Embedded Image Analysis System Based on B-ANN

Laurentiu-Mihai Ionescu[1)], Alin Gheorghita Mazare[1)], Daniel Visan[1)], Adrian Ioan Lita[2)], Ioan Lita[1)], Gheorghe Serban[1)]

[1)] Department of Electronics, Computer and Electrical Engineering, University of Pitesti, Faculty of Electronics, Computers and Communications, Pitesti/Romania
[2)] Axiplus Engineering SRL., Bucharest/Romania
laurentiu.ionescu@upit.ro, ioan.lita@upit.ro

***Abstract**: An increasingly common requirement is related to image analysis and automatic extraction of events "in situ" - on the place where image is acquired - without the need to send the image remotely for analysis. There are integrated systems in the video cameras that allow the analysis of images especially in applications for perimeter monitoring and there is research done for different analysis systems that can be integrated near the camera. However, these existing solutions either do not allow high levels of analysis or, if they allow, involve the use of expensive and difficult to implement technologies. In this paper we present a solution, based on B-ANN that allows image analysis, classification and extraction of events of different types. The proposed solution can be implemented on a FPGA circuit for commercial use and allows, through the digital and analog interfaces, the connection to different types of cameras.*

1. INTRODUCTION

Image analysis, object extraction and classification are becoming a feature increasingly present in many applications. Thus, video cameras become complex sensors that extract information of interest from images. There are 2 methods by which the camera-image analysis device system can be implemented. Most commonly, video cameras transmit video streams to analysis devices: DVRs with integrated analysis modules or servers with VMS (Video Management Software) that has integrated intelligent image analysis modules. There are commercial solutions and research still being done to increase the efficiency of existing analysis algorithms or to develop new video analysis algorithms. Dedicated manufacturers of video cameras for monitoring and DVR or NVR recording solutions have allocated resources also for the development of video management softwares (VMS) running on servers and containing video analytics modules aimed mainly at surveillance (person recognition, recognition incident, LPR etc) [1]. There are also companies specialized in the development of VMS software solutions that have developed important segments for the inclusion in VMS of video analytics modules in multiple fields: surveillance, perimeter protection, security, fire prevention [2]. Most solutions are server-centric, which is why research is being done to scale them to video systems that contain a large number of cameras - a single server could not cope with the many video streams it receives [3]. Research has advanced in this direction, including proposing artificial intelligence solutions (neural networks) for planning the arrangement of VMS server nodes in a major video monitoring network [4]. Other research directions are taking place in order to improve the detection methods of objects on images [5]. This first method involves transmitting the video stream between the camera and the image analysis device. Currently it is most often done via ethernet (rtsp and http protocols) using WAN through various transmission media including mobile data networks.

A second approach is for the video analysis to be done at the camera level, by integrating the analysis device into the camera. The solution decentralizes the image analysis, moving it to the image capture equipment. Thus, the communication channel to the central unit is released, only events associated with image captures or video sequences are transmitted. There are commercial solutions and studies that present this method and its advantages. Existing commercial solutions such as the one presented in [6] allow the

delimitation in an image of regions that are then monitored for motion detection. These solutions mainly target an area of image analysis: perimeter monitoring. For other areas the solutions are very rare, expensive or completely missing. There is research that has been done in these fields and that has focused in particular on the implementation of intelligent recognition algorithms on integrated PC systems [7]. Their recognition performance is significant but implementation on modern integrated systems (for example Raspberry PI) still results in quite large response times. An alternative solution is to use FPGA hardware structures for "hardening" neural networks [8]. In these, the response time is significantly lower. The problem that arises here is the complexity of the system which makes it difficult to implement on low cost hardware structures - as they are used in commercial applications.

