yan, Y. T. Yu, et al. “Robust Traffic-Sign Detection and Classification Using Mobile LiDAR Data With Digital Images,” IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2018,11(5): 1715-1724.

[8] U. Kamal, T. I. Tonmoy, S. Das, et al. “Automatic Traffic Sign Detection and Recognition Using SegU-Net and a Modified Tversky Loss Function With L1-Constraint,” IEEE Transactions on Intelligent Transportation Systems, 2020,21(4): 1467-1479.

[9] A. Krizhevsky, I. Sutskever, G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” New York: ACM, 2017: 60, 84-90.