Design of Convolutional Neural Network Based on FPGA

HASNAE EL KHOUKHI, YOUSSEF FILALI, MY ABDELOUAHED SABRI, ABDELLAH AARAB

(LISAC) :Laboratoire d'Informatique, Signaux, Automatique et Cognitivisme Faculty of Sciences Dhar-Mahraz, Sidi Mohamed Ben Abdellah University, Fez, MOROCCO

Abstract—recently with the rapid development of artificial intelligence AI, various deep learning algorithms represented by Convolutional Neural Networks (CNN) have been widely utilized in various fields, showing their unique advantages; especially in Skin Cancer (SC) imaging Neural networks (NN) are methods for performing machine learning (ML) and reside in what's called deep learning (DL). DL refers to the utilization of multiple layers during a neural network to perform the training and classification of data. The Convolutional Neural Networks (CNNs), a kind of neural network and a prominent machine learning algorithm go through multiple phases before they get implemented in hardware to perform particular tasks for a specific application. State-of-the-art CNNs are computationally intensive, yet their parallel and modular nature make platforms like Field Programmable Gate Arrays (FPGAs) compatible with the acceleration process. The objective of this paper is to implement a hardware architecture capable of running on an FPGA platform of a convolutional neural network CNN, for that, a study was made by describing the operation of the concerned modules, we detail them then we propose a hardware architecture with RTL scheme for each of these modules using the software ISE (Xilinx). The main objective is to show the efficiency of such a realization compared to a GPU based execution. An experimental study is accomplished for the PH2 database set of benchmark images. The proposed FPGA-based CNN design gives competitive results and shows well its efficiency.

Keywords—Skin imaging; Machine Learning; Convolutional Neural Networks; FPGA; VHDL

Received: March 28, 2021. Revised: January 21, 2022. Accepted: February 15, 2022. Published: March 23, 2022.

1. Introduction

In recent times, due to new technologies and the availability of increasingly great computing power, Machine Learning (ML) has won a vital role in numerous fields, such as Computer Vision (CV) and health. In particular applications, Deep Neural Networks (DNNs) have been proven to surpass conventional ML methods, and even human experts are offering the opportunity to implement complex algorithms without compromising real-time computing on embedded devices. Indeed, the area of CV has extremely evolved, principally because of promising practical results achieved when using DNNs, such as the Convolutional Neural Networks (CNNs) [1].

Convolutional Neural Network (CNN) is a deep learning algorithm that recently has brought revolution in computer vision area. CNN shows excellent performance in solving complex computer vision problems including image classification [2] [3], object detection [3] [4], semantic segmentation [4] and image retrieval [5].

The CNNs initially inspired by the discovery made by Hubel and Wiesel of neurons sensitive to regional aspects and selective in orientation in the visual system of the cat, present an extension of Multi-Layer Perceptron (MLPs) designed to extract the characteristics of input images automatically. They are invariant to slight image distortions implementing the notion of weight-sharing and allowing reducing the number of network parameters considerably [6]. It was firstly used by Fukushima with his neocognitron, in which the weights are required to be equal to detect lines, points, or corners at all possible locations in the image, thus implementing weight sharing [7]. More precisely, the method consists of multiple layers of computationally intensive convolution operations followed by memory intensive classification layers, which still challenge the state-of-the-art computing platforms to accomplish real-time performance with high energy efficiency [8].

Because of their flexible architecture and intrinsic parallelism, Field Programmable Gate Array (FPGAs) presented a technology offering flexible and fine-grained hardware that can exploit parallelism inside several IP algorithms [9],[10]. FPGAs are rather distinct from other families of programmable circuits offering the highest level of logic integration. Many applications, such as on-board systems in autonomous cars, require high energy efficiency and real-time performance. Therefore, hardware accelerators like the Graphical Processing Unit (GPU), FPGAs, and Application-Specific Integrated Circuits (ASICs) have been used to develop CNNs throughput [11].

In accordance with this purpose, the CNN classification method is used by IP to make it easier for experts to comment on diseased cells in the medical field [12]. It allows them to make a simpler interpretation by classifying complex images, which facilitates early diagnosis as soon as it reduces death and treatment costs.

In this paper, we propose a hardware architecture that improves the speed of CNN through multiprocessing of repetitive operations, including convolution operations, that account for real-time CNN processing in embedded environments in CNN algorithms.

The remainder of the paper is organized as follows. Sections II give an overview of the CNN method. In section III, the proposed FPGA-based CNN design is explained, implementation details and obtained results are also given. To conclude, Section IV presents the conclusion.

