Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 45(2), 2018, Pages 263–274
ISSN: 1223-6934

FPGA design and hardware implementation of a convolutional neural network for classification of saccadic eye movements

Carlos Cano, Ruxandra Stoean, and Gonzalo Joya

Abstract. The paper presents an efficient design and implementation of a convolutional neural network on an FPGA device. The aim is not only theoretical but also practical, since the solution will be used in a medical clinic dealing with SpinoCerebellar Ataxia type 2 as part of a larger project. Hence, the current work targets both high learning capabilities as well as portability. The former has been tackled through the apppointment a convolutional neural network while the latter is concerned with the hardware implementation of the complex network on a FPGA. The preliminary results encourage the further exploitation of the proposed solution.

1991 *Mathematics Subject Classification.*
Key words and phrases. Field programmable gate arrays (FPGA), Convolutional Neural Network (CNN), Saccade classification, Hardware optimization, High-level Synthesis (HLS) , Algorithm design and analysis.

1. Introduction

Spino Cerebellar Ataxia type 2 (SCA 2) is a neurodegenerative disease. As such, the condition has no cure and produces a gradual and increasing alteration of the nervous system. Under these circumstances, there is a high interest in the development of monitoring and diagnosis systems that allow an early detection of the disease, for timely rehabilitation planning, as well as portability, for direct use in hospitals/clinics and a minimal need for patient displacement and queuing [1].

One non-invasive diagnostic technique that allows the possibility to register the weak electrical potentials generated by the eye movement when following the trajectory of an object (saccades), is electrooculography. Such obtained saccadic records can be further used for classification of patients into healthy, presymptomatic and sick. The principal aim for the artificial learning support is to allow the early detection of people in the primary stages of the disease, where symptoms are so weak that they have no external manifestation (i.e. presymptomatic). The practical target is to implement the learning algorithm on a portable device for a direct use in the medical unit. The complete task is being handled by a team of researchers from the University of Malaga (Spain), Technical University of Manabi (Ecuador) and University of Craiova (Romania). Medicine has obtained very good results from the high potential of machine learning [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], however preliminary models with such traditional techniques have shown the complexity of this particular classification task [14].

Received July 4, 2018. Accepted November 30, 2018.

Since convolutional neural networks (CNN) have lately also become very popular for medical problems [15], [16], [17], the current algorithmic direction of the team explores its applicability for this sequence classification. A first theoretical test of its implementation on a portable FPGA device has been described in [18].

The focus of the current study is on the complete practical integration in the device of all the modules designed previously in [18]. Moreover, this paper outlines in detail all the main features of each part of the complete design. Finally, the work proves once again the computational advantage of the FPGA over a GPU-based implementations [19], especially as regards heterogeneous parallelism and custom data types, which is a particularly hard task for GPUs [20].

The rest of this document is organized as follows. In the *Hardware vs Software solution* section, the main ideas about hardware implementation are pointed out, explaining the key motivation for doing this project in a hardware language and using the chosen device. In *Proposed architecture*, the complete architecture is shown and the behavior of the design is exposed. *Convolutional neural network block* analyzes the main ideas in the development of the hardware block; also the optimization mechanism and the methodology are described. *Processor and interconnection* analyzes the resources involved in the interconnection between processor part and logic part. Also, the essential knowledge of the processor part is introduced. The results of the tests carried out are detailed in *Results*, highlighting the outcome obtained in each part of the methodology. Finally, the *Conclusions* section summarizes the main ideas and conclusions of the work.

2. Hardware vs Software solution

Before explaining the proposed solution, a brief comparison between hardware and software implementations [21] is necessary to be stated.

There are a lot of physical differences in hardware against software programming. This entails in a set of dual parameters.

- Main design paradigm: A hardware platform, like FPGA, has a limited quantity of logic resources for developing the complete design. The hardware developer has to decompose the solution in the space domain. In contrast, the software solution has the time as the constraint of the design. Software is composed by a series of rules executed by the processor one by one, and is limited by the duration of all rules.
- Main limitation: Hardware is limited by the time of the clock period, the operation has to be done between consecutive clock cycles. On the other hand, this limitation is represented by the instruction set in a software solution.
- Main behavior: Parallelism is the other main dual difference. In hardware, all blocks are executed in a concurrent way. We can make the design sequential, by waiting for the outputs of the previous stages. However the natural work-flow is parallelism. In software the behavior is just the opposite. Although The code is executed instruction by instruction, T the software can behave like a concurrent system. Because the system can have more than one processor buy this processor have to execute the instructions in a sequential form.
- Reuse: In software from open-source there are many functionalities which have been already developed and the programmer can use them in his/her own designs.