This paper presents a solution for image analysis and classification using an artificial binary neural network (B-ANN). The solution can be integrated in this way in a System on Chip Zynq 7000, composed of an ARM processor and an FPGA type Artix 7 - B-ANN is fully integrated in the FPGA while the processor provides the "serving" part. ” of B-ANN with training templates and test inputs. This kind of network is used, as presented in the paper, to identify and classify of the images. The goal is to perform training and testing at the level of Zynq 7000, so at the level of an integrated equipment that can be attached to a video camera. The effect is to perform a real-time analysis of the image and its classification, but also to learn new images - a process that takes place is also real-time. As output from the SoC Zynq 7000 integrated with a camera we will have only the events generated as a result of the analysis and classification of images that can be transmitted as event messages of very low capacity, so it does not require transmission channels with high bandwidths. Nowadays, when 5G seems quite close to implementation, this advantage seems insignificant. In reality, things are different. First of all, there is a type of application that takes place in areas where 3G/4G or future 5G coverage is lower: here we mention applications in agriculture. Secondly, Smart City applications, even if they take place in covered areas, need many camera-type devices to be managed - all of which transmit video streams that need to be analyzed so they occupy the communication channel. Third, there is a positive effect of camera image analysis for Smart City applications: protection of personal data. Instead of sending video streams to a server where they can be stored and eventually then accessed and used fraudulently, only the events of interest will be delivered. So, the proposed solution is useful in several areas. In the paper we will present the implementation and use of the solution in agriculture to identify, through video analysis, the presence of pests in an agricultural crop. The next section presents the structure of the binary neural network. Section 3 presents the structure of the system in which B-ANN was implemented and section 4 presents the results obtained.

2. The Model and Its Implementation

The neural network model used is the binary one – Binary Artificial Neural Network B-ANN. It is more efficient to implement on FPGA. The model uses only binary values 0 and 1 for inputs, outputs and weights. Thus, the inputs will represent pixels for a monochrome image (0 black and 1 white). The outputs will represent the classes of objects that are identified (0 is not identified, 1 is identified). The value of a weight can be 0 - inactive, 1 - active. These approximations reduce the degree of diversity required for convergence. To correct this, random sequence generators were introduced that participate in the learning stage, as will be described below.

A block diagram of the B-ANN network is shown in figure 1. The main components can be observed: neurons and weights.

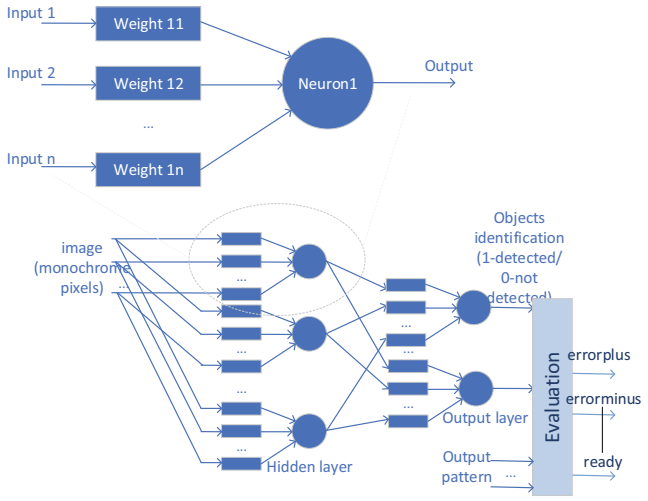

Fig. 1. B-ANN structure: top a single neuron with weights modules, bottom 2 layer network with evaluation circuit.

An important feature of B-ANN is that weight is not just a value but a functional module. The weight tracks the input (which is a value from the output of a neuron in the previous layer) and determines the output (input to a neuron in the next layer) based on a relationship between the input and a random update coefficient - to preserve the diversity of the weights in the network. A perceptron neuron is a simple repeater. The structure of a neuron in the hidden layer or the output layer is shown in figure 2.

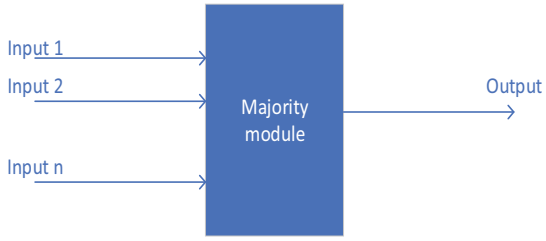

Fig. 2. The structure of neuron (hidden and output layer).

It can be seen that the neuron is reduced to a majority binary circuit. This implements the sum of inputs and the activation function. Figure 3 shows a functional diagram of the weight module. The way it was built is important for convergence.