2. Overview on the CNN

CNN is a type of advanced Artificial NN (ANN) and differs from regular ones in terms of signal flow between neurons. It currently presents the most powerful model for classifying images. Typical NN transmits signals along the input-output channel in only one direction called a "forward flow," without allowing them to loop back through the system [13].

Convolutional Neural Network (CNN) is a deep learning algorithm that excels at image classification. CNN uses convolution to extract image features and then uses certain features to distinguish objects. It is intended to learn spatial hierarchies of features [14] automatically and adaptively by preparation. When the characteristics of an image vote for the most likely class to which the image belongs, the image may be categorized.

The approach consists of various layers, such as convolutional and fully-connected ones, and usually starts with a convolutional layer, where it takes input images and decomposes them into different maps of entities such as lines, edges, curves, etc. It is applied to the entity maps extracted over the entire network. Maps of extracted features of the last layer (usually a fully connected layer) are classified into output classes using a classifier such as the SoftMax classifier [15]. Most of the operations are performed, and pooling layers, which are used to avoid overfitting; and a classification layer, classify final results.

Essentially a typical CNN consists of two parts:

The convolution and pooling layers, whose goals are to extract features from the images. These are the first layers in the network.

The final layer(s), which are usually Fully Connected NNs, whose goal is to classify those features.

Usually in the context of CNNs, ReLU, except for the activation function of the final layer, which is selected according to the nature of the problem. The most common cases are:

- Sigmoid activation functions work for binary classification problems.
- Softmax activation functions work practically for both binary and multi-class classification problem.

CNNs usually start with a convolutional layer, where it takes input images and decompose them into different feature maps .Multiple processes are applied to the extracted feature maps throughout the entire network. Extracted feature maps from the last layer are classified into output classes employing a classifier like Softmax classifier.

Principally, there are four layers implemented for the CNN algorithm, the convolution layer (CONV), the correction layer (RELU), the pooling layer (POOL), and the fully-connected layer (FC). So, the basic CNN architecture is formed by a stack of the cited independent processing layers (see Fig. 1):

- Input Layer, this layer holds the raw input of image, 28 width, and 28 height .
- CONV layer, which processes the data of a receiver field,
- Activation function Layer, Linear Correction Unit often misnamed “RELU” about the activation function,
- POOL layer, which allows the information to be compressed by reducing the intermediate image (often by subsampling),
- FC layer, which is a perceptron-type layer,
- Softmax (SOF) layer, a type of classification layer which presents the final output layer for the CNN,
- Output layer,

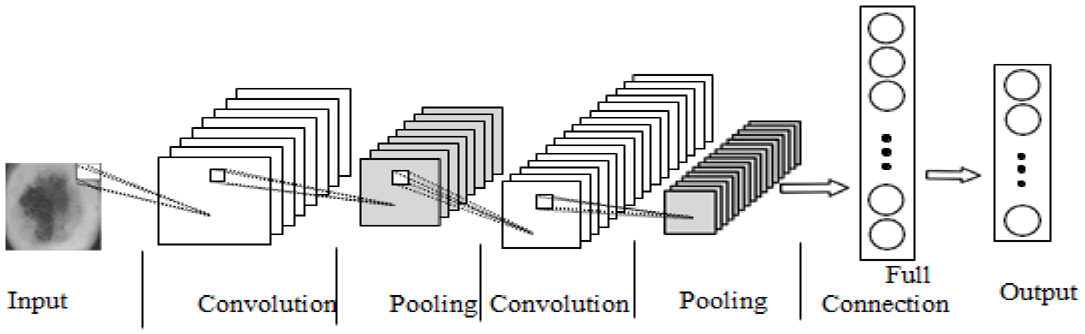


Fig. 1. The CNN architecture

2.1. Convolution layer (CONV)

The CONV layer is a critical component of CNNs and is often the first and most computationally intensive layer. Its persistence is determined by detecting the presence of a set of features in the input images. In fact, it does a convolution operation on the input and then passes the result to the next layer. Convolution filtering is performed by dragging a window that represents the feature on the image and computing the convolution product between the feature and each part of the scanned image [16].The tensors and variables used in this work are listed in table 1.