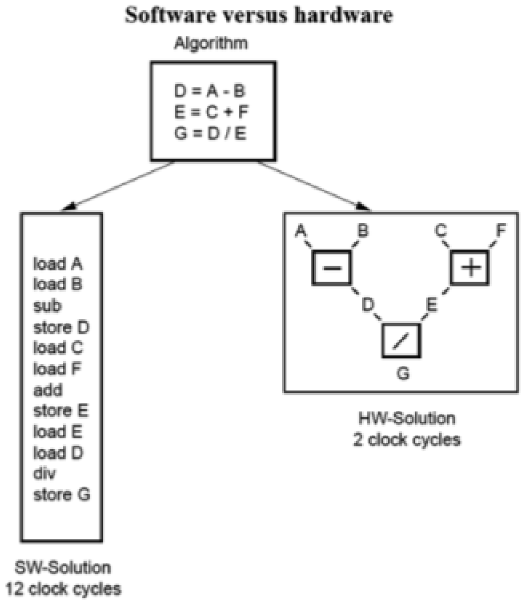


FIGURE 1. Software solution and hardware solution.

The intellectual property in the hardware environment is more suspicious and there are only few functionalities released.

A little example can be seen in Figure 1 that demonstrates some differences between software and hardware solutions.

There are three main distinctions between both designs that need to be emphasized:

(1) Data Access: As can be seen in the software solution of Figure 1, the majority of the instructions are about data access, seven of them for loading data and three for storing, thus around 80 % of the operations are used for data instructions. Software needs to write and then read every instruction processed, those being the most recurrent but also the longest operations, spending around 10 clock cycles for access to RAM memory in each instruction. On the other hand, hardware avoids the multiple data access, it allows connection to different blocks without memory interface, or to more suitable interface like FIFO.

(2) Mathematical Operations: Software needs much time to process a set of mathematical basic operations, referring to the fact that these instructions need to be done one by one[1]. If the number of processors is higher in the system, more operations can be performed, although in the same way, one by one. In contrast, hardware implementation has many little mathematical processors, called digital signal processors (DSP), included for doing this type of operation. Many multiplications and sums can be done within one iteration. Apart from this comparison, graphic processor unit(GPU) has become a hopeful option, because it

[1]In general Single Instruction Multiple Data(SIMD) is only possible with no decimal point data type and few bits for its representation.

contains a large amount of little processors, but it works necessarily with another type of processor.

(3) Parallelism: The processor executes the code instruction by instruction, not making possible to benefit of the parallelism of the real problem. The hardware solution is able to implement different blocks that process the data in a concurrent form. Moreover each block is composed of many structures that execute the data in a parallel way too, therefore it can better suit a problem with parallel nature. In this work, it had been chosen to utilize a heterogeneous device [22], which has a processor part and a hardware part to benefit from both paradigms. For this purpose the best option in our consideration was ZYNQ device from Xilinx, with two ARM Cortex A9 and a large FPGA part.

3. Proposed architecture

In Figure 2 our general architecture can be observed. This system works as follows: This system works as follows: the processor receives the weights from an external source[2], or in the future from the training calculation. It stores these values in the shared memory RAM. This is done by the AXI BUS, with the AXI RAM CONTROLLER. Next, the inputs from an external source are received by the processor and these are sent to the CNN hardware block with AXI BUS. Then the processor transmits the start signal with the same bus to the CNN hardware block. The CNN hardware block has a DONE signal for announcing the end of its work. When the processor notes this signal, it asks the CNN block for the outputs and the CNN responds with these.

For implementing the CNN we have chosen a fully hardware design with the HLS language. This is explained with details in the next section. The processor part and the interconnection result will be explained in the section 5. The results of the integration will be exposed in section 6.

4. Convolutional Neural Network block

Before starting explaining the hardware, one of the biggest issues in FPGA programming should be mentioned. The issue with the programming of FPGA has always concerned development time and difficulty. New paradigms of hardware programming have been consequently developed recently. Vivado High Level Synthesis is the language introduced by Xilinx, which provides programming in C++, C or SystemC, and focuses on the system level in contrast to the register level of the traditional languages like Verilog and VHDL. HLS is a set of technologies permitting the transformation of the code written in medium-level language to registers (called synthesis by Xilinx). The same functionality could be expressed through many different codes, which could be synthesized in several ways. Code styling and indication for synthesis are thus of the most important as each result of synthesis produces an implementation with a specific resource and performance. This study puts forward a code and a set of indications to achieve the best performance as well as a low usage of resources. Further information on the HLS can be found in [23].

