FPGA BASED IMPLEMENTATION OF CONVOLUTIONAL NEURAL NETWORK FOR HYPERSPECTRAL CLASSIFICATION

Xiaofeng Chen[1], *Jingyu Ji*[1], *Shaohui Mei*[1,*], *Yifan Zhang*[1], *Manli Han*[2], *Qian Du*[3]

[1] School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China.
[2] Aeronautical Computing Technique Research Institute, Xian 710068, China.
[3] Department of Electrical and Computer Engineering, Mississippi State University, MS 39762, USA.

ABSTRACT

convolutional neural network (CNN) has been widely used for hyperspectral classification. Current researches of CNN based hyperspectral image classification is mainly implemented on graphics processing unit (GPU) platform. However, GPU is not suitable for onboard processing due to the problem of space radiation and power supply on image acquiring platform. Therefore, in this paper, FPGA is selected to implement CNN based hyperspectral classification for further onboard processing. Specially, a hardware model is designed for the forward classification step of CNN using hardware description language, including computation structure for CNN, implementation of different layers, weight loading scheme, and data interfere. Simulation results over Pavia data set validate the proposed FPGA based implementation is coincide with that on GPU platform.

Index Terms— Convolutional neural network, hyperspectral, classification, FPGA

1. INTRODUCTION

Hyperspectral imaging technologies usually gather reflectance or radiance information of ground material over tens of hundreds of contiguous bands. According to such abundance spectral information, pixels in a hyperspectral image can be classified into a number of pre-defined categories of ground materials. As a result, it has been widely utilized to address a variety of earth remote sensing problems, such as environmental mapping, global change research, assessment of trafficability, plant and mineral vegetation, and etc. [1].

Recently, deep learning based classification methods have obtained significant achievements in hyperspectral applications due to their superior performance in feature learning. For example, Hu et al. [2] firstly applied convolutional neural network (CNN) to classify hyperspectral images directly in spectral domain. Mei et al. [3] integrated spatial information into CNN using mean and standard variation of pixels in a spatial neighborhood as input. Lee et al. [4] proposed a new deeper and wider network, which can optimally explore local contextual interactions by jointly exploiting local spatio-spectral relationships of neighboring individual pixel vectors. Zhao et al. [5] proposed a spectral-spatial feature based classification (SSFC) framework that jointly uses dimension reduction and deep learning techniques for spectral and spatial feature extraction. Chen et al. [6] proposed approach employs several convolutional and pooling layers to extract deep features from HSIs, which are nonlinear, discriminant, and invariant. In general, deep learning has been widely utilized in the field of hyperspectral image classification.

When using CNN for hyperspectral image classification, both the training of the network and the deployment of the model require a large amount of computation. Current researches of deep learning based classification of hyperspectral images is usually conducted on graphics processing unit (GPU) platform. However, GPU is not suitable for onboard processing due to the problem of space radiation and power supply on image acquiring platform. Therefore, it is necessary to implement current deep neural network on CPU and FPGA platform for onboard processing. Compared with FPGA, CPU requires need more calculate time, higher price and much power consumption. Obviously, CPU isnt a good choose to deploy CNN model. Generally FPGA is the most suitable platform for deploying CNN network model.

In this paper, FPGA is selected for implementation platform of CNN based hyperspectral classification because of its flexible architecture, fast computing speed, low power consumption, high stability and relatively low price. A hardware model for CNN [7] is designed by hardware description language, including computation structure for CNN, implementation of different layers, weight loading scheme, and data interfere. Finally, the proposed FPGA based implementation is evaluated by classification experiment over Pavia University dataset.

* This work is partially supported by National Natural Science Foundation of China (61671383), the Fundamental Research Funds for the Central Universities (3102018AX001), and the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University(ZZ2018021,ZZ2018036).

