Comparison of Performance of Optimized HSI CNN models on Desktop and Embedded Platforms

Jiří Čech, Martin Rozkovec
Institute of Information Technologies and Electronics
FMMIS, Technical University of Liberec
Liberec, Czech Republic
jiri.cech@tul.cz, martin.rozkovec@tul.cz

***Abstract*—We compare different platforms for inference of convolutional neural networks in this paper. We trained various neural networks to determine the material in the source hyperspectral cube. Then we convert them to inference format and compare the inference results. We used tools under Xilinx Vitis AI for FPGA implementation. We try to minimize the size of the proposed networks by pruning them and provide further comparisons. FPGA platforms show to be energy efficient but still slower than a graphics card in terms of performance.**

Keywords—Hyperspectral imaging, Neural networks, Xilinx FPGA, Vitis AI, DPU

I. Introduction

Hyperspectral imaging (HSI) is a process when series of images are taken from a scene, each containing only a part of the electromagnetic spectra emitted by visible objects. Obtained three-dimensional images are called hyperspectral cubes. A pretty simple example of such a cube is an RGB picture, where each pixel contains information about emitted spectra in red, green, and blue bands. The size of a typical cube is not dependent only on the resolution of the sensor but also on the number of spectral bands taken, which varies, in real applications, between tens to thousands.

Analysis of the cube (i.e., hyperspectral analysis) is usually used to classify objects and materials in the scene based on their spectral properties. This can be useful in areas such as food or drug control, mining, and military and surveillance applications. The analysis of the cube has one order higher performance requirements than the analysis of 2D images.

Neural networks are a modern and effective solution for image analysis. They were proven to be highly effective when dealing with large amounts of samples taken from high-resolution image sensors. Thus, it is expectable to find them effective for the hyperspectral analysis as well.

We can distinguish three main types of hyperspectral systems based on the resources required to obtain the cube and to perform its analysis:

1) Stationary solution: The hyperspectral camera, together with the lens, is connected to the control computer. Each part of the system could be controlled separately or together by a controller in the camera. This solution is used to research, develop, and test a hyperspectral camera system and is placed on an optical table for the best performance. The measured data are sent to the host computer, where they are stored and usually also analyzed.

2) Portable solution: The hyperspectral camera is small and compact, with good protection from the environment. The camera can measure hyperspectral cubes and provide data compression or data analysis before sending results to a connected computer. Reducing the size of data transmitted by the results allows remote operation of the system. That led to using this solution in the industry and the field and on drones or robots.

3) Satellite solution: This is a highly optimized solution for minimum power consumption with limited communication bandwidth [1]. It is used for long-term measurements, where the measured data are first analyzed for the quality of information and then temporarily stored in local storage or compressed and sent to an observer.

II. Motivation

Our goal is to use the FPGA platform for the classification of measured hyperspectral cubes and to compare achieved results with other platforms. Modern and effective tools for classifying hyperspectral cubes are convolutional neural networks (CNN). Those networks are required to be trained on a high-performance computer or a cluster composed of lower-performance computers. After the training phase, we can convert the trained network into an inference format and use it to compute CNN (inference) on various platforms [2][3][17]. The most popular platforms are:

1) Central Processing Unit (CPU): A simple computer with a powerful processor without a graphic card. The advantage is easy installation and good software support, but they have average performance for CNN inference.

2) Graphic Processing Unit (GPU): An extension of the CPU with a high-performance graphics card. Require installation of specific versions of compute drivers for your hardware and offer the best performance for CNN inference. Higher prices and complex cooling of the system complicate industrial use.

3) External accelerator: Is an ASIC or FPGA, which, like the GPU, extends the CPU capabilities of the CNN training or inference [3][4]. It can be easily connected via USB or elsewhere. The disadvantages of this solution are the limited interconnection bandwidth between the CPU, memory, and accelerator.

4) FPGA integration: The whole system is integrated into the FPGA [5] via the Xilinx Zynq platform. We can combine the programmability of the processor with the FPGA. This system on the chip solution minimizes data transmission delays and increases usability and power per watt. We can use a Linux-based operating system with many custom accelerators. It is programmed using Vitis AI tools [6], and we use the Deep learning unit (DPU) to accelerate the inference of CNN in the FPGA.