[2]In a future implementation we will try obtain the weights (networks parameters) in the own processor by the training algorithm execution.

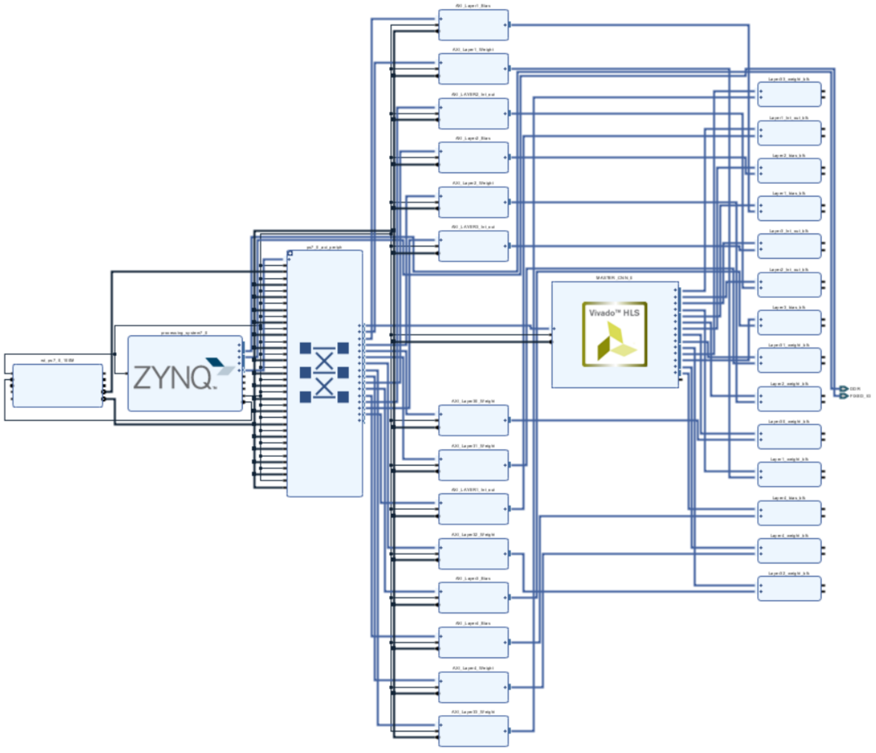


Figure 2. Proposed general architecture.

As concerns the CNN paradigm to be implemented into the FPGA, the concepts related to this type of deep neural network – such as convolutional filters, ReLu nonlinearity, max-pooling or softmax activation – will not be presented in detail in this work, but readers unfamiliar with the topic are invited to consult [24].

The architecture of the CNN model consists of a convolutional layer with a ReLu activation function, followed by one max-pooling function, connected to another convolutional layer, which has the same activation function and max-pooling. The output stage consists of two fully connected layers, the first with a ReLu activation function and the last layer with only 3 neurons and a softmax activation function.

The next step is searching for the best implementation for this architecture. We choose Keras API because it offers a friendly high-level language tool for both the description of the network and the training. Consequently, it allowed the study of different architectures with various numbers of layers and diverse types of training. However, we need to check the functionality implemented in HLS, and we therefore have to make a design layer by layer, with a simple operation. This fact is not possible under Keras. Because this fact is not possible with Keras, we consider MATLAB as the best solution and, every time we implement one layer, we also make a design in HLS. We have tried different coding styles and applied HLS directives for optimizing the design, and at every iteration of our methodology we have checked the implementation. The methodology used can be seen in Figure 3.

