DOI: 10.21917/ijivp.2014.0122

FPGA IMPLEMENTATION OF ADAPTIVE INTEGRATED SPIKING NEURAL NETWORK FOR EFFICIENT IMAGE RECOGNITION SYSTEM

T. Pasupathi[1], A. Arockia Bazil Raj[2] and J. Arputhavijayaselvi[3]
Department of Research and Development, Kings College of Engineering, India
E-mail: [1]pasu.tamil@gmail.com, [2]brazilraj.a@gmail.com, [3]dean@kingsindia.net

Abstract
Image recognition is a technology which can be used in various applications such as medical image recognition systems, security, defense video tracking, and factory automation. In this paper we present a novel pipelined architecture of an adaptive integrated Artificial Neural Network for image recognition. In our proposed work we have combined the feature of spiking neuron concept with ANN to achieve the efficient architecture for image recognition. The set of training images are trained by ANN and target output has been identified. Real time videos are captured and then converted into frames for testing purpose and the image were recognized. The machine can operate at up to 40 frames/sec using images acquired from the camera. The system has been implemented on XC3S400 SPARTAN-3 Field Programmable Gate Arrays.

Keywords:
Image Recognition, Spiking Neuron, FPGA, Artificial Neural Networks, Feature Extraction

1. INTRODUCTION

By automatic identification of real time users, personalized services such as a face recognition-based smart TV program, Passport security alert system, Bank password, Lap password systems can offer a set of programs that are customized to user profile and it will be identified and matched with previous database [1]. In traditional fingerprint and iris identification, users will pass through a 'pause and declare' procedure for authentication purpose. This may not be suitable for consumer application. On the other hand face Recognition does not have that type of short coming. Here, we have developed a face recognition system that can be very helpful in a real time environment to facilitate intelligent services. Comprehensive reviews of the related works can be found in [2], [3].

Furthermore, face recognition can make use of a wide range of inexpensive consumer camera such as DV cameras, and embedded camera in mobile devices. Artificial Neural Networks increase the generalization accuracy in face recognition [4].

Artificial Neural Networks (ANNs) are widely used in the areas of the following categories, i)Function approximation including time series prediction, fitness approximation and modeling, ii)Classification, including pattern and sequence recognition, novelty detection and sequential decision making, iii)Data processing including filtering, clustering, blind source separation and compression, iv)Robotics, including directing manipulators, prosthesis, Control, including Computer numerical control [5]. Here we mainly focused on face recognition for home environment application & security purpose. Now a day's pipelined implementation is essential for all kinds of operation to achieve the efficient and reduced area structure when we implemented in devices like FPGA and ASIC. The biologically inspired ANNs are parallel and distributed information processing systems. This system requires the massive parallel computation. Thus, the high speed operation in real time applications can be achieved only if the networks are implemented using parallel hardware architecture [6]. Another current end of neural research focuses on elementary neural mechanisms such as spiking neurons. Their rather simple and asynchronous behaviors have motivated several implementations on analog devices, whereas digital implementations appear as quite unable to handle large spiking neural networks, for lack of density [7].

2. DESCRIPTION OF THE SYSTEM

A high speed ANN digital architecture is implemented in this paper for increasing the speed of neuron functions for input and output mapping system.

Our implementation operates in four different phases:

Phase I includes acquiring and preprocessing of images. Phase II includes development of pipelined architecture of Neural Network for extracting weights and thresholds through the back propagation algorithm from the training and testing images. Phase III includes developing ANN, based on the concept of spiking neuron in order to reduce the complexity of the digital system. Phase IV includes recognition of images (decision finding).

Detailed description is given below,

Phase I: In our implementation the size of the images was 750 × 682 pixels. These images were viewed and further cropped into 150 × 150 pixels. Since the memory would have been insufficient to store a large size of images and videos, it would have been not practical to use the actual 856 × 682 pixel images. Testing and training images used in our implementation are shown in Fig.1. After the JPEG images were obtained, they were pre-processed and segmented and features are extracted [11]. Preprocessing includes noise suppression, deblurring, image enhancement and edge detection. Segmentation includes texture segregation, colour recognition and clustering. Initially videos acquired by the camera and then converted into frames using the following MATLAB comments,

```
obj = videoinput('winvideo',4);
preview(obj);
for i = 1:n
img = getsnapshot (obj);
Image (img);
closepreview;
```

The acquired JPEG images are converted to indexed images based on a RGB color system. Each pixel of training and testing image was classified into one of 256 categories, and each

category is represented by an integer in the range from 0 to 255, where 0 represents black and 255 represents white. Each assigned colour index number served as an ANN input and, therefore, there are 22500 (150 × 150) inputs for a single image. These inputs are given to develop Neural Network. During training phase of ANN the images are represented with binary output data. Nine images of same person Fig.1 and another nine images different persons Fig.2 are used to train and test the ANNs. Features of the images are extracted for the excellent recognition of images.

