7

Video Based Face Recognition Using Convolutional Neural Network

Shefa A. Dawwd and Basil Sh. Mahmood
Computer Engineering Department, University of Mosul, Iraq

1. Introduction

This chapter addresses an improved approach to video face recognition (VFR). Techniques to recognize faces in video streams have been described in the literature for more than 20 years (Wang et al., 2009). Early methods were based on the still-to-still techniques which aimed at selecting good frame and did some relative processing. Recently researchers began to truly solve such problems by spatio-temporal representations. Most of the existing systems address video-based face recognition problems as follows: First, detect face and track it over time. Sometimes selecting good frames which contain frontal faces or valued cues is necessary. Next, when a frame satisfying certain criteria (size, pose, illumination and etc.) is acquired, recognition is performed, sometimes, by using still-to-still recognition technique. Figure 1 shows the whole process. In addition, some methods also utilize combination cues, such as audio, gait and so on, to make a comprehensive analysis and take decision (Yang & Waibel, 1996).

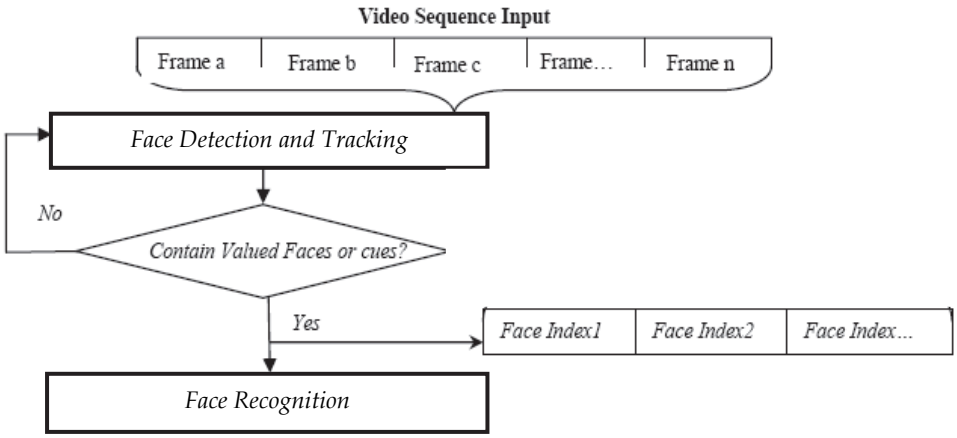


Fig. 1. Video based Face recognition system

The system proposed in this chapter employs the neural network techniques for both face detection and recognition. The face detector uses the frame color information while a

grayscale frame is required for the recognition process. The two essential processes are implemented in real time using FPGA to achieve the requirement of video processing. A short description of the proposed system is described in the following sections. Section 2 includes a short introduction on face detection and tracking. After that, a concentration on the recognition process which is based on using the Convolutional Neural Network (CNN) is presented in the rest of this chapter.

2. Face detection and tracking

Real-time face detection is important in the video-based face detection. A real-time face detection methods can uses color information to detect and validate human face (Dawwd et al., 2008; Dawwd, 2009). A hybrid adaptive face detection system which combined the advantages of knowledge based and neural methods is presented in face detection process. This system is a special-purpose object detector that can segment arbitrary objects in real images with a complex distribution in the feature space in real time. This is achieved after training with one or several previously labeled image(s). The adaptive segmentation system uses local color information to estimate the membership probability in the object, respectively, background class. The system can be applied to detect and to localize the human face in colored images in real time. To increase the detection speed, the system is implemented primarily in hardware using FPGA techniques (see Fig. 2).