TABLE I. TENSORS INVOLVED IN THE INFERENCE

C	***Input feature maps***	M ·Wf ·Hf
G	***Output feature maps***	N · Wf · Hf .
K	***Learned filters***	M · Wk · Hk
b	***Learned biases***	n

A CONV layer contains M input and N output channels. As given away in Eq (1) below, we remark that Wf×Hf feature map for every input channel. The M×Wf×Hf input convolves into one of the output channels and generates a Wf×Hf output feature map with a kernel size of a M×Wk×Hk. The weight of the neural network is trained in the convolution kernels. The output size of N×Wf×Hf is achieved by N kernels. Fig. 2 shows a convolution with a single kernel.

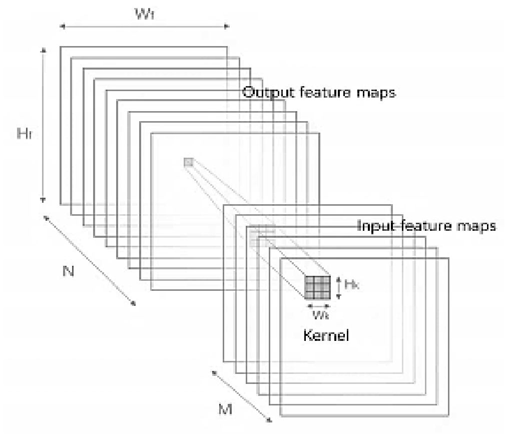


Fig. 2. Structure of Standard Convolutional Layer

The convolution and activation part of CNN is directly inspired by visual cortex cells in neuroscience [20]. This is also the case with the pooling layer, which is periodically inserted between successive conv layers (l).

$$G^{(l)}{}_{[n,x,y]} = \sum_{i=1}^{Wk} \sum_{j=1}^{HK} \sum_{m=0}^{M-1} C^{(l)}{}_{[m,x+i,j+y]} \,.\, K^{(l)}{}_{[n,i,j]} + b^{(l)}{}_{[n]} \quad (1)$$

2.2 Rectified Linear Unit (ReLu) layer

To guarantee nonlinearity in the network as well as to get rid of unnecessary information and Activation function is applied to each input pixel [17]. When a network is activated, it becomes nonlinear. It helps the network learn high-order polynomials [18], allowing it to learn and execute more complex tasks.

Let u be a real number,

$$ReLU(u) \begin{cases} u & u > 0 \\ 0 & u \leq 0 \end{cases} \quad (2)$$

$$SeLU = \lambda \begin{cases} u & u > 0 \\ \alpha(e^{u} - 1) & u \leq 0 \end{cases} \quad (3)$$

$$Sigmoid(u) = \frac{1}{1+e^{-u}} \quad (4)$$

Rectified Linear Units (ReLU) Eq(2), Scaled Exponential Linear Unit (SeLU) Eq(3), Sigmoid Eq(4), and others are common activation functions.

2.3 Pooling layer (MAX_POOL)

POOL layer is another building block of a CNN, permanently positioned between two layers of convolution. It receives multiple feature maps as input and adds each of them to the pooling activity and is used to progressively decrease the image size and retains its essential characteristics [19]. It is shown in Fig.3.

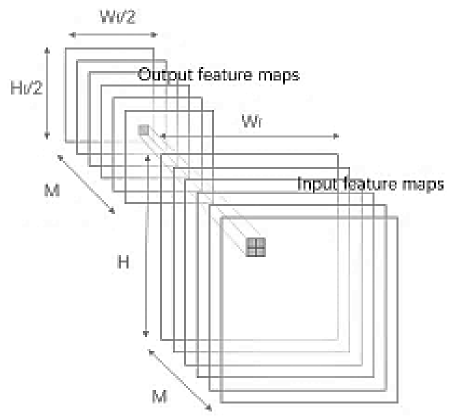


Fig. 3. Structure of Pooling Layer

The convolution and activation part of CNN is directly inspired by visual cortex cells in neuroscience [20]. This is also the case with the pooling layer, which is periodically inserted between successive conv layers.

2.4 Fully-connected layer (FC)

FC layer is permanently the last layer of neurons, convolutional or not, so it is not characteristic of a CNN. It accepts a vector as input and creates a new one as output by applying a linear combination and then probably an activation function to the values received at the input. One or more FC Layers are typically introduced as the final layers of the CNN, committed to the task of the supervised classification [21]. Fig.4 depicts the relation in a completely connected layer with M input neurons and N output neurons.