Phase II: In the second phase a high speed ANN architecture is developed for increasing the speed of neuron functions for input and output mapping of the system. In our proposed ANN, the input layer has twelve neurons and uses log sigmoid transfer function. The output layer has one neuron and uses log sigmoid transfer function. Neural Network is trained using back propagation algorithm.

NN trains the network on training data using Supervised Learning which uses a Gradient Descent [8]. The training process was carried out, until a maximum of 5000 epochs (iterations) (illustrated in Fig.4), or the desired mean squared error was achieved. The number of hidden layer is varied from 150 to 1000. The successful learning rate for every image is determined after the training process. This is carried out till the desired Mean Square Error (MSE) and the target output is achieved. At the desired MSE the weights and bias are extracted and considered as optimized parameters. The extracted weights and biases are given into the FPGA.

Fig.1. Image AAA-Trainig image set

Fig.2. Image BBB-Testing image set

The neural network is trained with the error function,

$$E[\omega,\theta]=\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\left(O(x_i,y_j)-z_{ij}\right)^2 \quad (1)$$

where, n_x and n_y are the dimensions of the image in terms of number of pixels in the x, y directions. Each pixel is specified by x_i, y_j, z_{ij} where z_{ij} in the gray intensity of the pixel. $O(x_i, y_j)$ is the output of the neural network when the input is the position of the pixel i, j. Once the Neural network has been trained, we can use it for image recognition. This can be done by comparing the error function between the trained ANN and another different image (Testing image). Below a given threshold the two images are assigned to be the same. The trained ANN is considered as compressed version of the image. In which all the information of the image are now encoded into the weights and thresholds. We can find that "similar" image gets a much smaller value of the error function than a "different" image.

The error of output neuron k after the activation of the network on the n^{th} training example $\{x(n), y(n)\}$ is:

$$e(n) = x_k(n) - y_k(n). \quad (2)$$

The network error is derived from the sum of the squared errors of the output neurons:

$$E(n)=\sum e_k^2(n). \quad (3)$$

The total mean squared error is the average of the network errors of the training examples

$$E_{AV}=\frac{1}{N}\sum_{n=1}^{N}E(n). \quad (4)$$

In the same approach testing images are acquired by camera and then trained by ANN. The extracted weights and bias are taken into FPGA and compared with trained inputs. More complex images will require larger architectures. Design flow of image recognition is described in Fig.3. The features of training and testing images are compared by ANN, if the images are same then the following result is displayed "IMAGES ARE SAME" otherwise the displayed message is "IMAGES ARE NOT SAME". This paper reports on the implementation of an Artificial Neural Network (ANN) on Field Programmable Gate Array (FPGA). The work was carried out as an experiment in mapping a bit-level, logically intensive application onto the specific logic resources of a fine grained FPGA. By exploiting there configuration capabilities of the FPGA, individual layers of the network are time multiplexed onto the logic array [9],[12]. This allows a larger ANN to be implemented on a single FPGA at the expense of slower overall system operation. Developing ANN based image recognition system consists of the following steps.

Feed forward Artificial Neural Network is developed and trained using the following MATLAB comments,

[inputs, targets] = image dataset;

net = newff (inputs, targets, 10)

trainFcn = net.trainFcn

As shown in the Fig.3 the network has been structured for this application. To start this process the initial weights are chosen randomly. Then the training begins. There are two approaches are used to train the NN.