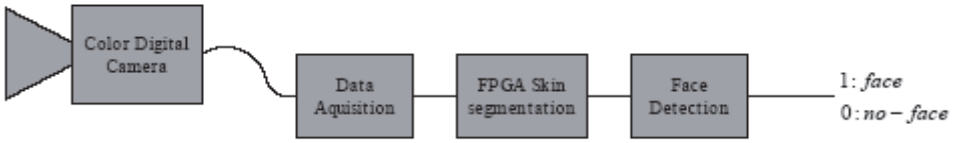
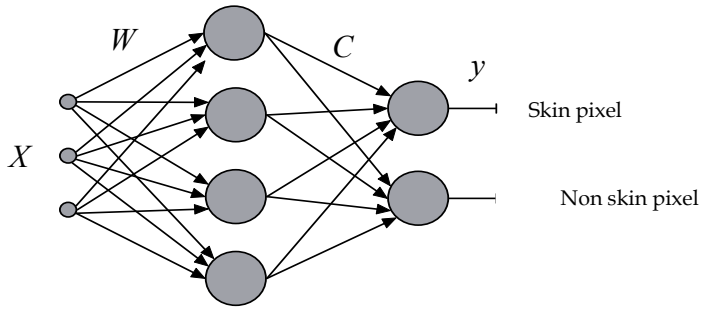


Fig. 2. Face Detector

Here, a method to detect skin color is presented. The skin detector uses a multi layered perceptron (MLP) with three inputs, one hidden layer and two output neurons (see Fig. 3). Each pixel is represented by either RGB (red, green and blue) or Yuv color components. These three color components are used as inputs by the neural network. The network output is given by:

$$y = \sum_{j=1}^{Q} c_j \varphi_j(X) + \beta \tag{1}$$

where $\varphi_j(X)$ is the output of the j-th hidden neuron, and c_j is the synaptic weight of the output neuron. To estimate the neural network parameters (i.e. synaptic weights and biases), a training set containing thousands skin and non-skin pixels was extracted from set of images. The network was trained using backpropagation algorithm. The generalization ability of the trained network is tested using a set containing several thousands of skin and nonskin pixels. The training and test sets were extracted from images containing skin colors of people from different races and under different lighting conditions. The final block of Fig.2 focus on the geometric face shapes of the segmented skin regions to distinguish them from the regions other than faces. Once the face is detected, then it can be traced afterward. Color and shape are important cues for tracking, based on which many methods are proposed, in (Yang & Waibel, 1996) a review of different robust face detectors and trackers are introduced.



where X = {R, G, B} or {Y, u, v}

Fig. 3. Neural Network structure for pixel skin detection

3. Convolutional neural network

Convolutional neural networks (CNN) with local weight sharing topology gained considerable interest both in the field of speech and image analysis. Their topology is more similar to biological networks based on receptive fields and improves tolerance to local distortions. Additionally, the model complexity and the number of the weights are efficiently reduced by weight sharing. This is an advantage when images with high-dimensional input vectors are to be presented directly to the network instead of explicit feature extraction that results in reduction which is usually applied before classification. Weight sharing can also be considered as alternative to weight elimination in order to reduce the number of the weights. Moreover, networks with local topology can more effectively be migrated to a locally connected parallel computer than fully connected feedforwared network (Neubauer, 1998).

The term CNN is used to describe an architecture for applying neural networks to two-dimensional arrays (usually images), based on spatially localized neural input. This architecture has also been described as the technique of shared weights or local receptive fields (Browne & Ghidary, 2003).The concept of sharing weights is that, a set of neurons in one layer using the same incoming weight. The use of shared weights leads to all these neurons detecting the same feature, though at different positions in the input image (receptive fields); i.e. the image is convolved with a kernel defined by the weights. The weight sharing technique has the interesting side effect of reducing the number of free parameters, thereby reducing the capacity" of the machine. Weight sharing also reduces the gap between test error and training error. This advantage is significant in the field of image processing, since without the use of appropriate constraints, the high dimensionality of the input data generally leads to ill-posed problems. Processing units with identical weight vectors and local receptive fields are arranged in a spatial array, creating architecture parallels to models of biological vision systems (Fukushima & Miyake,1982).