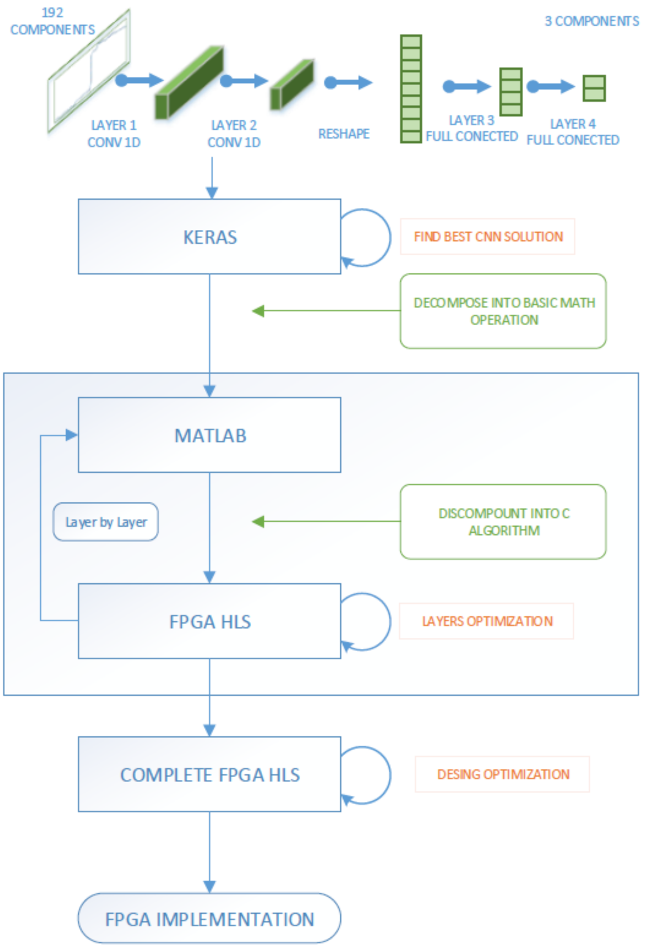


Figure 3. Methodology of Keras-Matlab-HLS interplay.

The implementation on hardware follows the structure showed in Figure 4. This architecture has been fully explained in the paper [18].

5. Processor and Interconnection

The processor part and the integration of all the components has been achieved in the following manner.

5.1. Processor. This work uses the processor part for configuring and controlling the CNN hardware block. In Figure 4 the processor appears like another hardware module but it is already integrated within the device. The processor block diagram can be seen in Figure 5. The main resources used in the design are emphasized: the