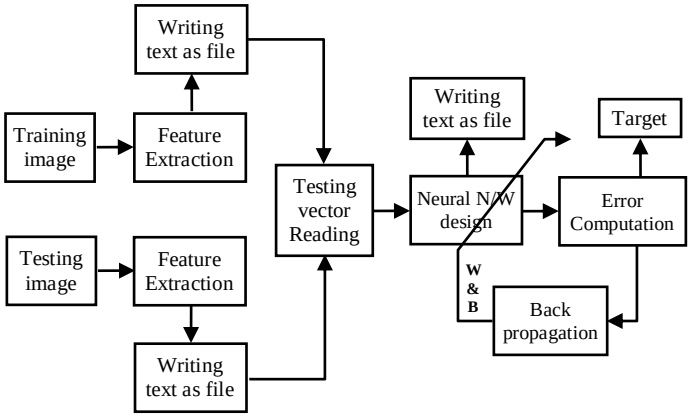

Fig.3. Design flow of ANN for image recognition

They are supervised and unsupervised learning process. In this case Supervised training is used which uses a known dataset (named as training dataset) to make predictions. The training dataset includes input data and response values. From these values, the supervised learning algorithm seeks to build a model that can make predictions of the response values for a new dataset [10], [11].

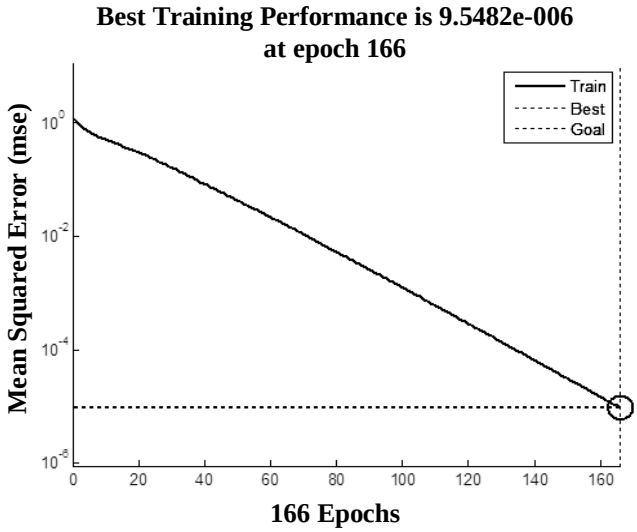

Fig.4. MATLAB based illustration ANN training (Number of iteration = 5000)

3. FEATURE EXTRACTION USING MATLAB

The feature extraction takes a matrix of pixels and returns a vector of information that is consistent for any one input matrix (the same input always gives the same output vector). In pattern recognition feature extraction is a special form of dimensionality reduction; the feature extraction follows the following steps.

Step 1: Get the training image.

Step 2: Convert the image into gray scale and resize the image into fixed size.

Step 3: Convert the gray scale pixel into equivalent binary pattern.

Step 4: Set the threshold value, if the pixel value is greater than the threshold value means consider as '1'. If the pixel value is below than the threshold value means consider as '0'.

Step 5: Form the matrix and arrange into single dimensional vector, and convert the same into a text file.

Step 6: Follow the above steps for testing image and obtain another text file.

These two text files are fed to the Artificial Neural Network to perform the identification.

If the training and testing images are same it displays the target output. If the image is not matched, uses back propagation algorithm to back propagate the error values until we get the target output.

4. IMPLEMENTATION OF BACK PROPAGATION ALGORITHM

Back propagation has proven to be so powerful that it currently accounts for 80% of all neural network applications [9]. In Back propagation, a third neurons layer is added (the hidden layer) and the discrete thresholding function is replaced with a continuous (sigmoid) one as shown in the Fig.4. But the most important modification for Back propagation is the generalized delta rule, which allows for adjustment of weights leading to the hidden layer neurons in addition to the usual adjustments to the weights leading to the output layer neurons.

Each propagation involves the following steps:

During the step – I

1) Generating propagation output activations from the forward propagation of the inputs (training pattern) through the neural network.
2) Output obtained in step1 is propagated towards the back through the neural network using the training pattern target output in order to generate the deltas of all output and hidden neurons.