4. Neocognitron neural network

Fukushima (Fukushima & Miyake, 1982) were amongst the first to experiment with convolutional neural networks and obtained good results for character recognition by

applying convolutional neural networks within an image pyramid architecture: processing layers alternate between convolution and sub-sampling. This architecture is called Neocognitron. This multi-scale architecture has been now widely adopted and appears to provide a robust representation in many object recognition problems. A practical architecture of the Neocognitron is shown in Fig. 4.

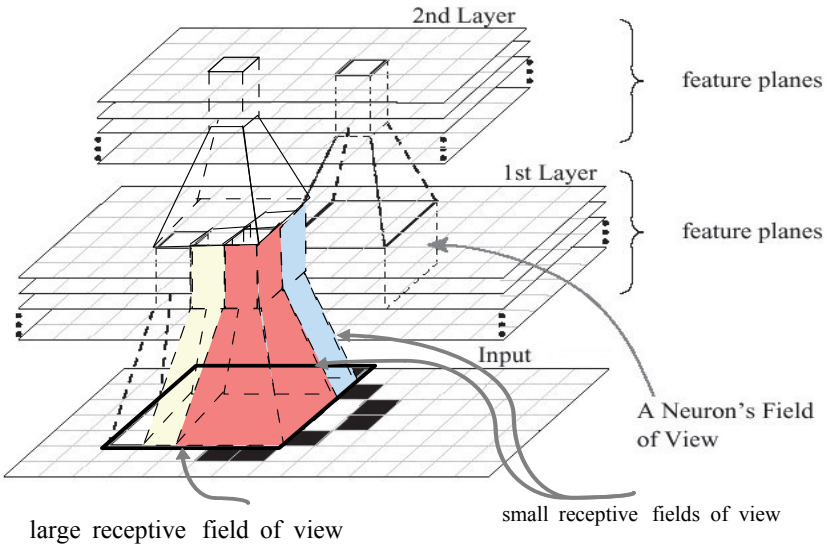


Fig. 4. Neocognitron main layers and receptive fields

Each layer extracts certain shape-features, as for example edge orientation, from a localized region of the preceding layer and projects the extracted information to the next higher layer. The complexity and abstractness of the detected features grow with the layer height, until complicated objects can be recognized. A layer consists of a number of feature planes, each of which is assigned to recognize one specific image feature.

Neurons belonging to the same plane are identical in the sense that they share the same synaptic weights. This architecture, showing a high degree of self-similarity, seems particularly dedicated to be implemented on a parallel hardware platform.

For simplicity, another illustration of the Neocognitron when the feature planes are arranged serially and the receptive fields are represented as circles is shown in Fig.5. Note that the modified Neocognitron (MNEO) is proposed and implemented in this chapter.

The Neocognitron consists of a cascade connection of a number of modular structures preceded by an input layer *Uo*. Each of the modular structures is composed of two sub-layers of cells, namely a sub-layer *Us* consisting of S-cells, and a sub-layer *Uc* consisting of C-cell (S-cells and C-cells are named after simple cells and complex cells in physiological terms, respectively). Regard to CNN cells and layers names, S-cells refer to cells in convolution layers whereas C-cells refer to cells in down-sampling layers. In the Neocognitron, only the input interconnections to S-cells are variable and modifiable and in contrast to the down-sampling layers in CNN, the input interconnections to C-cells are fixed and unmodifiable.