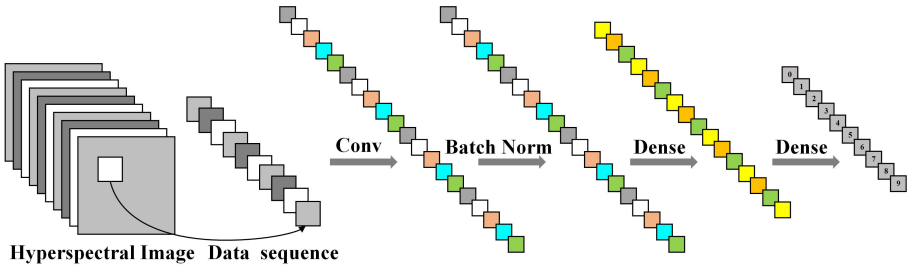

Fig. 1. The framework of the CNN to be implemented on FPGA platform for classification of HSIs.

2. PROPOSED METHOD

Table 1. the parameters for the CNN.

name	Configure
Convolution Layer	Kernel size = 11
	Feature map = 20
Batch Normalization Layer	\
PRELU1	\
Dense layer1	neurons=100
PRELU2	\
Dense layer2	neurons=9

CNN is capable to learn more complex features in hyperspectral images, which can significantly improve the classification accuracy. Since FPGA has been widely utilized as processor for airborne or spaceborne platform that the hyperspectral images are acquired, the CNN that has been demonstrated as an effective tool for both feature learning and classification of hyperspectral images is implemented on FPGA platform. As shown in Fig. 1, the CNN to be implemented contains one convolutional layer, one batch normalization layer and two dense layer. The parameters of all the layers are listed in Table. 1. It should be noted that only the forward calculation of CNN is implemented since the training step can be conducted off-line by GPU platform to update parameters used in the proposed FPGA based implementation.

2.1. The design of computation structure

The computation structure of CNN on FPGA platform is designed as shown in Fig. 2. The pipelined structure is used to maximize the computational efficiency and the utilization of logic resource on FPGA. According to the structure of CNN shown in Fig. 1, the computation structure is divided into four sequentially connected modules. As shown in Fig. 2, when 'module 1 pass the results to 'module 2, it would be in state of idle. Therefore, after output results to 'module 2, 'module 1can receive new input data for processing. Such sequential-computing and parallel-work structure make full use of the system's computing resources.

Generally, thousands of parameters are involved in a CNN to explore the characteristic of big data. The model of CNN [7] that we employ has more than 187240 parameters. All of these parameters can be optimized jointly with subsequent classifiers to take full advantage of both feature extraction and classification. Since these parameters are called repeatedly during in the calculation, they are stored on the on-chip ROM. By writing the parameters to the MIF file (Memory Initialization File), program will automatically store the corresponding data to the corresponding on-chip ROM in the simulation. Modules will pass the calculated results to each other, so FIFO (first in first out) is used to buffer the intermediate data between modules.

2.2. Implementation of different layers

2.2.1. Convolution layer

Convolution used to explore local spectral-spatial feature in CNN based hyperspectral classification by using the weighted summation of pixel value in neighboring spectral bands and spatial context. It mainly includes two kind of operation: multiplication and addition. The convolution of vectors $A(k) = [a_1, a_2, a_3, ..., a_k]$ and $B(k) = [b_1, b_2, b_3, ..., b_k]$ is defined as follows:

$$A \otimes B = \sum_{i=1}^{n} a_i b_i + \epsilon, \tag{1}$$

where '$\otimes$' represents convolution and ϵ is the bias of convolution kernel in CNN. High-dimensional convolution in CNN can also be interpreted using equation (1).

The calculation structure of convolution layer in FPGA platform is designed as shown in Fig. 3. This example containing 11 floating-point multipliers and 12 adder in the structure, in which the input A of multipliers corresponds to the 11 different parameters of the convolution kernel, the input B corresponds to the 11 data from a pixel of hyperspectral image. The input of adder is the output of multipliers, and the last adder output the result to FIFO for future calculation.

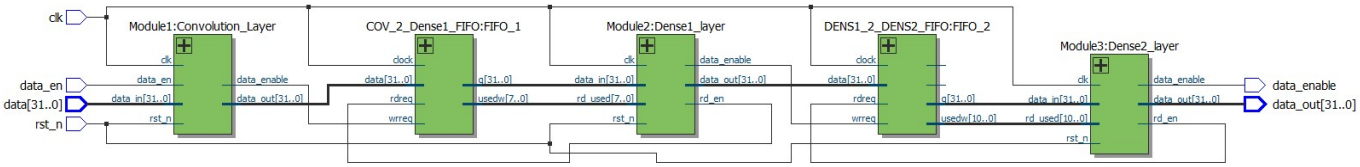

Fig. 2. The computation structure of CNN on FPGA platform.