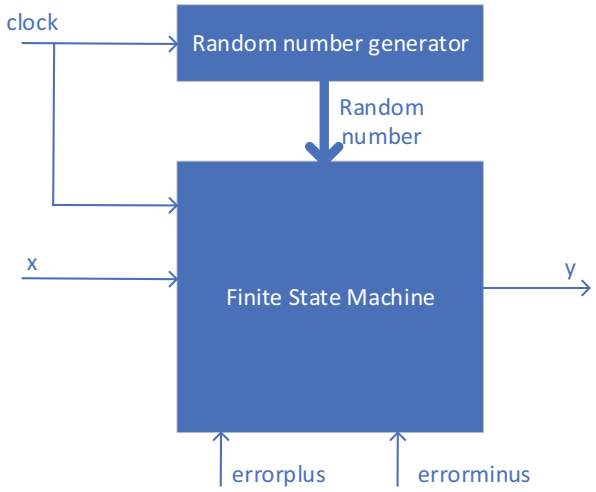

Fig. 3. The structure of the weight module.

It is made of a 4-state FSM (Finite State Machine). Two of these will lead to the output of 1 and two to 0. The output of the automatic reaches the input of the neuron. One state is given by the value of the input, one by the value of the negated input, one is 1 logical and the other 0 logical. So, two of the states of the module will determine the output 1 logically and the second 0 logically. The transition from 1 to 0 and vice versa is given by the input, error and the 4-bit value returned by the pseudo-random sequence generator. Each weight module has a pseudo-random sequence generator.

Another important module is the one responsible for making the comparison during learning (evaluation module). It transmits 2-bit error to weights - it can be errorPlus and errorMinus. ErrorPlus in 1 means that the output is higher (value compared to the decimal) than expected so a decrease in the values by weight is required while errorMinus in 1 means that the output obtained is lower than expected. Zero on both means that the output obtained is the expected one.

3. THE SYSTEM

The system block diagram is shown in figure 4. B-ANN is implemented on the FPGA module of the Zynq7000 circuit (Zybo model from Digilent) while the interface and learning commands is done by software running on the ARM processor.

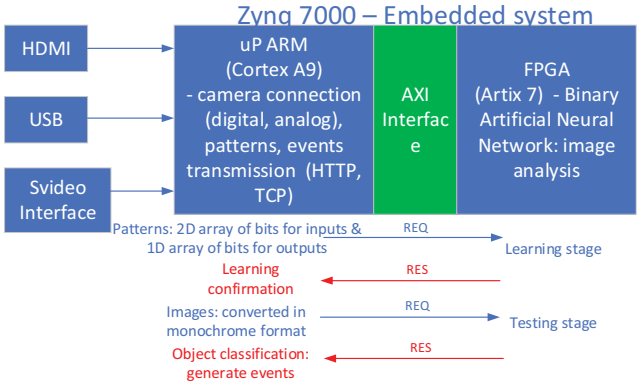

Fig. 4. System block diagram.

ARM runs a Xilinux operating system (it is an Ubuntu compilation) and at its level the software application was made in C (using the native Linux GCC compiler). The access to the FPGA and therefore to the B-ANN module is made through the Xillybus interface. At the processor and operating system level Xillybus is installed as a driver and visible as an accessible file for writing and reading. At the FPGA level the interface is perceived as a random-access memory, each location corresponds to a pattern input for training or test input. The address of the interface "memory" as well as the implementation technologies used are shown in figure 5.

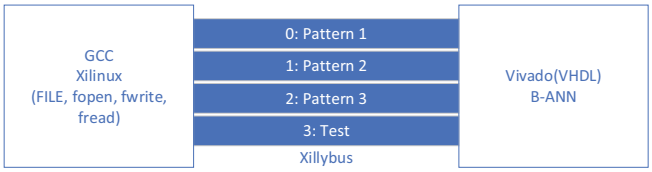

Fig. 5. Xillybus configuration and technologies used.

4. RESULTS

The solution was tested in a pest recognition application using a pheromone trap. The block diagram of the test system and captures (images) with the tested system is shown in Figure 6.