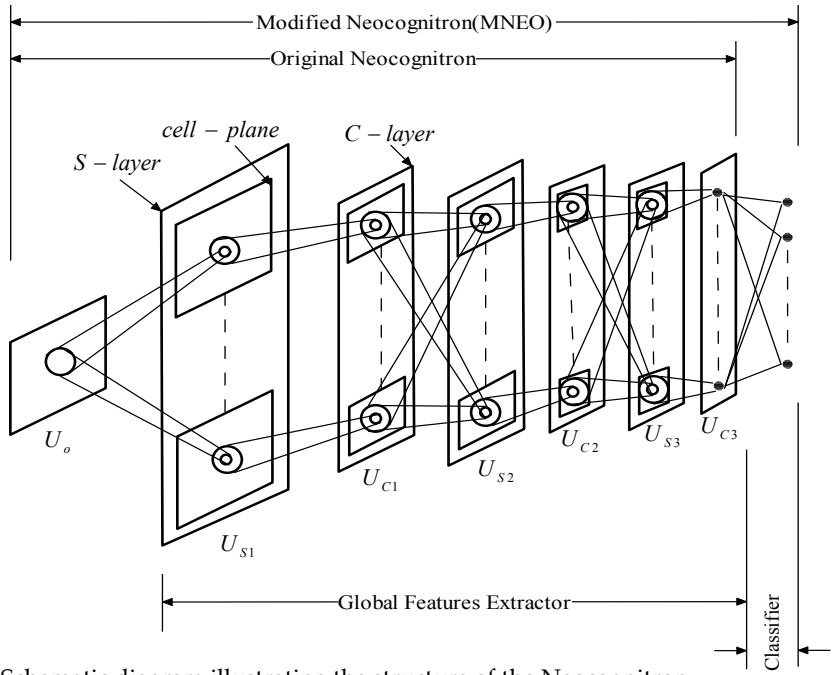


Fig. 5. Schematic diagram illustrating the structure of the Neocognitron

4.1 Cells employed in the neocognitron

All the cells employed in the Neocognitron are of analogue type: i.e., the input and output signals of the cells have non-negative analogue values. Each cell has characteristics analogous to a biological neuron. In the Neocognitron, four different types of cell are used, i.e., S-cells, C-cells, Vs-cells and Vc-cells.

An S-cell has a lot of input terminals, either excitatory or inhibitory. If the cell receives signals from excitatory input terminals, the output of the cell will increase. On the other hand, a signal from an inhibitory input terminal will suppress the output. Each input terminal has its own interconnecting coefficient whose value is positive. Although the cell has only one output terminal, it can send signals to a number of input terminals of other cells.

An S-cell has an inhibitory input which causes a shunting effect. Let *u(1), u(2),…….u(N)* be the excitatory inputs and *v* be the inhibitory input. The output *w* of this S-cell is defined by (Fukushima & Miyake, 1982):

$$w = \varphi\left[\frac{1+e}{1+h} - 1\right] = \varphi\left[\frac{e-h}{1+h}\right] \tag{2}$$

where:

$$e = \sum_{\mathrm{v}=1}^{N} a(\mathrm{v}).u(\mathrm{v})$$
$$h = b.v$$

$$\varphi[x] = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

where $a(v)$ and b represent the excitatory and inhibitory interconnecting coefficients, respectively

The cells other than S-cells also have characteristics similar to those of S-cells. The input-to-output characteristics of a C-cell are obtained from the last equation if we replace $\varphi[\ ]$ by $\psi[\ ]$, where $\psi[\ ]$ is a saturation function defined by:

$$\psi[x] = \begin{cases} \dfrac{x}{\alpha + x} & x > 0 \\ \\ 0 & x \leq 0 \end{cases} \tag{3}$$

The parameter α is a positive constant which determines the degree of saturation of the output. In the computer simulation and in hardware implementation, the parameter α is chosen to be equal to zero. S-cells and C-cells are excitatory cells, i.e., the output terminals of these cells are connected only to excitatory input terminals of other cells. On the other hand, Vs-cells and Vc-cells are inhibitory cells, whose output terminals are connected only to inhibitory input terminals of the other cells. A Vs-cell has only excitatory input terminals and the output of the cell is proportional to the sum of all the inputs weighted with the interconnecting coefficients. That is a Vs-cell yields an output proportional to the (weighted) arithmetic mean of its inputs. A Vc-cell also has only excitatory input terminals, but its output is proportional to the (weighted) root-mean-square of its input. Let $u(1)$, $u(2)$,…….$u(N)$ be the inputs to a Vc-cell and $c(1)$,$c(2)$,…….$c(N)$ be the interconnecting coefficients of its input terminals. The output w of this Vc-cell is defined by:

$$w = \sqrt{\sum_{v=1}^{N} c(v).u^2(v)} \tag{4}$$

4.2 Formulae governing the network

S-cells have inhibitory inputs with a shunting mechanism. The output of an S-cell of the k_l-th S-plane in the l-th module is given by (Fukushima & Miyake, 1982)

$$u_{sl}(k_l, n) = r_{ol}.\varphi\left[\frac{1 + \sum_{k_{l-1}=1}^{K_{l-1}} \sum_{v \in S_l} a_l(k_{l-1}, v, k_l).u_{cl-1}(k_{l-1}, n+v)}{1 + \dfrac{r_{ol}}{1+r_{ol}}.b_l(k_l).v_{cl\text{-}1}(n)} - 1\right] \tag{5}$$

where:

$$v_{cl-1}(n) = \sqrt{\sum_{k_{l-1}=1}^{K_{l-1}} \sum_{v \in S_l} c_{l-1}(v).u_{cl-1}^2(k_{l-1}, n+v)}$$

$\varphi[\,]$: a function defined by equation 2
$v_{cl-1}(n)$: Vc-cell,layer *l-1*, position *n*
$a_l(\,), b_l(\,)$: modifiable weights
$c_{l-1}(\,)$: positive fixed weights
r_{ol}: selectivity parameter
S_l: receptive field

The selectivity parameter r_{ol} in the above equation controls the intensity of the inhibition. The larger the value of r_{ol} is, the more selective becomes the cell's response to its specific feature. r_{ol} is believed that it is a key factor to control the ability of the Neocognitron to recognize deformed patterns. If the selectivity is too high, the Neocognitron loses the ability to generalize and cannot recognize deformed patterns robustly. If the selectivity is too low, the Neocognitron loses the ability to differentiate between similar patterns of different categories. The values of fixed interconnections $c_{l-1}(\,)$ are determined so as to decrease monotonically with respect to $|v|$. The size of the connecting area S_l of these cells is set to be small in the first module and to increase with respect to depth *l*.

The interconnections from S-cell to C-cell are fixed and unmodifiable as mentioned. Each C-cell has input interconnection leading from a group of S-cells in the S-plane preceding it (i.e., in the S-plane with the same k_l–number as that of the C-cell). This means that all of the S-cells in the C-cell's connecting area extract the same stimulus features but from slightly different positions on the input layer. The values of the interconnections are determined in such a way that the C-cell will be activated whenever at least one of these S-cells is active, Hence, even if a stimulus pattern which has elicited a large response from the C-cell is shifted a little in position, the C-cell will still keep responding as before, because another neighboring S-cell in its connecting area will become active instead of the first. In other words, a C-cell responds to the same stimulus feature as the S-cell preceding it, but is less sensitive to a shift in position of the stimulus feature

Quantitatively, the output of a C-cell of the k_l-th C-plane in the *l*-th module is given by:

$$u_{cl}(k_l, n) = \psi\left[\frac{1 + \sum_{v \in D_l} d_l(v).u_{sl}(k_l, n+v)}{1 + v_{sl}(n)} - 1\right] \tag{6}$$

where:

$$v_{sl}(n) = \frac{1}{K_l}\sum_{k_l=1}^{K_l}\sum_{v \in D_l} d_l(v).u_{sl}(k_l, n+v)$$

$\psi[]$: a function defined by equation 3
$v_{sl}(n)$: Vs-cell,layer *l*, position *n*
$d_l(v)$: fixed interconnection which determined so as to decrease monotonically with respect to $|v|$
D_l: receptive field, the size of D_l is set to be small in the first module and to increase with the depth of *l*