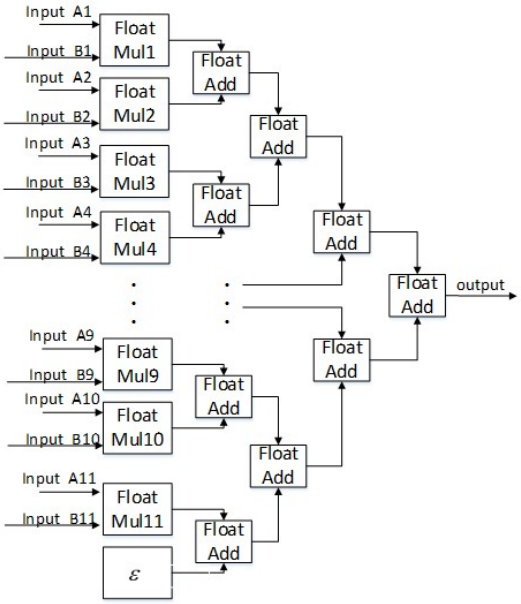

Fig. 3. The calculation structure of convolutional layer.

2.2.2. *Batch Normalization layer*

Batch normalization layer is used to improve the gradient of the flow through the network, and thus allowing a greater learning rate to improve the learning efficiency. The formula of batch normalization is as follows

$$\hat{x} = \frac{x - E(x)}{Var(x)} \times w + \epsilon, \tag{2}$$

in which x represents the input from previous convolution operation, $\hat{x}$ is the output, $E(x)$, $Var(x)$, and ϵ are fixed parameters. The calculation structure of batch normalization layer designed as shown in Fig. 4. It is observed that float-point adder, subtractor, multiplier, divider are involved in this layer.

2.2.3. *Dense layer*

The Dense layer is a classifier in the whole convolution neural network, and the calculation method is the same as the convolution layer. In our implementation, it share same calculation structure with convolution layer.

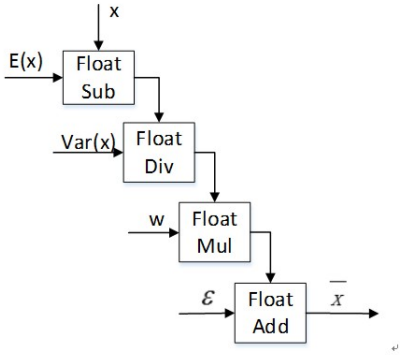

Fig. 4. The calculation structure of batch normalization layer.

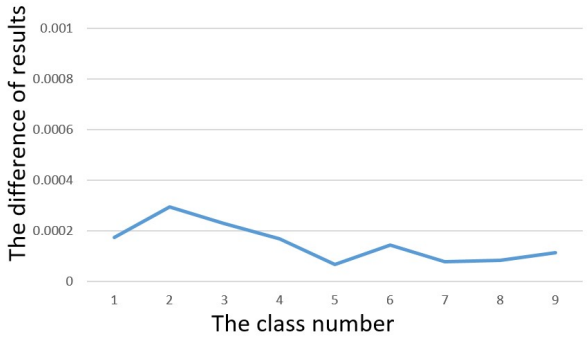

Fig. 5. The difference of the results between FPGA and GPU based implementation of CNN for hyperspectral classification.

3. EXPERIMENTS

3.1. Datasets

In the experiment, Pavia University dataset, which is acquired by the ROSIS sensor during a flight campaign over Pavia, nothern Italy, is adopted for evaluation. This dataset contains 610×340 pixels with a ground resolution of 1.3 meters. The number of spectral bands is 103. 9 classes are adopted for quantitative evaluation, including asphalt, meadows, gravel, trees, sheets, bare soil, bitumen, bricks and shadows.