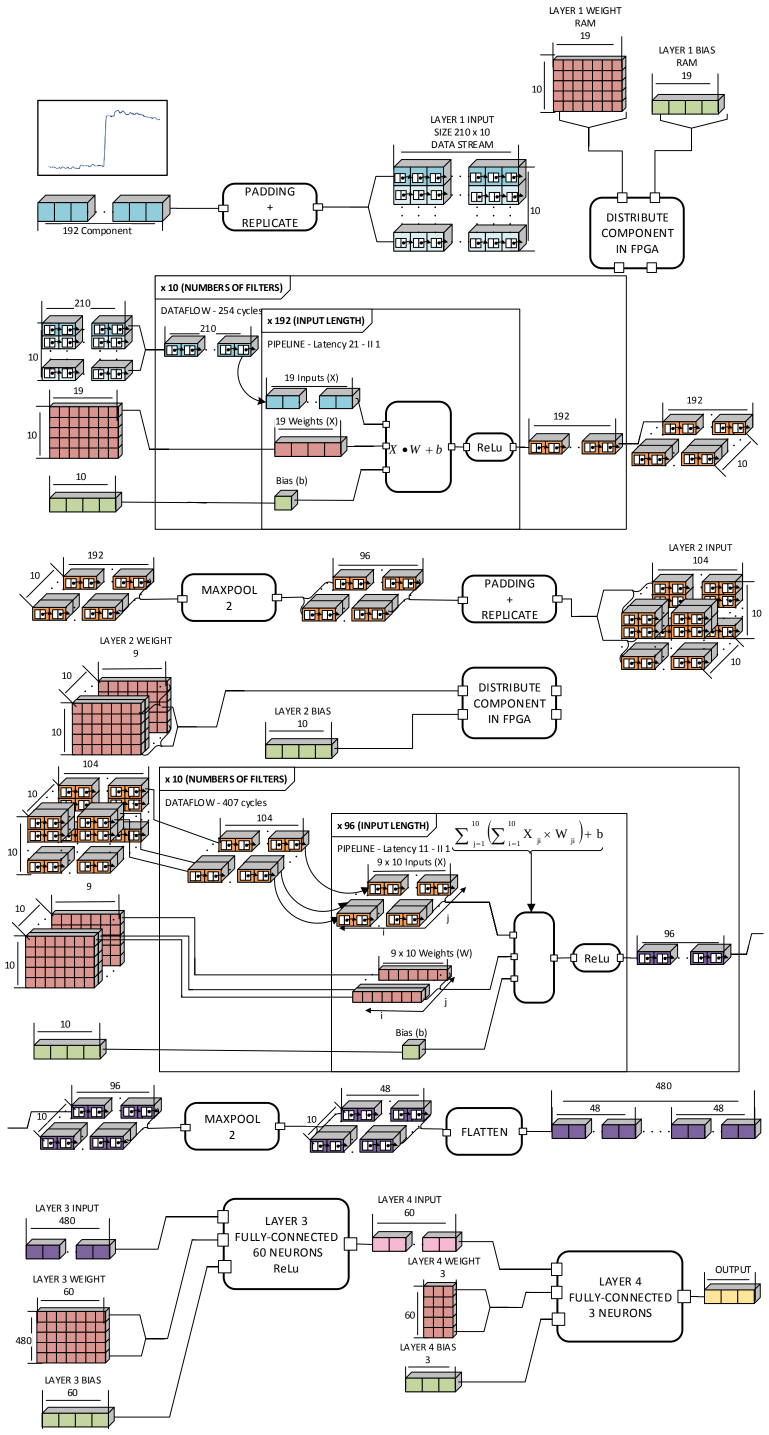


FIGURE 4. Complete flow of CNN.

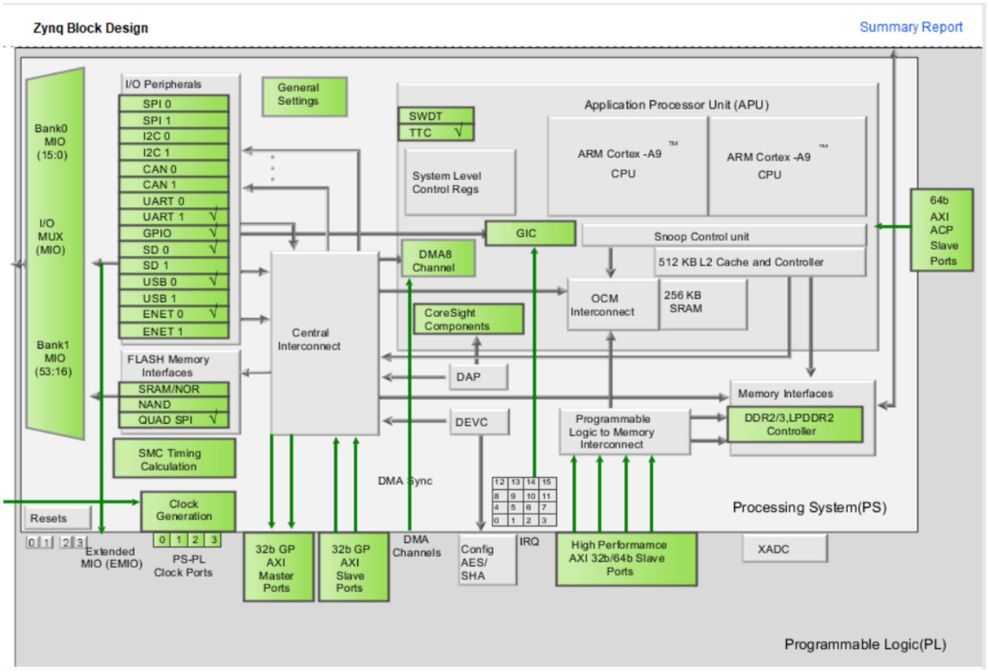


FIGURE 5. Processor block diagram.

two ARM CORTEX-A9 with their DDR3 Memory RAM, the AXI BUS CONTROL and the UART module for communicating with the external computer.

5.2. Interconnection. In this design we have considered to use individual dual port memory ram for the weight due to two main reasons. The first is to facilitate the parallelism of the CNN hardware block, and the second is for sharing the values between the processor part and hardware block. One port is for the CNN block and the other is for the processor. It was necessary to put another block for storing and reading the values from the processor to the memory block - this is AXI RAM CONTROLLER. For the communication between both the AXI RAM CONTROLLER and the processor, as well as between the CNN hardware block, and the processor, the design implemented the AXI BUS interface. It allows the control and configuration of the CNN block with a standard driver; also the communication with the CNN block includes sending the inputs and receiving the outputs.