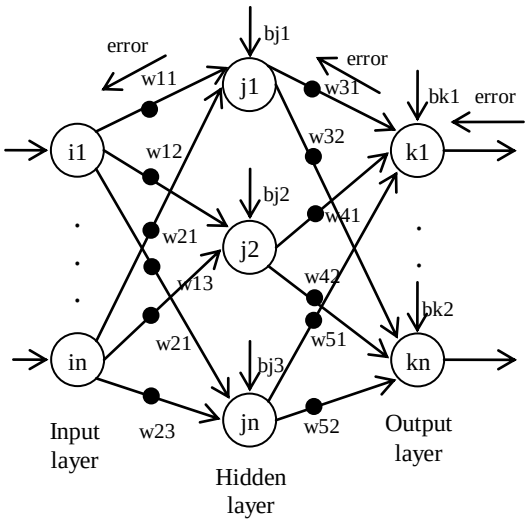

Fig.4. Illustration of back propagation algorithm

During the Step – II: Updating weights

For each weight-synapse follow the following steps:

1) Gradient of the weight is obtained by multiplying its output delta and input activation.
2) Bring this weight in the opposite direction of the gradient by subtracting a ratio of it from the weight.

This ratio influences the speed and quality of learning. The sign of the gradient of a weight indicates where the error is increasing; this is why the weight must be updated in the opposite direction.

Repeat step 1 and 2 until the performance of the network is satisfactory.

5. DESIGN OF PIPELINED ARCHITECTURE OF ANN USING SPIKING IN FPGA

With the introduction of FPGAs, it is feasible to achieve the following features (i) Fully custom hardware for designed application. (ii) Flexibility: Changes in the designs can be accomplished within a short time, and thus result in considerable savings in cost and design [12]. Pipelined architecture of ANN is shown in Fig.5. Spartan-3 (XC3S400 PQ208-5) FPGA operates at 4Mhz operating frequency, by developing clock manager unit master clock is divided into multiple frequency and treated as operating frequency for other units such as serial communication manager, time machine, RAM, etc., to achieve the pipelined structure. Once the accumulator array value reaches the threshold means the neuron gets fired, and the decision will be taken by that neuron.

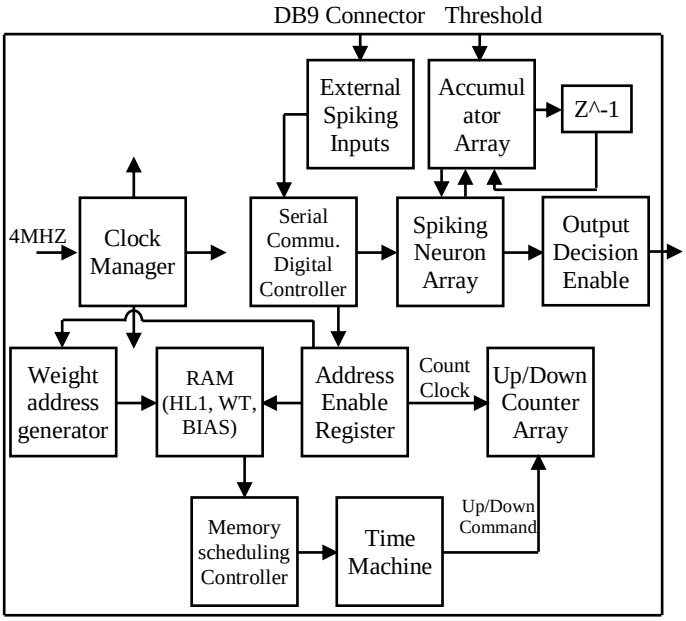

Fig.5. FPGA based pipelined architecture of image recognition system

In case of multi-layer perceptron every neuron is triggered in each propagation cycle. But the basic idea of SNN is not to fire the every neuron at each propagation cycle, but rather fire only when an intrinsic quality of the neuron related to its membrane electrical charge reaches the desired threshold value. When a neuron fires, it generates a $spike_{ON}$ signal which travels to other neurons which, in turn, $spike_{ON}$ or $spike_{OFF}$ their potentials in accordance with this signal.

In SNN environment, the activation level is modeled as some differential equation with the inputs acquired from the MATLAB (neuron's state), with incoming spikes pushing this value higher, and then either firing or decaying over time.

This design choice of SNN leads to a great simplification of the image recognition architecture. In SNN environment the $Spike_{ON}$ and $Spike_{OFF}$ signals are used to explain the concept of synaptic weights in time instead of currents.