2021 International Conference on Applied Electronics (AE) | 978-80-261-0973-0/21/$31.00 ©202110.23919/AE51540.2021.9542900

ISBN 978-80-261-0973-0,

III. NETWORK TRAINING

We use the Tensorflow framework under the Ubuntu system, which provides us with the necessary support for network training and Vitis AI tools. We need to convert trained networks to a usable inference format and run a benchmark on available CPU, GPU, and FPGA platforms.

A. Hyperspectral cube

We used the hyperspectral dataset "Pavia University" with spatial dimensions 610×340, each containing 103 spectral bands. Each spatial pixel is classified into one of 9 classes. You can see the example of a classified image in Fig. 1.

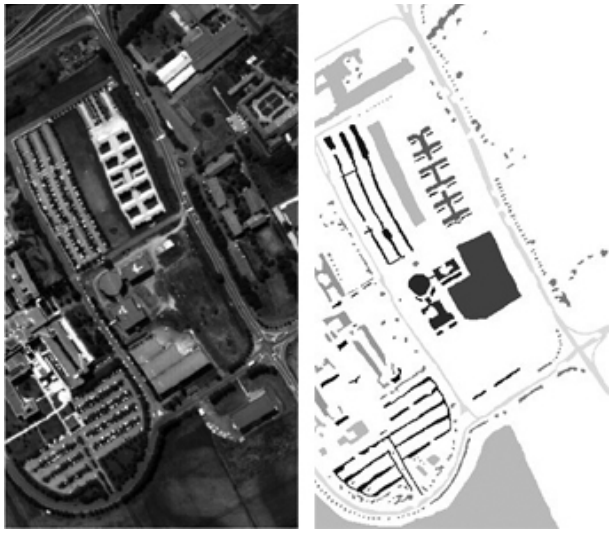

Fig. 1. An example of classification (Pavia University)

B. Trained networks

The selected dataset was randomly divided into smaller datasets. 30% of the dataset was used for training, 20% for testing, and the rest (50%) for final validation. The input vector of the CNN is a selected pixel and its surroundings with final dimensions of 7×7×103, which represent spectral and spatial information about the pixel and its neighborhood [7][8][14]. We propose three basics CNN structures based on a reference given in [8], where the first "P_CD" is a simple CNN with one convolution layer (kernel size 3×3) and one dense layer, Fig. 2. The second "P_CCCD" is a deeper CNN with three convolution layers (same kernel size 3×3). The last proposed structure, "P_CD2" is similar to "P_CD", but it has a larger stride size. That reduces the overlap of the convolution and the output size of the layer. In Fig. 3 we have stride one as "a)" and stride two as "b)". A comparison of the proposed networks in size and achieved accuracy is summarized further in TABLE I.

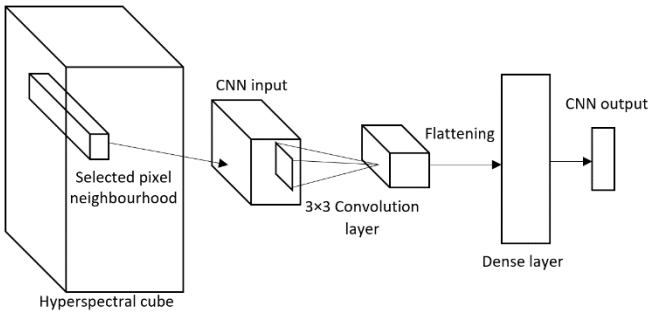

Fig. 2. Structure of simple CNN as "P_CD", (P for Paiva dataset, C for convolutional layer and D for Dense layer)

