FPGA-based Hardware Acceleration for SVM Machine Learning Algorithm

Jakjoud Fatimazahra[1*], *Hatim Anas*[2], and Abella Bouaaddi[1]

[1]Laboratory of Energy Engineering, Materials and Systems, National School of Applied Sciences, Ibn Zohr University, Agadir, Morocco
[2] National School of Applied Sciences, Cady Ayad University, Marrakech, Morocco

Abstract. object recognition algorithms are both large consumers of computing power and memory which affects the quality and performance especially when it comes to large image datasets, in this paper we propose an algorithm for fruit/plant recognition that we will accelerate it using the PYNQ Board to evaluate the execution time and the accuracy of the classifier.

1 Introduction

Object recognition algorithms require a high-performance hardware architecture to ensure proper functioning. Due to the rapid development of image processing techniques, the algorithms complexity and image resolution lead us to choose a high end up in order to perform image processing tasks. For this aim we find that the image processing tasks are always executed by specific processors like DSPs. More the complexity of functions increases we have to increase further the number of parallel units that become very expensive.
Hardware acceleration is the adapted solution to improve performance by using hardware architectures dedicated to parallel processing. Field Programmable Gate Array (FPGA) offer a hardware architecture to implement with lower costs. the latest version of FPGAs include configurations of general purpose logic resources (LUTs) and registers with specialized modules such as slice, memory and multipliers DSPs.

In this paper, we present two object recognition approaches dedicated to plants and crops supervision, which will allow recognizing the fruits out of their leaves. That system is in great demand when it comes to a system for spooning fruits or robots which supervise the growth of fruits and vegetables. Both approaches are based on the SVM classifier. In the first one, we execute the entire model on the Processing System part of the Zynq, while we use the processing logic of Zynq for accelerating the model in the second approach.

2 Related work

In this section, we provide an overview of the related works that regards machine learning implemented on embedded systems and hardware acceleration methods. Rong Xie et al [1] present a method of quick edge detection based on Zynq with the use of XfOpencv library. Accelerating edge detection process is done due to parallel computing capabilities of FPGA to increase process speed and reduce time-consuming, the accelerated algorithm with XfOpencv has some advantage in computational time compared to other environment. More recently Tzanos et al. [2] exhibit a hardware acceleration based on Naïve Bayes to exceed embedded ARM processors, the proposed system can reduce the execution time up to 16,8x times while training part and up to 14x times in prediction phase. Training implementation is based on three points, first one is to store many data in local BRAM to have direct access from the fpga to reduce transferring data latency, second point is to avoid memory access bottlenecks by partitioning large arrays into multiple smaller individual arrays, the last point is to pipelining loops which assured the maximum parallelism. In [3] was proposed a VLSI architecture of Naïve Bayes classifier on FPGA for real time classification of facial expressions, this approach can perform real time classification operating at a frequency of 241,55 MHz. [4] A framework to facilitate implementation of deep learning algorithms using PYNQ platform is proposed [4], this solution will help data scientist and hardware developers to combine the use of Deep learning model with architecture FPGA based.

Other works are developed with the aim of reducing complexity. Indeed, PYNQ platform is used to increase accuracy performance of binary weights with respect to allow power consumption [5]. Zhang et al. [6] developed binarized neural networks framework to reduce FPGA hardware resources complexity.

* Corresponding author: jfatimzehra@gmail.com

3 Proposed method

Our method is based on a set of pre-processing operations before classification phase. The size of the database is 5000 images split into two classes (2500 images contain fruits and 2500 images of tree leaves) in order to normalize image size. Before starting feature extraction phase, we resized the images to have a standard size of 150x150x3 then, using Histogram of Oriented Gradient HOG, we extract the images' features to be classified; the last step consists in training SVMs to have a convergent classification model. Our hypothesis concerning the acceleration of our algorithm is to develop an overlay dedicated to the part of the resizing of images and to evaluate the execution time and the performances of the classifier with and without acceleration.