5.3. Integration. After having completed the interconnection, it became necessary to integrate all parts, and transform the code into a hardware language. The IDE provides some reports for checking the functionality and for improving some parts. This will be shown in the section 6. The next step is to implement the design, it is the process in which Vivado transforms the hardware specification into a real map of resources; the results of this part will also be shown in the next section.

6. Results

The methodology used reported different results for each part with a different meaning in each one.

First, we have the results of the CNN hardware block implementation. The important findings in the FPGA design are the usage of the hardware resources and the running time result. Vivado HLS estimates the usage of the hardware and offers reports for each part of the implementation. The main report is the block utilization, as it can be seen in Table 1.

The usage of LUT, the most important resource, can be noted with a use of only 61%. This is a good mark because we are exploiting a large amount but This is a good mark because, although we are exploiting a large amount of them, we have a remains of 39% for other necessary actions such as putting the CNN design in connection with the processor part and the others.. The second important mark is the usage of DSP of 81%. This is a good result because it is very close to the total.

In order to compare the performance of the constructed implementation with that of a standard PC, two laptops usable in hospital environments have been considered. A comparison between the running times can be seen in Table 2. The last column gives the ratio between the reference computers and the CNN hardware block. The PC devices need between 8 to 12 times more time than the proposed hardware design.

The second finding refers to the integration reports. After the block diagram is completed, this is transformed to hardware language by the synthesis process. The results are provided in Figure 6. Referring to resource usage, the values are similar to the estimation before. The RAM occupation is total, it is a normal point with all the blocks already included and the DSP usage is 85%, a better point than before. For Power consume, the marks reported are in a good range. And lastly, for the time report, all values are in blue so all work as expected. Next step is the result of the implementation, and all marks received are satisfactory.

The final outcome of this work is referring to the connection with the processor part. The results obtained in terms of precision are distinct, because the data type impacts on the precision of the result but it has no effect in the classification. The values obtained from each software can be seen in the Table 3. These results are referring to the classification accuracy for the three classes. The prediction capability is not high at all, which means that the CNN architecture and learning have to be improved. However, the interest of the proposed study is to show the viability of an efficient implementation of a CNN framework on a FPGA.

7. Conclusions

A FPGA design and hardware implementation of a CNN have been put forward in this work.

The CNN architecture consists of two convolutional layers (with ReLu nonlinearity and max-pooling) and two fully connected layers.

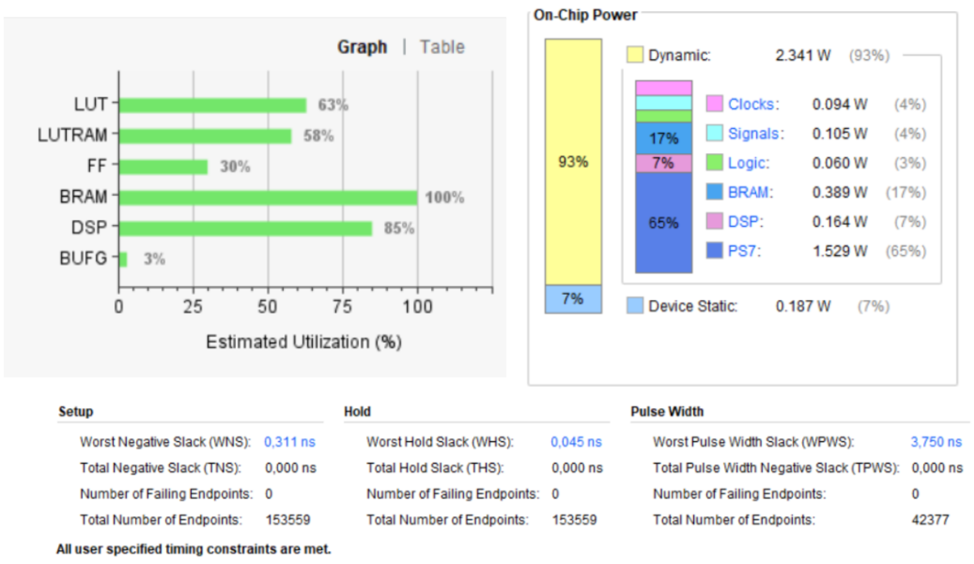
We integrated all parts into a hardware device, with all the necessary blocks. We have presented a general architecture, with a processor part for controlling and configuring the device. We have synthesized and implemented this design and configured the network through the processor.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	2108
FIFO	110	-	5950	12210
Instance	0	172	24886	17100
Memory	1	-	36	135
Multiplexer	-	-	-	1188
Register	-	-	138	-
Total	111	172	31010	32741
Available	280	220	106400	53200
Utilization (%)	39	78	29	61

TABLE 1. Resource usage estimates.