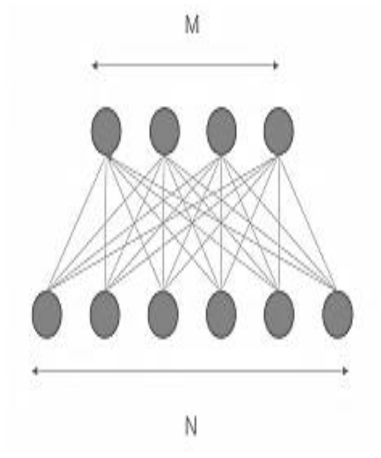


Fig. 4. Structure of Fully Connected Layer

Equation 5 displays the expression for each neuron in input X and output Y, where w represents the weight of each relation and b represents the bias of each output neuron.

$$Y_{[n]} = \sum_{m=0}^{M-1} X_{[m]}.W_{[m,n]} + b_{[n]} \quad (5)$$

2.5 Final Classification (Activation Layers SOF)

The final classification step is performed using the Softmax (SOF)functions work practically for both binary and multi-class classification problem.

3. Proposed FPGA-BASED CNN Approach and Implementation Details

3.1 Proposed methodology

Generally, the abilities of FPGAs structure to exploit temporal and spatial parallelism make them commonly used as implementation platforms for multiple IP real applications. Such parallelization is topic to the processing mode and hardware constraints of the system, forcing the designer to reformulate the algorithm. More parallelism means higher system performance and, consequently, more throughputs.

In this paper, we present design architecture for the CNN on FPGA. In fact, an FPGA-based CNN implementation for the classification of SC images, are accomplished and tested, showing well its efficiency and giving competitive results. We try to provide VHDL-based FPGA hardware implementations for several modules on the CNN network. The objective of this study is to evaluate the feasibility of this implementation and to prove its efficiency compared to a GPU-based implementation.

The CNNH involves various layers, such as CONV and FC. Most of the operations are performed, and POOL layers, which are used to avoid overfitting. In addition, a classification layer is applied for classifying final results.

The CNNH developed and implemented the parallel architecture and the convolutional network. Details are as follow:

- Pre-processing

Step 1: Reading input image,

- Features extraction

Step3: Applying the COV1 layer process,

Step4: Applying the RELU layer process,

Step5: Applying the MAX_POOL1 layer process.

Step6: Applying the COV2 layer process,

Step7: Applying the RELU layer process,

Step8: Applying the MAX_POOL2 layer process.

- Classification

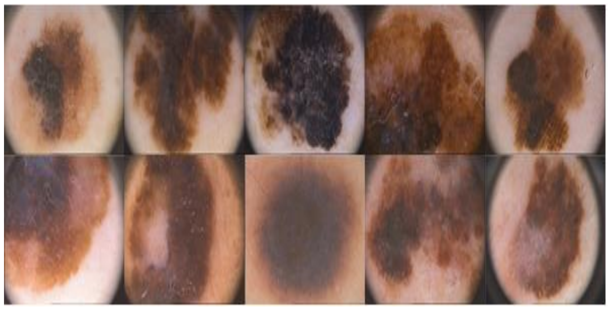
Step9: Applying the FC process,

Step10: Applying the classification layer process using the SOF layer.

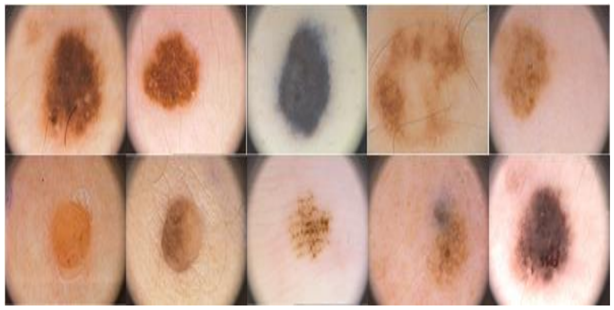
3.2 Experimental Results

The proposed CNNH algorithm is an adaptive variant of CNN method, based on a hardware implementation using FPGA. The performed algorithm is designed with the VHDL language and simulated on the Nexys Video is a more powerful board which utilizes the Xilinx Artix-7 XC7A200T FPGA and is shown in Fig.5. The work of implementing a CNN neural network was initiated by the Xilinx ISE software, in which we used vivado HLS (High Level Synthesis) to synthesize and estimate the resources occupied by the FPGA. The algorithm was tested on PH2 (see Fig.6) Database images of 200 images: 160 non-melanomas (80 common nevi, 80 atypical nevi), and 40 melanomas SC, with their real sizes764*575[22]. We are directing an experimental study on the effectiveness of the implemented algorithm by presenting and interpreting the obtained results.