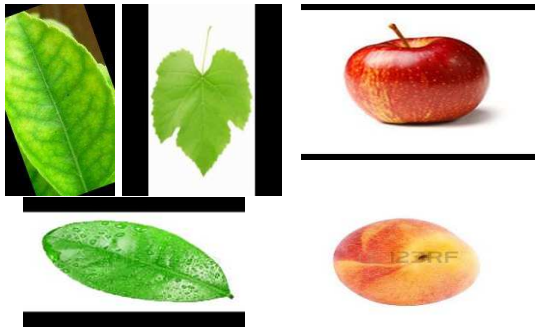

Fig.1. Dataset image examples

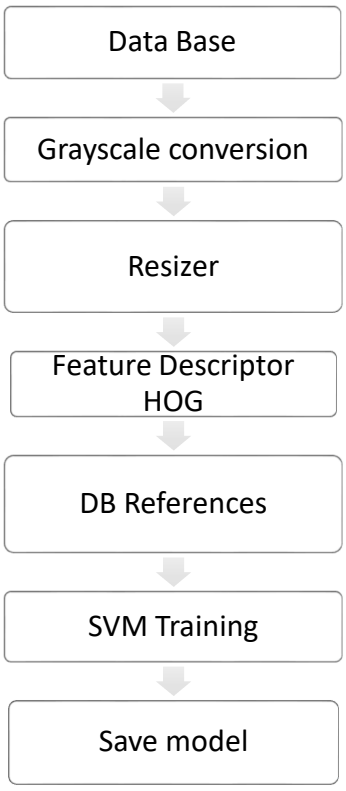

Fig.2. Training SVM model

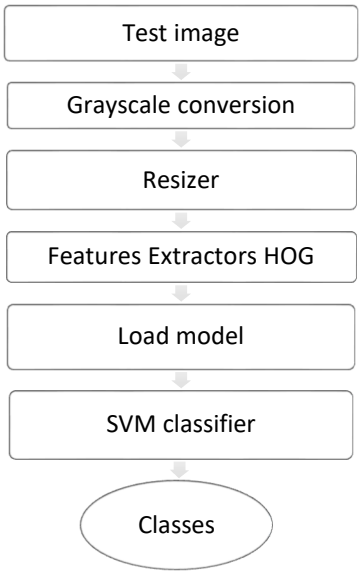

Fig.3. Classification SVM model

3.1 Feature Description

Feature description [7] is a representation of image by extracting important information. The extracted feature by HOG are computed by describing the local distribution of the edge orientations and the corresponding gradient magnitude, this is realized by defining two locally units cell (8x8 pixels) and block that contains 2x2 cells, this gives 16x16 pixels for each hog feature [8]. At each pixel located at (x,y) an orientation of local image gradient is computed after determining the magnitude $\rho(x,y)$ and direction $\gamma(x,y)$

$$\rho(x,y) = \sqrt{L_x(x,y)^2 + L_y(x,y)^2} \quad (1)$$

$$\gamma(x,y) = arctg\left(\frac{L_y(x,y)}{L_x(x,y)}\right) \quad (2)$$

Which L_x, L_y are first order Gaussian derivatives of the image.
For each pixel in the orientation image, a histogram of orientations is constructed over a cell. To generate the feature descriptor vector, all adjacent cells are grouped into normalized block histogram, these blocks will be concatenated to form a descriptor.

3.2 Support vector machine

SVM is a successful classifier in supervised learning. It shows high performance in object recognition application, the main objective of the SVM [9] is to find the optimal hyperplane to maximize the separation of two class in case of binary classification. For each side of the main hyperplane, two other parallel hyperplanes are constructed and the algorithms try to find the best separating hyperplane which maximizes the distance between secondary hyperplanes.

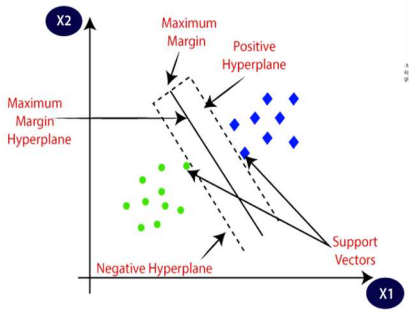

Fig 4. SVM optimal Hyperplane

3.3 Image resizer

we have leaned towards the normalization size of data, since the database images are large. To minimize the execution time, we decided to reduce the size of the images. Given the size of the database (5000 images), That will take more time to resize them to 150x150x3. To solve this problem, we have dedicated the execution of this part to the Programmable logic of the PYNQ.

Python Productivity for Xilinx Zynq (PYNQ) is an open source project developed to design embedded systems for ZYNQ 7020 System on chip (SoC) [5]. PYNQ Z1 is an FPGA Board equipped with the Pynq framework and in Zynq7020, an FPGA and a processor mounted on the same chip. Specifically, the processing system PS consist of dual core ARM cortex A9 CPU with build-in Linux Ubuntu operating system and the programmable logic PL (FPGA) unit is mounted with an Artix 7 family which contains 13300 logic slices, 630 KB of fast block RAM and 220 DSP slices, 512 MB DDR3 with 16-bit bus.

Pynq is adaptable for hardware acceleration [4], it includes Python drivers that execute an application-programming interface (API) for FPGA bitstream download and data transmission. With the use of Vivado the programmable logic circuits are presented as hardware libraries called overlays.