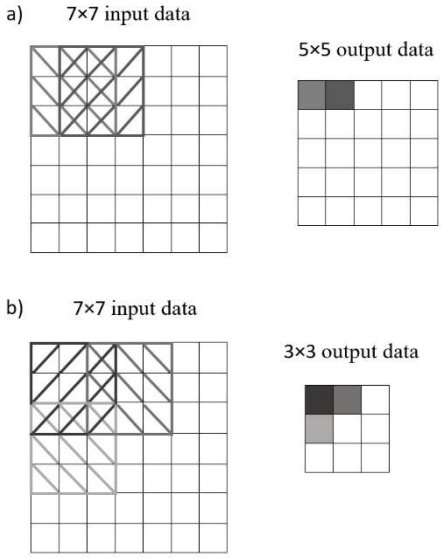

Fig. 3. Influence of different convolution stride size

C. Reduction of CNN structure

Pruning is a reduction process that removes minor weights in the neural network (near-zero values) [9]. The strategy aims to lower computation requirements. After each weight change, we must retrain the neural network to achieve higher accuracy. When the network is pruned, the rank of weights matrix for each layer may be lower, reducing the size of neurons in that layer and retrain the network. We obtain a small network structure with high accuracy and less demanding computation requirements by iterating this process. Changes in the design of the network are shown in Fig. 4, where "a)" is the original network, "b)" is with the pruned weights as the removed connections, and "c)" is after further pruning iterations and reduction of neurons.

The pruned results for the proposed networks are marked with "_p" to "P_CD_p", "P_CCCD_p" and "P_CD2_p".

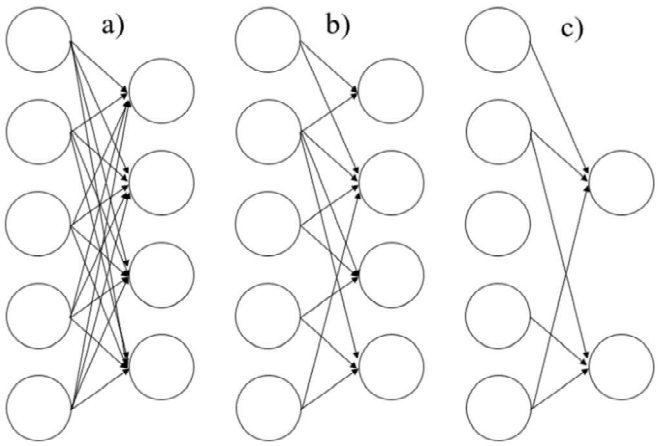

Fig. 4. Pruning process of the network

D. Reduction of CNN memory size

Network training (computational operation and stored weights) is provided in "Float32" format for high accuracy. That means a network with 1M parameters takes up 4MB of memory space. By reducing the numeric format to a fixed precision "Int8", we reduce the necessary memory space to 1MB. Simultaneously, we use multiplication for the fixed numbers, which is faster than for float numbers [10]

(regarding the maximum frequency of the HW solving the task)[10]. This process is called quantization and can slightly reduce network accuracy.

TABLE I. summarizes information about the proposed networks, their number of weights, the required computation workload, and accuracy during each reduction phase (quantization and pruning).

TABLE I. NEURAL NETWORKS ACCURACY

Network	Weights [M]	Workload [M]	Acc [%]	Quantized Acc [%]
P_CD	2.2	4.4	96.3	93.6
P_CD_p	0.9	1.9	97.5	95.8
P_CCCD	3.4	6.8	98.7	98.1
P_CCCD_p	1.0	0.6	97.0	97.2
P_CD2	1.8	3.6	97.8	96.2
P_CD2_p	0.1	0.2	97.6	96.7

E. Hardware setup

We use a high-performance solution for each of the three proposed platforms. Detailed specifications are:

1) CPU: The processor is from the X-series of Intel Core i7-7820X processors with 8 cores and 16 threads with a maximum frequency of up to 4.3GHz, power consumption up to 140W. Tensorflow provides network inference for CPUs with the Keras submodule.

2) GPU: Nvidia Geforce RTX 2070 with 8GB RAM and maximum performance 42RTX-OPS, power consumption is up to 185W. Tensorflow provides network inference for GPUs with submodule Keras and CUDA 10.2.

3) FPGA: Evaluation kit ZCU106 with Zynq UltraScale+ XCZU7EV MPSoC with available sources of 504K Logic Cells, 1728 DSP slices, power consumption up to 20W. The Xilinx provided DPU IP core [11] is used to compute CNN.

We tested several designs with different single-core DPU configurations at 300MHz, with the corresponding Peta-Linux build and recompiled proposed networks. The utilization of resources for each design is given in TABLE II. where the title of the design is based on the theoretical parallel performance of the DPU.