Ref	Processor	RAM	Time	$\frac{Computer_{time}}{FPGA_{time}}$
Computer 1	Intel Core i5-5257U	8 GB 1866MHz DDR3	937μs	11.86
Computer 2	Intel Core i7-8550U	16 GB 2400MHz DDR4	612μs	7.74
Hardware	-	-	79 μs	1

TABLE 2. Implementation in FPGA versus that in standard PCs. Comparison of running times.



Setup

Worst Negative Slack (WNS):	0,311 ns
Total Negative Slack (TNS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	153559

Hold

Worst Hold Slack (WHS):	0,045 ns
Total Hold Slack (THS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	153559

Pulse Width

Worst Pulse Width Slack (WPWS):	3,750 ns
Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	42377

All user specified timing constraints are met.

FIGURE 6. Integration reports.

	KERAS	MATLAB	HLS SIM	ZYNQ
Accuracy	65%	65%	65%	65%

TABLE 3. Accuracy in the classification in each part 1.

We have checked and compared the outcomes reported from the processor with the rest of implementations. All the results obtained are the same. The accuracy of the solution is not good enough for the given classification problem, however the aim of this study had been the achievement of the same implementation in hardware rather than in Keras and the verification of the results in terms of efficiency and competitiveness. Thus, we will be able to implement an improved architecture of a CNN when such one is obtained.

What is important in this work from the practical point of view is that the task to be solved is a real-world problem that needs a portable effective solution to be used in actual diagnosis within hospitals and clinics. A direction for future work envisages the optimization of the network architecture for achieving a more efficient classification. Since, in the current form, training had been performed within a traditional computer, a second future path will consist in analyzing the possibility of implementing both the training and the operation tasks within the ZYNQ-7000 device.

Acknowledgment

Carlos Cano would like to thank his teammate Roberto A. Becerra Garcia for his collaboration. The original version of the CNN architecture was proposed by him.

He also expresses his gratitude to the Faculty of Sciences of the University of Craiova for the grateful welcome and the facilitation of his work during his Erasmus stay.

References

[1] R.V. García-Bermúdez, F. Rojas Ruiz, J. González Peñalver, O. Valenzuela Cansino, L. Velázquez-Pérez, R. Becerra García, Evaluation of electro-oculography data for ataxia SCA-2 classification: A blind source separation approach, *Proceedings of the 2010 10th International Conference on Intelligent Systems Design and Applications, Design and Applications* (ISDA10) (2010), 237–241.

[2] F. Cabitza, G. Banfi, Machine learning in laboratory medicine: waiting for the flood? *Clinical Chemistry and Laboratory Medicine* **56** (2018), no. 4, 516–524.

[3] C. Stoean, R. Stoean, A. Sandita, C. Mesina, C.L. Gruia, D. Ciobanu, How Much and Where to Use Manual Guidance in the Computational Detection of Contours for Histopathological Images? *Soft Computing* (2018), https://doi.org/10.1007/s00500-018-3029-9.

[4] R. Stoean, C. Stoean, A. Sandita, D. Ciobanu, C. Mesina, Interpreting Decision Support from Multiple Classifiers for Predicting Length of Stay in Patients with Colorectal Carcinoma, *Neural Processing Letters* **46** (2017), no. 3, 811–827.

[5] Z. Obermeyer and E.J. Emanuel, Predicting the Future Big Data, Machine Learning, and Clinical Medicine, *The New England Journal of Medicine* **375** (2016), no. 13, 1216-1219.

[6] C. Stoean, R. Stoean, A. Sandita, D. Ciobanu, C. Mesina, C.L. Gruia, SVM-Based Cancer Grading from Histopathological Images using Morphological and Topological Features of Glands and Nuclei. *9th International KES Conference on Intelligent Interactive Multimedia: Systems and Services, Intelligent Interactive Multimedia Systems and Services* 2016, Vol. 55, *Smart Innovation, Systems and Technologies* (2016), 145–155.

[7] C. Stoean, In Search of the Optimal Set of Indicators when Classifying Histopathological Images, *IEEE Post-Proceedings of the 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, IEEE Press (2016), 449–455.