Fig. 5. Artix-7 XC7A200T FPGA



(a) Melanoma



(b) Not- Melanoma

Fig. 6. A sample of Database images PH2 of tow catégories (a) for Melanoma and (b) for Not- Melanoma

3.3 Generalized model analysis

Unfortunately, not all aspects of CNNs are so intuitive to find out and understand. for instance, there's always an extended list of parameters that require to be set manually to permit the CNN to perform better, e.g. what percentage features to settle on and the way many pixels to think about in each feature etc.

The parameters of the proposed CNNH architecture are summarized in Table2 below:

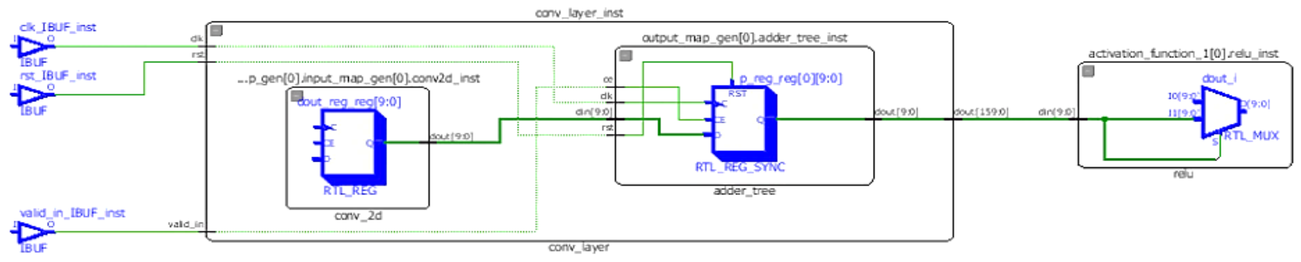
Table 2.Parameters of the proposed CNNH architecture

Layer	***Input Size (map)***	***Kernel Size***	***Same (padding)***	***Layer Activation***	***Output Size (map)***
Input Image	28*28	---	---	---	24*24
COV1	24*24	5	0	RELU	12*12
POOL1	12*12	2	0	MAX_POOL	8*8
COV2	8*8	5	0	RELU	4*4
POOL2	4*4	2	0	MAX_POOL	1*1
FC	1*1	1	0	RELU	1*1
Classification	1*1	---	---	SOF	1*1

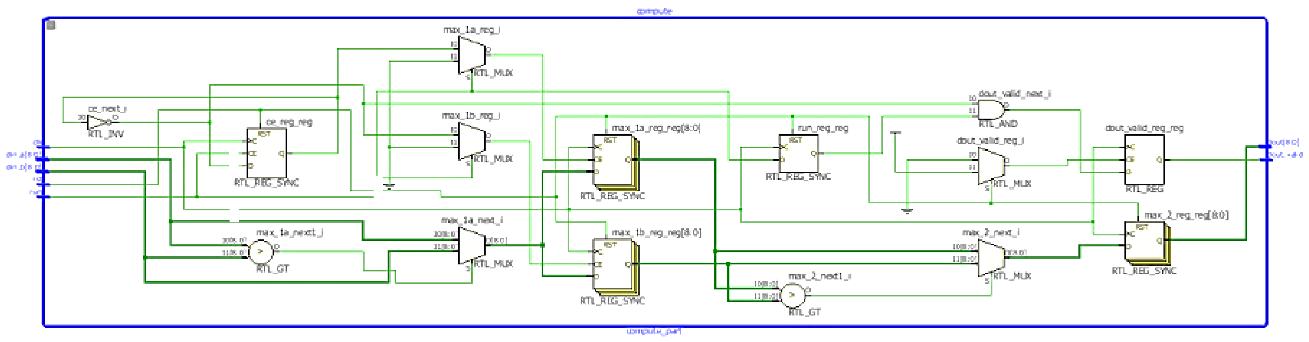
The different CNN modules realized in VHDL for the purpose of hardware implementation on FPGA.

Before explaining the operation of each convolution module, we specify that the image is loaded from an external memory pixel by pixel and that a pixel of the input image is represented by 9bits. The filter used is the kernel filter represented by 5 bits.