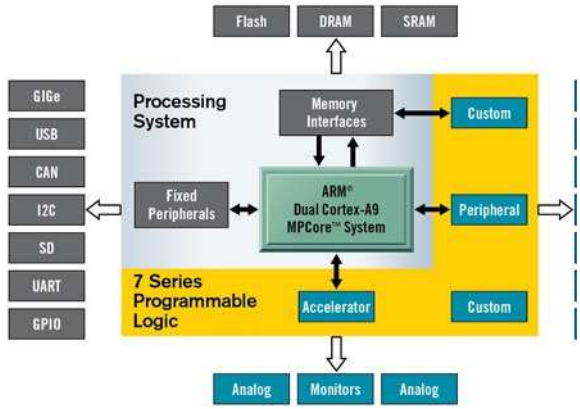

Fig.5. ZYNQ7020 Architecture

The hardware architecture is designed to send data to the IP of resizing via the DMA and reading back data all the IPs are developed by XfOpenCV which contains a set of kernels and image processing primitives that replicate the functionality of OpenCV library.

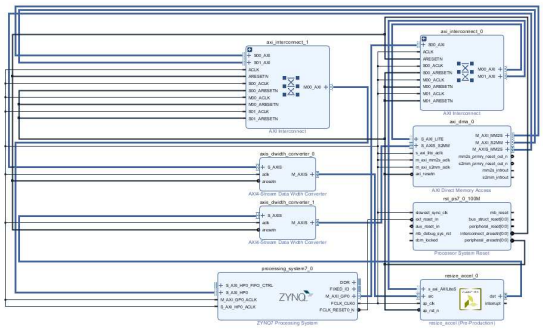

Fig.6. Block Diagram in Vivado

4 Result and Future Directions

After bitstream generation and overlay, we prepared the PYNQ software environment by configuring all modules that we are going to use while running the program. The SVM training is done on a host machine which lasted 25 hours in order to achieve convergence then the model is tested on ZYNQ. the table below shows the execution results according to two experiment. First, we have run the whole algorithm on the ARM which gave an accuracy of 97,5% in 5.4 seconds then we accelerated resizing stage and the result goes up to 97,5% in 4.8 seconds.

Table 1. Classification result

Classifier	Accuracy	Time (s)
Hog SVM Classifier	97,5%	5.4
Accelerated Hog SVM classifier	97,5%	4.8

Our purpose was to find a solution to accelerate the execution of object recognition algorithms without losing the efficiency of the classifier, this solution is only a first step for the integral acceleration of the processing loop and the fact of using Pynq board is very essential and promising. With the simplicity of use we do not need to reprogram all the interactions interfaces but it is enough to design the hardware architecture which is adaptable with the solution general functionalities.

5 Conclusion

in this paper we propose an approach for the acceleration of pre-processing stage for object recognition algorithm on PYNQ board. This solution will help us to generate the required interfaces to interact with FPGA based implementation of object recognitions algorithms. The tool has been abstracted to a level at which someone with the minimum knowledge in embedded electronics can develop a working model. The hardware IP cores are developed with Vivado HLS 2019 and the SVM Classifier is programmed by the python language. both approaches gave an accuracy of

97.5% but the execution time of the accelerate classifier was reduced by 11.1% (0.6 seconds). in the future work we intend to accelerate feature descriptor and classification model.

References

1. Rong Xie, Xiaoqin Feng, A method of quick edge detection based on Zynq, 3rd international conference on cloud computing and internet of thing, IEEE, (2018)
2. Georgios Tzanos, Christoforos Kachris, Dimitrios Soudris, Hardware Acceleration on Gaussian Naive Bayes machine learning algorithm, international on modern circuits and systems technologies, (2019)
3. P.Chaudhary and M.K.Sharma, VLSI Hardware Architecture of Real time Pattern Classification using Nave Bayed Classifier, in Proceedings of the 2017 2nd International Conference on Multimedia Systems and Signal Processing- ICMSSP 2017
4. Luca Stornaiuolo, Marco D, Donatella Sciuto, On how to efficiently implement Deep Learning algorithms on PYNQ platform, Computer Society annual symposium on VLSI, IEEE, 2018
5. "Bnn-pynq," https://github.com/Xilinx/BNN-PYNQ/.
6. C,Zhang, P.Li, G.Sun, Y.Guan, B.Xiao, J.Cong, « Optimizing fpga-based accelerator design for deeo convolutional neural networks », in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, ACM, 2015, pp.161-170
7. Wei Zhou, Ling Zhang, Xin Lou, Histogram of Oriented Gradients Feature Extraction From Raw Bayer Pattern Images, IEEE Transactions on circuits and systems – II: Express Briefs, VOL. 67, No 5, May 2020
8. Harihara Santosh Dadi, Gopala Krishna Mohan Pillutla, Improved Face Recognition Rate Using HOG Features and SVM Classifier, Journal of Electronics and Communication Engineering, ISSN:2278-8735, Volume 11, Issue 4, Ver. I, Jul-Aug 2016, pp 34-44
9. Samy Bakheet, An SVM Framework for Malignant Melanoma Detection Based on Optimized HOG Features, Computation 2017,5,4