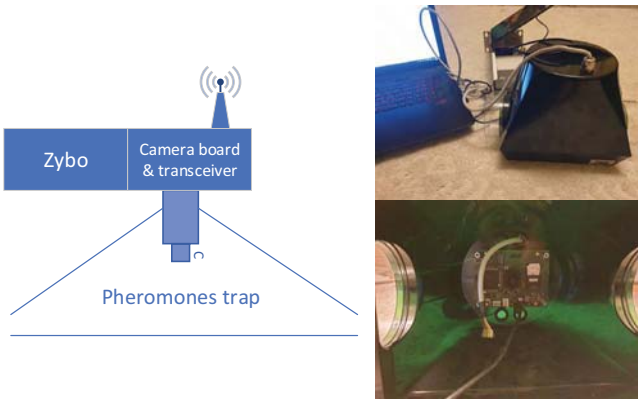

Fig. 6. Block diagram of the testing system (left) and pictures with the system (right): top – operating system connected to computer, bottom - a look inside the pheromone trap, video camera and camera board can be seen in the center.

You can see the Zybo (Zynq 7000 board) connected to the camera, which in turn is mounted on a system with a pheromone trap. The camera also has a set of LEDs that are operated when the image is captured.

The captured image is then converted to monochrome by a software module running on Xilinux. It is further transmitted as a pixel array (binary array) to the B-ANN in FPGA. We have two ways to use the system. The first way is for data collection and learning. In this case, next to the image we have a value of the expected outputs. In our case, we have to recognize three types of images that represent the insect in different stages of development (see figure 7).

The system was implemented on Zynq 7000 xc7z010clg400-1. During the implementation, different images sizes were tested. The input layer is set according to the image size. The size of the hidden layer was chosen to 10 neurons and the output layer has dimension 3 (how many objects must be classified).

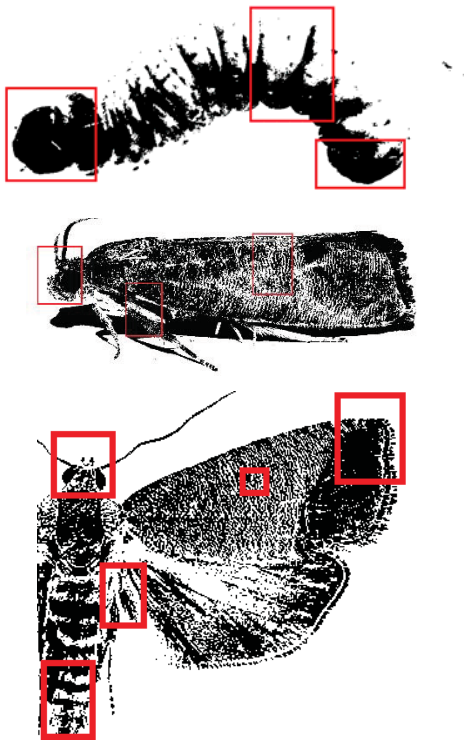

Fig. 7. 3 different patterns used to learning B-ANN. In fact, segments of the images were used in training to reduce the input layer size

The implementation report obtained with the Vivado Web Pack 2016.3 tool is presented in the table below.

Table 1. Results from implementation on Zynq 7000 xc7z010clg400-1.

Image segment (pixels)	Resource	Utilization
4x3	LUT	607 (3.45 %)
	FF	1300 (3.69 %)
	IO	20 (20 %)
	BUFG	1 (3.13 %)
7x5	LUT	2428 (13.8 %)
	LUTRAM	380 (6.33 %)
	FF	2660 (7.56 %)
	IO	45 (45 %)
	BUFG	1 (3.13 %)
9X9	LUT	4750 (26.99 %)
	LUTRAM	840 (14.00 %)
	FF	5880 (16.7 %)
	IO	91 (91 %)
	BUFG	1 (3.13 %)

Three implementations were performed for image segments to be analyzed of different sizes. One of the main limitations of the FPGA integrated in the Zynq7000 SoC is related to the number of pins that can be used: in the case of xc7z010 it is 100. In an image segment with a size of 81 pixels you can see an increase to 91 % of allocated I/O resource (81 are the inputs and 10 are the other pins). However, this is not an inconvenience – the tests were performed for input segments that must be analyzed loaded in parallel. In our case, the loading will be done through the Xillybus interface - which allows byte-by-byte writing - so parallel loading is no longer used. Significant are the other resources which, as can be seen, do not exceed 20 % of the capacity of the circuit.