4.3 Learning rules

The Neocognitron is trained layer by layer starting from the first hidden layer. After training of the first hidden layer with images containing only simple features, the training set for the

next layer, containing more complex patterns, is propagated through the first layer is reached. This procedure is repeated until the output layer is reached. Thus higher layers represent features of increasing complexity. One advantage of this approach is that the first hidden layer does not have to be retrained for each classification problem since, for typical visual recognition tasks, edge usually have to be extracted at the first level. The reinforcement learning rule proposed by Fukushima is used here both for supervised and unsupervised training of the Neocognitron (Fukushima & Miyake, 1982):

$$\begin{aligned} \Delta a_l(k_{l-1}, v, \hat{k}_l) &= q_l . c_{l-1}(v) . u_{cl-1}(k_{l-1}, \hat{n} + v) \\ \Delta b_1(\hat{k}_l) &= q_l . v_{cl-1}(\hat{n}), \end{aligned} \tag{7}$$

where q_l is a positive constant which determines the speed of increment.

4.4 Recognition by the neocognitron

In order to help with the understanding of the principles by which the Neocognitron performs pattern recognition, Fig. 6 shows a rough sketch of the working of the network in the state after completion of self-organization.

The network is assumed to be self-organized on repeated presentations of a set of stimulus patterns such as "A", "B", "C" and so on. In the state when self-organization has been completed, various feature-extracting cells are formed in the network, as shown in Fig. 6.

If pattern "A" is presented to the input layer U_0, the cells in the network yield outputs as shown in Fig 4. For instance, the S-plane with $k1$=1 in sub-layer U_{s1} consists of a two-dimensional array of S-cells which extract ∧-shaped features. Since the stimulus pattern "A" contains a ∧-shaped feature at the top of this S-plane, it yields a large output as shown in the enlarged illustration in the lower part of Fig 6.

A C-cell in the succeeding C-plane (i.e., the C-plane in sub-layer U_{c1} with $k1$=1) has interconnections from a group of S-cells in this S-plane. For example, the C-cell shown in Fig 6 has interconnections from the S-cells whenever at least one of these S-cells yields a large output. Hence, the C-cell responds to a ∧-shaped feature situated in a certain area in the input layer and its response is less affected by the shift in position of the stimulus pattern than that of the preceding S-cells. Since this C-plane consists of an array of such C-cells, several C-cells which are situated near the top of this C-plane respond to the ∧- shaped feature of the stimulus pattern "A". In sub-layer U_{c1}, besides this C-plane, other C-planes extract features with shapes like ⟋ , and so on.

In the next module, each S-cell receives signals from all the C-planes of sub-layers U_{c1} . For example, the S-cell of sub-layer U_{s2} shown in Fig.6 receives signals from C-cells within the thin-lined circles in sub-layer U_{c1} Its input interconnections have been reinforced in such a way that this S-cell responds only when ∧-shaped, ⟋-shaped, and ⟊-shaped features are presented in its receptive field with configuration like ⩚

Hence, pattern "A" elicits a large response from this S-cell, which is situated a little above the center of this S-plane. Even if the positional relation of these three features is changed a little, this cell will still keep responding, because the preceding C-cells are not so sensitive to the positional error of these features. However, if the positional relation of these three features is changed beyond some allowance, this S-cell stops responding.

Same approach can be followed if face features is required to be detected. For example, in Fig. 7, the eye features (F1 to F4) are composed to form the eye pattern.