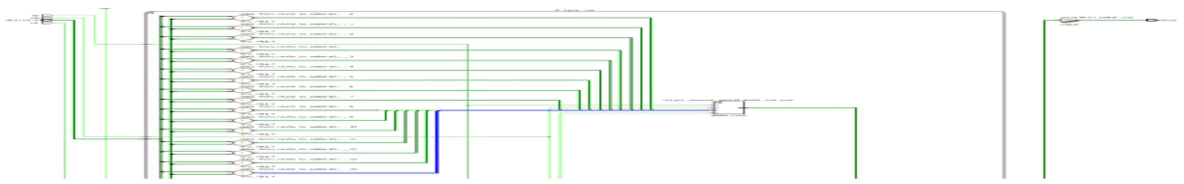
Figure 7 shows a diagram of the CNNH modules used herein. The convolution method differs only within the size of the input buffer (see a), (b) may be a diagram of the hardware module of the most important pooling layer. To perform the utmost pooling and (c) may be a diagram of the fully connected layer hardware module. The weights and have map values utilized in the fully connected layer are stored within the block memory and are calculated one by one from the memory.



(a). RTL Convolution Layer Module and RTL RELU



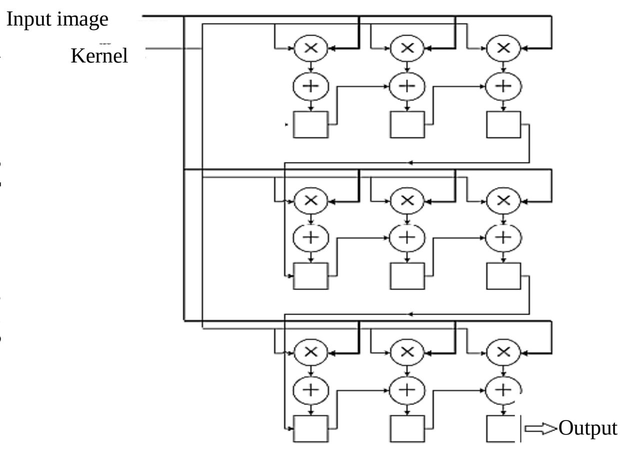
(b). RTL Maxpooling



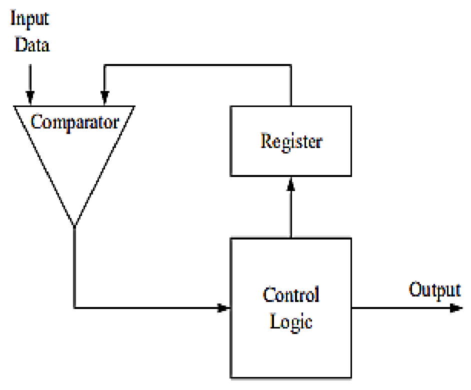
(c). RTL FC layers

Fig. 7. Hardware architecture of the implementation of CONV, RELU, POOL, and FC layers

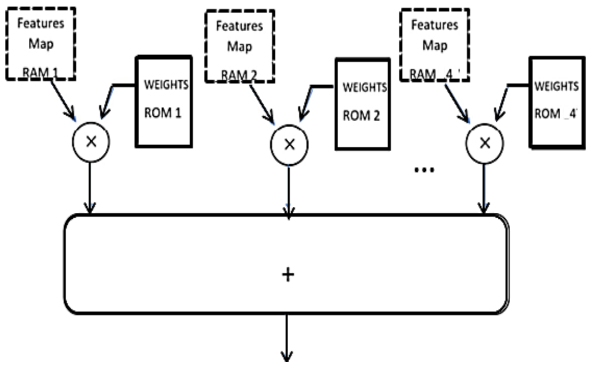
The CNNH algorithm, whose parameters ar described in the previous Table 1, was modeled in VHDL code. In our FPGA implementation, we realize operations with the circuit shown in Fig. 9 composed of blocks called convolutional Processing Elements (PE) (see Fig. 8). Through a Register Transfer Level (RTL) design, the solution was optimized and exported as a processing logic (PL). FPGAs use hardware for configurable PL, thus making these very fast and with flexible processing devices.