When the Neuron1 fires, a $spike_{ON}$ signal is applied to Neurn4. This gives the increment of current in Neuron4 by 1 unit. When Neuron2 fires, another $spike_{ON}$ signal is sent to Neuron4 causing the increment of current in Neuron4 by another unit, the new value becomes 2 units. When Neuron3 also fires, another $spike_{ON}$ signal is given to Neuron4 causing the increment of current in Neuron4 by another unit, now the new value becomes 3 units. After the interval $t_1 + w_1$ has elapsed, a $Spike_{OFF}$ signal is sent to Neuron4 causing current to be now equal to 1 unit, and this happened for next time interval also. All $Spike_{ON}$ value is incremented by the counter by one and every $Spike_{OFF}$ causes the counter to decrement by 1. The particular $spike_{ON}$ or $Spike_{OFF}$ signal, is selected by the corresponding address line [7].

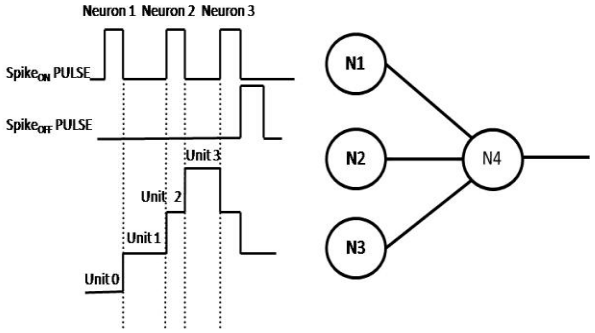

Fig.6. Representation of spiking neuron

6. RESULTS AND CONCLUSION

We have proposed and designed a new hardware pipelined architecture system for neural network based on specialized spiking neuron for image recognition system. One of the most important features of spiking neuron is that the current activation level is normally considered to be the neuron's state. Those spiking neuron concept are also cooperative in order to solve complex recognition problems. As an example of the image recognition system application, network can be trained well to identify special points on an image. Autonomous image recognition or intelligent image identifying systems for industry, home, cars use this kind of system.

REFERENCES

[1] Shatrughan Modi, "Automated Coin recognition system using ANN", *International Journal of Computer Applications*, Vol. 26, No. 4, pp. 13, 2011.

[2] Aleix M. Martinez, "Matching expression variant faces", *Vision Research*, Vol. 43, No. 9, pp. 1047-1060, 2003.

[3] Chengjun Liu, "Gabor-based kernel PCA with fractional power polynomial models for face recognition", *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No. 5, pp. 572-581, 2004.

[4] J. Harkin, F. Morgan, L. McDaid, S. Hall, B. McGinley and S. Cawley, "A reconfigurable and biologically inspired paradigm for computation using network-on-chip and spiking neural networks", *International Journal of Reconfigurable Computing*, Vol. 2009, pp. 1-13, 2009.

[5] Pallabi Parveen and Bhavani Thuraisingham, "Face Recognition Using Various Classifiers: Artificial neuron network, Linear discriminant and principal component analysis", Technical report UTDCS-05-06, 2006.

[6] B. Schrauwen, M. D'Haene, D. Verstraeten and J.V. Campenhout, "Compact hardware liquid state machines on FPGA for real-time speech recognition", *Neural Networks*, Vol. 21, No. 2–3, pp. 511-523, 2008.

[7] Vaibhav Garg, Ravi Shekar and J.G. Harris, "Spiking Neuron Computation with the Time Machine", *IEEE Transactions on Biomedical Circuits and Systems*, Vol. 6, No. 2, pp. 142-155, 2012.

[8] Juan A. Ramírez-Quintana, Mario I. Chacon-Murguia and Jose F. Chacon-Hinojos, "Artificial Neural Image Processing Applications: A Survey", *Engineering Letters*, Vol. 20, No. 1, pp. 68, 2012.

[9] G. Cauwenberghs, D.H. Goldberg and A.G. Andreou, "Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons", *Neural Networks*, Vol. 14, No. 6-7, pp. 781-793, 2001.

[10] Rich Caruana and Alexandru Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms", *Proceedings of the 23rd International Conference on Machine Learning*, pp. 161-168, 2006.

[11] M. Egmont-Petersena, D. de Ridder and H. Handels, "Image processing with neural networks-a review", *Pattern Recognition*, Vol. 35, No. 10, pp. 2279-2301, 2002.

[12] C.E. Cox and W.E. Blanz, , "GABGLION-A Fast Field Programmable Gate Array Implementation of a Connectionist Classifier", *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 3, pp. 288-299, 1992.