TABLE II. DPU DESIGNS UTILIZATION

Design	LUT [k]	FF [k]	BRAM	DSP	Power [W]
DPU_512	38	42	93	124	5.7
DPU_1024	45	56	140	232	6.6
DPU_2304	53	77	215	436	8.7
DPU_4096	72	109	310	704	13

The smallest configuration is even usable on boards like Zedboard with Zynq XC7Z020. For further comparison with CPU and GPU, we select only the smallest design DPU_512, and the largest design DPU_4096.

IV. NETWORKS INFERENCE

We provide inferences for all networks on CPU, GPU, and FPGA (DPU_512, DPU_4096) platforms. We measured the required time for each test for the first classification "cold run" and the average time for the next 20 runs, "warm run". The same dataset of selected 33k hyperspectral pixels was used in each run. The results of the average computation time for inference one hyperspectral pixel are shown in TABLE III. In parentheses is the cold run time, which is slower due to the initialization of the computation.

TABLE III. INFERENCE COMPUTATION TIME [μS]

Network	CPU	GPU	DPU_512	DPU_4096
P_CD	119 (138)	15.5 (165)	539 (580)	350 (420)
P_CD_p	105 (118)	14.1 (154)	228 (296)	162 (232)
P_CCCD	501 (509)	25.2 (373)	883 (965)	526 (610)
P_CCCD_p	174 (181)	15.0 (189)	123 (195)	84 (156)
P_CD2	240 (253)	16.0 (166)	420 (493)	284 (354)
P_CD2_p	121 (128)	13.9 (143)	55 (123)	46 (114)

*All in [μs], time of the cold run is in the parentheses

A comparison of computational times related to CPU is shown in Fig. 5. Where we can see the change of the ratio for unpruned and pruned networks. A particularly significant difference has GPU speedup, which is significantly reduced by pruning and increased dramatically for the FPGAs.

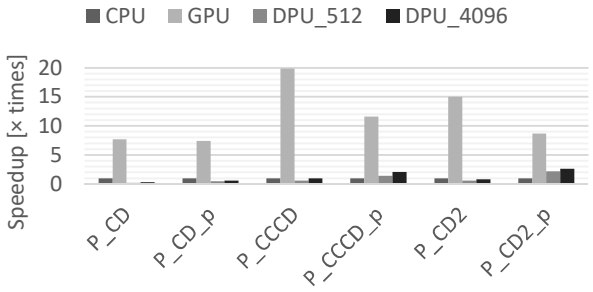

Fig. 5. Inference speedup related to CPU

In addition to computational speed, there is another critical parameter, energy efficiency, which is an essential criterion in cloud computing and low-performance applications. A comparison of energy efficiency related to CPU is shown in Fig. 6.

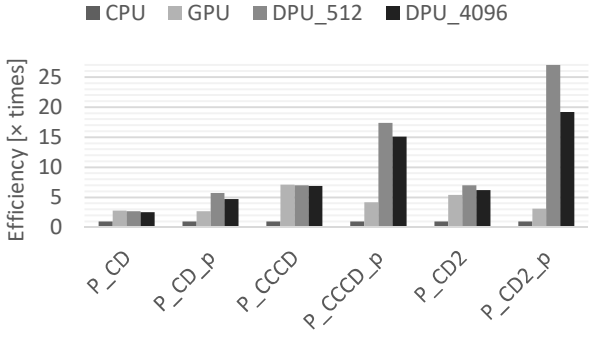

Fig. 6. Energy efficiency related to CPU

However, this does not mean that the CPU is bad for CNN inference. Of the tested platforms, the CPU has the shortest initialization time of 11μs, and the FPGA had 85μs and GPU 171μs.

V. Conclusions

We proposed three different structures of CNN, which were trained for the classification of the "Pavia University" dataset and achieved an overall accuracy of 98%. We provide network inference on three different high-performance platforms, CPU, GPU, and FPGA, where we used a single-core DPU accelerator. In comparison, the GPU platform was the fastest with the highest performance of 600GOPS, and the FPGA platform was the slowest. To change this, we provide network pruning, which significantly reduces their size. Unfortunately, compared to pruned CNN, the GPU was still faster than the FPGA, but now only three times.

More interesting is the comparison in energy efficiency, where for pruned CNN, FPGA was ten times better than GPU. On the other hand, the CPU platform had the lowest initialization time of the computing.