(a)



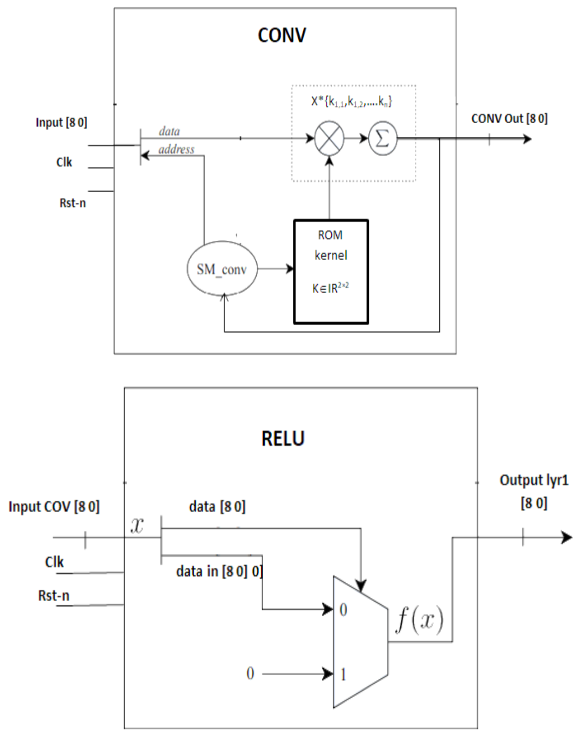
(b)



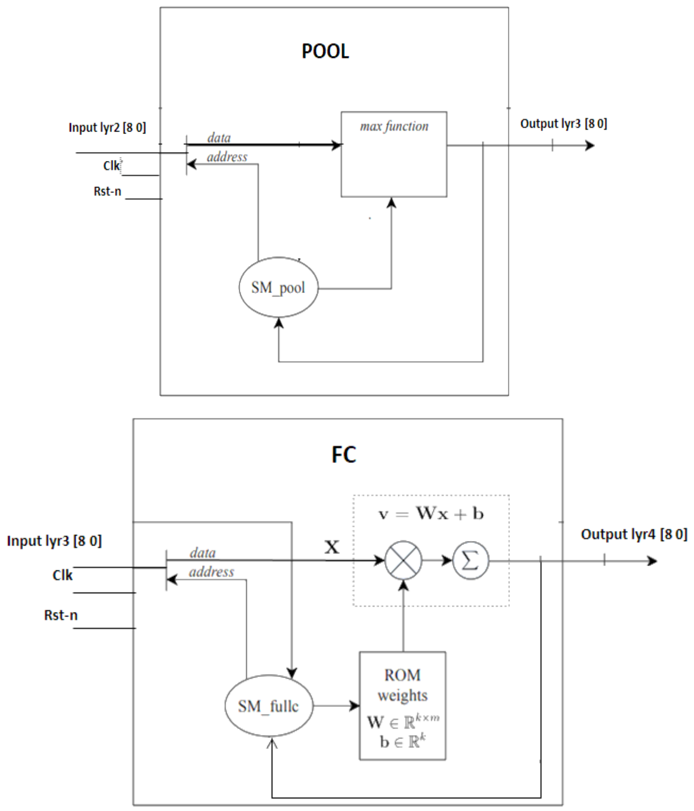
(c)

Fig. 8. Convolutional Processing Elements at the (a)PE in CONV, (b)PE in POOL, and (c)PE in FC levels

The detailed hardware architecture for our proposed design is shown in Figure 9.



(a). Hardware architecture of the implementation of CONV and RELU layers



(b). Hardware architecture of the implementation of POOL and FC layers

Fig. 9. Hardware architecture of the implementation of CONV, RELU, POOL, and FC layers

The FPGA has the advantage of accelerating computation through a variety of solutions. The first solution is to use parallelism and pipelining in the execution of computational operations. This type of solution is adopted in this article. Indeed, in the last chapter, we have defined the different implemented CNN modules. We have presented each block of each module and its operation in detail. We have given the RTL diagram of the ISE vivado (2016) software, FPGA The summary of the required resources. The proposed design is successfully implemented in FPGA, and the gained results illustrate well that the proposed CNNH algorithm produces higher quality classified images.

4. Conclusions

In recent years CNN's algorithms have shown extremely progress due to their electiveness at complex image recognition fields. They are currently adopted to solve an ever greater number of problems, ranging from speech recognition to image segmentation and classification. The continuing increasing amount of processing required by CNN's creates the field for hardware support methods. This paper primarily emphasizes on a hardware implementation of the CNN method using FPGA. We have proposed the CNNH approach, an FPGA-based CNN design for the classification of SC images. An experimental study and a qualitative analysis of the effectiveness of the implemented method on vivado (2016) software are directed. In all configurations, our method performs competitive results.