[8] R. Stoean, C. Stoean, A. Sandita, D. Ciobanu, C. Mesina, Ensemble of Classifiers for Length of Stay Prediction in Colorectal Cancer, *International Work-Conference on Artificial Neural Networks (IWANN 2015), Advances in Computational Intelligence, Lecture Notes in Computer Science*, Springer, **9094** (2015), 444-457.

[9] R.C. Deo, Machine Learning in Medicine, *Circulation* **132** (2015), no. 20, 1920-1930.

[10] C. Stoean, R. Stoean, Post-evolution of variable-length class prototypes to unlock decision making within support vector machines, *Applied Soft Computing* **25** (2014), Elsevier 159–173.

[11] F. Gorunescu, M. Gorunescu, E. El-Darzi, S. Gorunescu, A statistical framework for evaluating neural networks to predict recurrent events in breast cancer, *International Journal of General Systems* **39** (2010), no. 5, 471–488.

[12] C. Stoean, R. Stoean, Evolution of Cooperating Classification Rules with an Archiving Strategy to Underpin Collaboration, *Intelligent Systems and Technologies - Methods and Applications*, Springer (2009), 47–65.

[13] R. Stoean, C. Stoean, M. Preuss, D. Dumitrescu, Evolutionary Multi-class Support Vector Machines for Classification, *International Journal of Computers, Communications and Control*, Supplementary Issue (2006), 423–428.

[14] R.A. Becerra-García, R.V. García-Bermúdez, G. Joya-Caparrós, A. Fernández-Higuera, C. Velázquez-Rodríguez, M. Velázquez-Mariño, F.R. Cuevas-Beltrán, F. García-Lagos, R. Rodráguez-Labrada, Data mining process for identification of non-spontaneous saccadic movements in clinical electrooculography, *Neurocomputing* **250** (2017), 8–36.

[15] G. Litjens, T. Kooi, B. Ehteshami Bejnordi, A. Arindra Adiyoso Setio, F. Ciompi, M. Ghafoorian, J.A.W.M. van der Laak, B. van Ginneken, C.I. Snchez, A survey on deep learning in medical image analysis, *Medical Image Analysis* **42** (2017), 60–88.

[16] S. Postavaru, R. Stoean, C. Stoean, G. Joya Caparros, Adaptation of Deep Convolutional Neural Networks for Cancer Grading from Histopathological Images, *International Work-Conference on Artificial Neural Networks (IWANN 2017), Advances in Computational Intelligence*, Springer (2017), 38–49.

[17] R. Stoean, Analysis on the potential of an EA-surrogate modelling tandem for deep learning parametrization: an example for cancer classification from medical images, *Neural Computing and Applications*, https://doi.org/10.1007/s00521-018-3709-5 (2018).

[18] C. Cano-Domingo, An efficient FPGA implementation of a deep learning approach for the classification of saccadic movements in clinical electro-oculography, *International Conference on Applied Informatics (ICDD)* (2018).

[19] F. Ortega-Zamorano, J.M. Jerez, I. Gómez, L. Franco, Layer multiplexing FPGA implementation for deep back-propagation learning, *Integrated Computer-Aided Engineering* **24** (2017), no. 2, 171–185.

[20] E. Nurvitadhi, S. Subhaschandra, G. Boudoukh, G. Venkatesh, J. Sim, D. Marr, D. Moss, Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA* (2017), 5–14.

[21] P.R. Schaumont, *A Practical Introduction to Hardware/Software Codesign*, Springer Science Business Media (2012).

[22] Y. Li, X. Zhao, T. Cheng, Heterogeneous Computing Platform Based on CPU+FPGA and Working Modes, *12th International Conference on Computational Intelligence and Security (CIS)* (2016), 669–672.

[23] Xilinx, Vivado Design Suite User Guide High-Level Synthesis UG902(v2018.1) (2018).

[24] M. Nielsen, Neural Networks and Deep Learning, Available at `www.neuralnetworksanddeeplearning.com` (2017).

(Carlos Cano, Gonzalo Joya) Departamento Tecnología Electrónica, E.T.S.I. Telecomunicación, Universidad de Málaga, Campus de Teatinos, 29071 Málaga, Spain
E-mail address: `ccd@uma.es,gjoya@uma.es`

(Ruxandra Stoean) Department of Computer Science, Faculty of Sciences, University of Craiova, 13 A.I. Cuza Street, Craiova, 200585, Romania
E-mail address: `rstoean@inf.ucv.ro`