So far, we have achieved a maximum FPGA performance of 55GOPS, which is not bad compared to a similar solution [12][13][16]. They are implementing larger image processing networks such as AlexNet [15][18], VGG, etc. To further improve performance, we plan to test the triple-core DPU accelerator configuration, which will allow us to parallelize processes further.

Acknowledgment

This work was (partly) supported by the Student Grant Scheme at the Technical University of Liberec through project nr. SGS-2019-3017.

References

[1] R. Sandau, “Status and trends of small satellite missions for Earth observation,” Acta Astronautica., vol. 66, pp. 1–12, 2010.

[2] E. Torti, M. Acquistapace, G. Danese, F. Leporati and A. Plaza, "Real-Time Identification of Hyperspectral Subspaces," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 6, pp. 2680-2687, June 2014.

[3] E. Nurvitadhi, D. Sheffield, Jaewoong Sim, A. Mishra, G. Venkatesh and D. Marr, "Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC," 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, 2016, pp. 77-84.

[4] V. Parmar, J. Ahn and M. Suri, "Hyperspectral Image Classification for Remote Sensing Using Low-Power Neuromorphic Hardware," 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019, pp. 1-7.

[5] S. Pei, R. Wang, J. Zhang and Y. Jin, "FPGA-based acceleration for hyperspectral image analysis," 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, 2017, pp. 324-327.

[6] Xilinx, Vitis AI, 2020. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/vitis_ai/1_0/ug1414-vitis-ai.pdf

[7] Y. Chen, H. Jiang, C. Li, X. Jia and P. Ghamisi, "Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks," in IEEE Transactions on Geoscience and Remote Sensing, vol. 54, no. 10, pp. 6232-6251, Oct. 2016.

[8] X. Yang, Y. Ye, X. Li, R. Y. K. Lau, X. Zhang and X. Huang, “Hyperspectral Image Classification With Deep Learning Models,” in IEEE Transactions on Geoscience and Remote Sensing, vol. 56, no. 9, pp. 5408-5423, Sept. 2018.

[9] R. Reed, "Pruning algorithms-a survey," in IEEE Transactions on Neural Networks, vol. 4, no. 5, pp. 740-747, Sept. 1993.

[10] Yang, Jiwei, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang and Xian-Sheng Hua. “Quantization Networks.” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019): 7300-7308. arXiv:1911.09464

[11] Xilinx, DPUv3.1, 2020. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_1/pg338-dpu.pdf

[12] K. Tajiri and T. Maruyama, “FPGA Acceleration of a Supervised Learning Method for Hyperspectral Image Classification,” 2018 International Conference on Field-Programmable Technology (FPT), Naha, Okinawa, Japan, 2018, pp. 270-273.

[13] R. DiCecco, G. Lacey, J. Vasiljevic, P. Chow, G. Taylor and S. Areibi, "Caffeinated FPGAs: FPGA framework For Convolutional Neural Networks," 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, 2016, pp. 265-268, doi: 10.1109/FPT.2016.7929549.

[14] Liu S., Chu R.S.W., Wang X., Luk W. (2019) Optimizing CNN-Based Hyperspectral Image Classification on FPGAs. In: Hochberger C., Nelson B., Koch A., Woods R., Diniz P. (eds) Applied Reconfigurable Computing. ARC 2019. Lecture Notes in Computer Science, vol 11444. Springer, Cham.

[15] F. Al-Ali, T. D. Gamage, H. W. Nanayakkara, F. Mehdipour and S. K. Ray, "Novel Casestudy and Benchmarking of AlexNet for Edge AI: From CPU and GPU to FPGA," 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2020, pp. 1-4, doi: 10.1109/CCECE47787.2020.9255739.

[16] M. Blott, et al.,"Evaluation of Optimized CNNs on Heterogeneous Accelerators using a Novel Benchmarking Approach" in IEEE Transactions on Computers, vol. , no. 01, pp. 1-1, 5555.

[17] Kamel Abdelouahab, Maxime Pelcat, François Berry, Jocelyn Sérot. Accelerating CNN inference on FPGAs: A Survey. 2018. ffhal-01695375v2f

[18] Sparsh Mittal, A Survey on Optimized Implementation of Deep Learning Models on the NVIDIA Jetson Platform, January 2019Journal of Systems Architecture 97