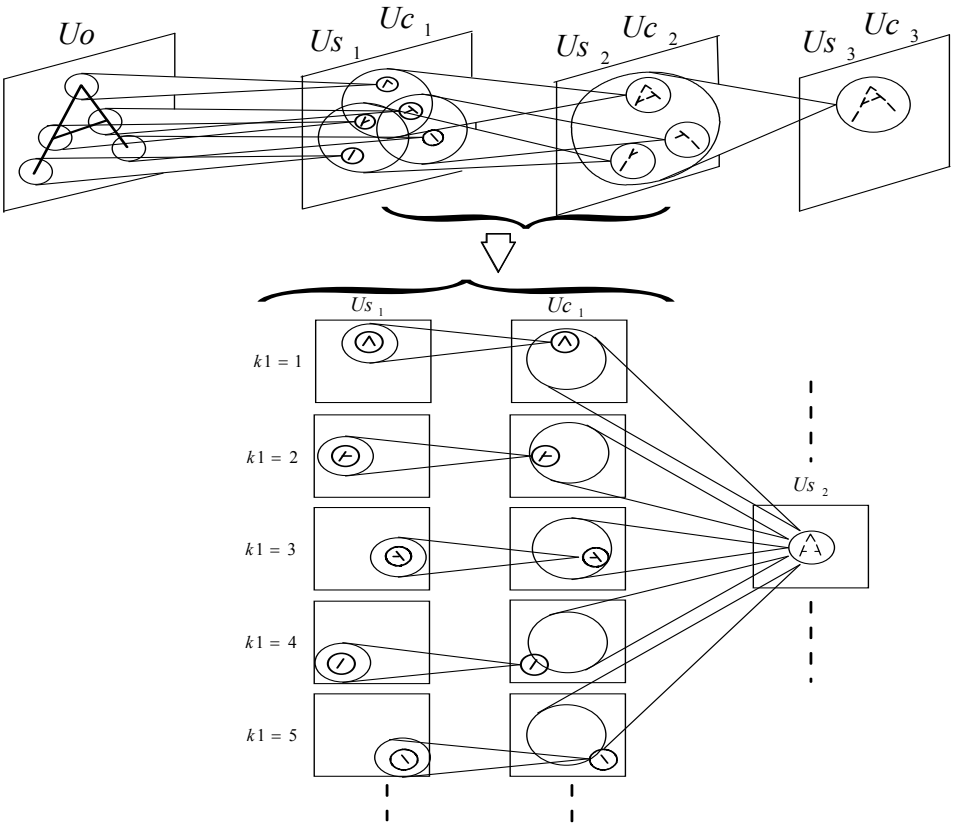


Fig. 6. (Fukushima & Miyake, 1982): An example of the interconnections between cells and the response of the cells after completion of the self-organization

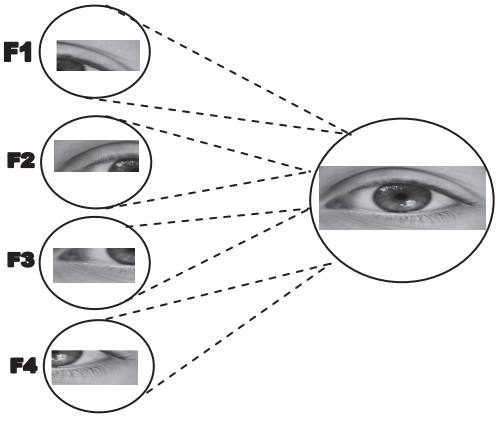


Fig. 7. Eye composition from its simple eye features

4.5 The proposed image recognition neural network system

The image recognition system described in this chapter consists of a hierarchy of several layers of artificial neurons, arranged in planes to form layers. The system consists of two parts: feature extractor and classifier. The feature extractor operates on an input image, which are then processed by the classifier (see Fig. 5). The Neocognitron is used as a feature extractor. An image is divided by the feature extractor into sub-images. The extraction of local features is based on the similarity among sub-images. The feature extractor is usually trained by unsupervised training algorithm. The training is achieved sequentially layer by layer, and the output of each layer will be considered as the input of the next layer.

The main role of the classifier is to relate the global features generated by the feature extractor (Neocognitron) to the desired recognition code. The classifier is usually feedforward and fully connected. The classifier is usually trained by supervised training algorithm. If two images belonging to the same category of a training set have different global features that result from the output of highest U_{cl} sub-layer of the Neocognitron, then, the classifier will associate these two different global features to the same recognition code. This is considered as the advantage of the classifier. It can be said that the Convolutional Image recognition system used in this chapter is based on the original Neocognitron but with some modifications and additional parts. This new structure is called MNEO to differentiate it from the original Neocognitron (see Fig. 5).