References

[1] G. Lo Sciuto, G. Susi, G. Cammarata e G. Capizzi: A spiking neural network-based model for anaerobic digestion process, in IEEE 23rd Int. Symp. on power electronics, electrical drives, automation and motion (SPEEDAM), 2016.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. Commun. ACM, 60(6):84–90, 2017.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, pages 248–255, 2009.

[4] Bharath Hariharan, Pablo Arbelaez, Ross B. Girshick, and Jitendra Malik. Simultaneous detection and segmentation. CoRR, abs/1407.1808, 2014.

[5] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, 4546 REFERENCES The Netherlands, October 11-14, 2016, Proceedings, Part I, pages 3–20, 2016.

[6] Kamel, A.; Maxime, P.; Jocelyn, S.; François, B. Accelerating CNN inference on FPGAs: A Survey; Technical Report; Universite Clermont Auvergne: Clermont-Ferrand, France, 2018.

[7] K. Fukushima. Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36 :193–202, 1980.

[8] Cardarilli, G.C., Cristini, A., Di Nunzio, L., Re, M., Salerno, M., Susi, G.: Spiking neural networks based on LIF with latency: Simulation and synchronization effects (2013) Asilomar Conference on Signals, Systems and Computers, pp. 1838-1842.

[9] Khanal, G., Acciarito, S., Cardarilli, G.C., Chakraborty, A., Di Nunzio, L., Fazzolari, R., Cristini, A., Susi, G., Re, M. ZnO-rGO composite thin film resistive switching device: Emulating biological synapse behavior (2017) Lecture Notes in Electrical Engineering, 429, pp. 117-123

[10] H. El Khoukhi, & M. A. Sabri, Comparative Study Between HDLs Simulation And Matlab For Image Processing, IEEE 2018 International Conference On Intelligent System And Computer Vision (ISCV), 2018.

[11] Yann, L.; Léon, B.; Yoshua, B.; Patrick, H. Gradient-Based Learning Applied to Document Recognition. Proc. IEEE 1998, 86, 2278–2324.

[12] Nishchal, K.V.; Teena, S.; Shreedharkumar, D.R.; Al, S. Object Identification for Inventory Management using Convolutional Neural Network. In Proceedings of the 2016 IEEE Applied Imagery Pattern Recognition Workshop, Washington, DC, USA, 18–20 October 2016.

[13] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, “Optimizing FPGA-based accelerator design for deep convolutional neural networks,” in ACM FPGA, 2015, pp. 161–170.

[14] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. Insights into Imaging, 9(4):611–629, Aug 2018.

[15] Khoukhi, H.E., Filali, Y., Sabri, M.A., Aarab, A. (2020). Design and implementation of content-based image retrieval on fpga card. International Journal of Advanced Trends in Computer Science and Engineering, 9 (5), pp. 8085-8093. https://10.0.119.70/ijatcse/2020/169952020.

[16] Ying, W.; Jie, X.; Yinhe, H.; Huawei, L.; Xiaowei, L. DeepBurning: Automatic Generation of FPGA-based Learning Accelerators for the Neural Network Family. In Proceedings of the IEEE Design Automation Conference, Austin, TX, USA, 5–9 June 2016.

[17] Zhang, M.; Li, L.; Wang, H.; Liu, Y.; Qin, H.; Zhao, W. Optimized Compression for Implementing Convolutional Neural Networks on FPGA. Electronics 2019, 8, 295.

[18] Matthieu C., Yoshua B., and Jean-Pierre D. Training deep neural networks with low precision multiplications. arXiv preprint arXiv:1412.7024, 2014.

[19] El Khoukhi H., Idriss F.M., Yahyaouy A., Sabri M.A. (2020) An Efficiency Study of Adaptive Median Filtering for Image Denoising, Based on a Hardware Implementation. In: Bhateja V., Satapathy S., Satori H. (eds) Embedded Systems and Artificial Intelligence. Advances in Intelligent Systems and Computing, vol 1076. Springer, Singapore. https://doi.org/10.1007/978-981-15-0947-6_9

[20] David H Hubel and Torsten N Wiesel. Receptive elds, binocular interaction and functional architecture in the cat’s visual cortex. The Journal of physiology, 160(1):106–154, 1962.

[21] A. Dundar; J. Jin; B. Martini; E. Culurciello, "Embedded Streaming Deep Neural Networks Accelerator With Applications," in IEEE Transactions on Neural Networks and Learning Systems , vol.PP, no.99,pp.1-12.

[22] J. Qiu et al., “Going deeper with embedded fpga platform for convolutional neural network,” in ACM International Symposium on FPGA, 2016.