We used a B-ANN with a variable number of neurons on the input, 10 neurons in the hidden layer and 3 neurons on the output (as we have shown, we learn 3 images).

To see the performance of the network we made a comparison at the learning time with other implementations with dedicated DL-ANN (made using Keras and Tensor Flow) having the same parameters (input layer, hidden layer, output layer).

Table 2. Learning time comparation.

ANN type	Learning stage time (ms)	Testing stage time (ms)
DL ANN (Tensor Flow, Keras) – PC run, GPU	6000 – 20000 (20 epochs × 120 ms – 1000 ms/ epoch)	< 1000
DL ANN (Tensor Flow, Keras) – Embedded PC (RPI)	60000 – 100000 (20 epoch × 3000 ms – 5000 ms/ epoch)	2000 – 3000
B ANN – FPGA	20 – 100 (epoch duration between 1 − 5 ms!)	<< 1000

For testing the response time to all three implementations is very short. However, we emphasize that the implemented B-ANN solution has a response time given by the propagation through the binary network with the values presented in the table below.

Table 3. B-ANN response time (from Vivado Implementation timing tool).

B-ANN Configuration (input × hidden × output)	Response time
12×10×3	~ 10 ns
35×10×3	~ 40 ns
81×10×3	~ 40 ns

The same response time can be observed for 35 and 81 neurons in input layer. The explanation is in the parallelism of the network. However, in the 12-neurons variant, the value was lower - as the number of inputs to the neuron increases, so does its horizontal complexity.

5. CONCLUSIONS

The presented solution implements a B-ANN artificial binary neural network on a modern SoC composed of FPGA class Artix 7 and an ARM Cortex A9 processor. Although, in general, the level of performance does not equal a DL-ANN implemented by "conventional" methods, it is still applied successfully and with superior performance in certain types of image recognition applications. As we have shown, by implementing B-ANN and integrating it into a SoC next to the video camera, it is allowed to detect and classify objects by analyzing images on the spot, with only the transmission of events of interest. Moreover, it is possible to implement a real-time learning algorithm of new templates also at the SoC level. In this way the central server is relieved of the tasks necessary to implement intelligent image analysis and the communication channels are freed from the transmission of video streams.

The solution was tested for learning 3 objects from an image. It is possible to extend the number of objects as well as increase the size of the analyzed images, thus extending the applications area. The ultimate goal is to create an "image sensor" capable, through intelligent analysis, to detect objects in an image and generate events related to them.

ACKNOWLEDGMENT

The research that led to the results shown here has received funding from the project “Increasing the institutional capacity of bioeconomic research for the innovative exploitation of the indigenous vegetal resources in order to obtain horticultural products with

high added value (BIOHORTINOV)", ID: PN-III-P1-1.2-PCCDI-2017-0332/P2, UEFISCDI.

REFERENCES

[1] https://www.securityinformed.com/cctv-software/make.mk-1264-ga.html

[2] https://trassir.com/products/intellektualnyie-moduli/

[3] S Jain, G Ananthanarayanan, J Jiang, Y Shu, "Scaling video analytics systems to large camera deployments", *HotMobile '19 Proc.*, Vol.1, February 2019, pp. 9–14

[4] J Jiang, G Ananthanarayanan, P Bodik, S Sen, "Chameleon: scalable adaptation of video analytics", *SIGCOMM '18 Proc.*, Vol.1, August 2018, pp. 253-266

[5] G Ananthanarayanan, P Bahl, P Bodík, "Real-time video analytics: The killer app for edge computing", *Computer*,Volume: 50, Issue 10, October 2017, pp. 58−67

[6] https://www.mobotix.com/en/products/access-control/people-counting-directions-of-movement

[7] Augusto Vega, Chung-Ching Lin, Karthik Swaminathan, Alper Buyuktosunoglu, Sharathchandra Pankanti, "Resilient, UAV-embedded real-time computing", *33rd IEEE International Conference on Computer Design Proc*., Vol.1, October 2015, pp. 1–4

[8] Shang Wang, Chen Zhang , Yuanchao Shu, Yunxin Liu, "Live Video Analytics with FPGA-based Smart Cameras", *Workshop on Hot Topics in Video Analytics and Intelligent Proc*., Vol.1, October 2019, pp. 9–14