3.2. Calculation Precision

In the experiment, the calculation results of FPGA are compared with the results of GPUs, the Pavia University dataset as test data. The parameter involved in the CNN is trained on GPU platform and directly download to FPGA as inputs. The mean for each classs difference is shown in the Fig. 5. Obviously, the difference of each class is less than 0.0004. This is because the trained parameter of decimal form must be converted to 32-bit single-precision floating-point number before loading to FPGA. Table. 2 lists the classification results of the proposed FPGA based CNN. It is observed that overall average accuracy is 94.58% and overall accuracy is 92.24%. These results is identical to that implemented on GPU platform, demonstrating that the FPGA based implementation is coincide with the GPU based implementation though slightly difference may occur in the calculation of CNN.

Table 2. Class-specific accuracy (%), overall accuracy (OA) and average accuracy for Pavia University dataset.

Class	Training	Testing	Accuracy (%)
Asphalt	50	6581	96.81
Meadows	50	18599	98.59
Gravel	50	2049	82.98
Trees	50	3014	89.31
Sheets	50	1295	99.62
Bare soil	50	4979	93.02
Bitumen	50	1280	90.78
Bricks	50	3632	84.15
Shadows	50	897	94.88
AA(%)			92.24
OA(%)			94.58

3.3. Calculation Performance

The calculation speed of FPGA under different logical resources, including LUT, REG, DSP, and MRAM in Kb, is listed in Table. 3. It's clearly that the calculation speed of FPGA is increasing with the increase of the logical resources occupied. The fastest speed of the CNN model based FPGA is 999 pixels/s with the clocks frequency is 200 MHz.

Table 3. The computing speed of FPGA under different logical resources.

LUT	REG	DSP	MRAM(Kb)	Speed (pixel/s)
108866	228941	2352	68.22	396
115020	241325	2590	65.1	441
148136	309711	3542	64.78	506
182130	390490	4410	64.66	511
268730	561019	6580	64.56	760
616660	1286528	15260	64.47	999

4. CONCLUSION

A hardware model for FPGA based implementation of CNN is designed by hardware description language to classify hyperspectral images. The computation structure, implementation of different layers, weight loading scheme, and data interfere are designed on FPGA platform according to the calculation of CNN. Simulation results demonstrate that the results of FPGA based implementation is coincide with that implemented on conventional GPU platform.

5. REFERENCES

[1] J. C Harsanyi and Chein I Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach," *IEEE Transactions on Geoscience & Remote Sensing*, vol. 32, no. 4, pp. 779–785, 1994.

[2] Wei Hu, Yangyu Huang, Li Wei, Fan Zhang, and Hengchao Li, "Deep Convolutional Neural Networks for Hyperspectral Image Classification," *Journal of Sensors*, vol. 2015, no. 2, pp. 1–12, 2015.

[3] Shaohui Mei, Jingyu Ji, Junhui Hou, Xu Li, and Qian Du, "Learning Sensor-specific Spatial-Spectral Features of Hyperspectral Images via Convolutional Neural Networks," *IEEE Transactions on Geoscience & Remote Sensing*, vol. 55, no. 8, pp. 4520–4533, 2017.

[4] H Lee and H Kwon, "Going Deeper with Contextual Cnn for Hyperspectral Image Classification.," *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, vol. 26, no. 10, pp. 4843, 2017.

[5] Wenzhi Zhao and Shihong Du, "SpectralcSpatial Feature Extraction for Hyperspectral Image Classification: A Dimension Reduction and Deep Learning Approach," *IEEE Transactions on Geoscience & Remote Sensing*, vol. 54, no. 8, pp. 4544–4554, 2016.

[6] Yushi Chen, Hanlu Jiang, Chunyang Li, Xiuping Jia, and Pedram Ghamisi, "Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks," *IEEE Transactions on Geoscience & Remote Sensing*, vol. 54, no. 10, pp. 6232–6251, 2016.

[7] J. Ji, S. Mei, X. Liu, X. Li, S. Zeng, and Z. Wang, "Exploring Kernel Based Spatial Context for CNN Based Hyperspectral Image Classification," in *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Nov 2017, pp. 1–7.