4.6 Modification of the Neocognitron(MNEO)

Since the layers of the Neocognitron in this work are independently trained, therefore, there are several possibilities for combining different kinds of neurons and learning rules. One method is proposed in his work which uses Mc Culloch-Pitts neurons in S-sublayers(Neubauer, 1998) instead of using complicated neurons based on the original Neocognitron. Also kohonen's topology preserving mapping algorithm is used for parameter adaptation (Dawwd, 2000). In order to reduce the training time, only one representation map is trained and then copy its representations to create the layer's planes. While the classifier discussed above is considered as an additional part for the original Neocognitron.

4.6.1 Simple model of neurons

In contrast to the Neocognitron, the MNEO uses S-neuron based on the Mc Culloch-Pitts model. Inhibitory cells are not used and consequently S-sublayers can be easily trained by any training algorithm. Therefore, the output of an S-cell of the k_l-*th* S-plane in the l-th module will be

$$u_{sl}(n,k_l)=\phi\left[\sum_{k_{l-1}}^{K_{l-1}}\sum_{v\in S_l} a_l(k_{l-1},v,k_l)u_{cl-1}(k_{l-1},n+v)\right] \tag{8}$$

where:

$$\phi(x)=1/(1+\exp(-x))$$

4.6.2 Learning algorithm of the MNEO

As mentioned earlier, since the Neocognitron is used for feature extraction, the unsupervised self-organizing learning algorithm (SOM) (Dawwd, 2000) can be used to

develop the representations in the S-sublayer(s). The SOM algorithm requires initializing the map size before starting of the training. Learning occurs only in S-sublayers. Essentially the algorithm modifies an unsupervised learning rule to cope with competition in a weight shared layer as follows:
First after an input has been presented to the network, the most active node (i.e. the winning neuron) is determined, second, the S-neuron connections are updated by using kohonen's rule. After learning has been completed, weight sharing is performed along the spatial to create the S-planes that represent the S-sublayer.
After learning has been completed and S-planes have been created, the input image is projected through S-sublayer. For each overlapped spatial window (each sub-image), the input vector is projected to each neuron in each S-plane at the same spatial coordinate, then the most active neuron among these planes is selected .The later operation determines which feature is included in that sub-image. Then the other neurons are set to zeros.

4.6.3 Complex model of neurons

For the designed network we do not care about the values and type of connections of C-cells to the input vector represented by the receptive field of the corresponding S-plane. As mentioned earlier to simplify the implementation of U_{cl} sub-layers, α is chosen to be zero. Since the inhibitory cells in the complex sublayer of the modified S-layer are not of use, therefore, the output of a C-cell of the k_l-th C-plane in the l-th module will be:

$$u_{cl}(k_l, n) = \psi\left[\sum_{v \in D_l} d_l(v).u_{Sl}(k_l, n+v)\right] \tag{9}$$

and $\psi[\ ]$ is defined as:

$$\psi[x] = \begin{cases} 1 & x > 0 \\ 0 & x \le 0 \end{cases} \tag{10}$$

5. Mapping neural networks on parallel computers

How can we map a specific neural network on a parallel computer to achieve maximum performance? (Schoenauer et al., 1998). The key concepts of an efficient mapping are load balancing over an array of processing elements, minimizing inter processing element communication and minimizing synchronization between the processing elements (PEs). Hence that each PE performs the computations required for a single neuron of the network. Furthermore, the mapping should be scalable both for different network sizes, and for different number of processing elements. In Fig. 8, the weight matrix presentation of a simple neural network (four neurons with four synapses each) is shown in the middle, while the left side shows the conventional presentation of the same network. The rectangle N in the mid part of Fig.8 denotes the activation function of the neuron. The circle wij represents the computation of the synapse: yi = wij*xj +yi-1 where yi-1 is the result from the proceeding synapse.
Direct implementation for non-linear sigmoid activation functions is very expensive. There are two practical approaches to approximate sigmoid functions with simple FPGA designs (Zhu & Sutton, 2003). *Piecewise linear approximation* describes a combination of lines in the

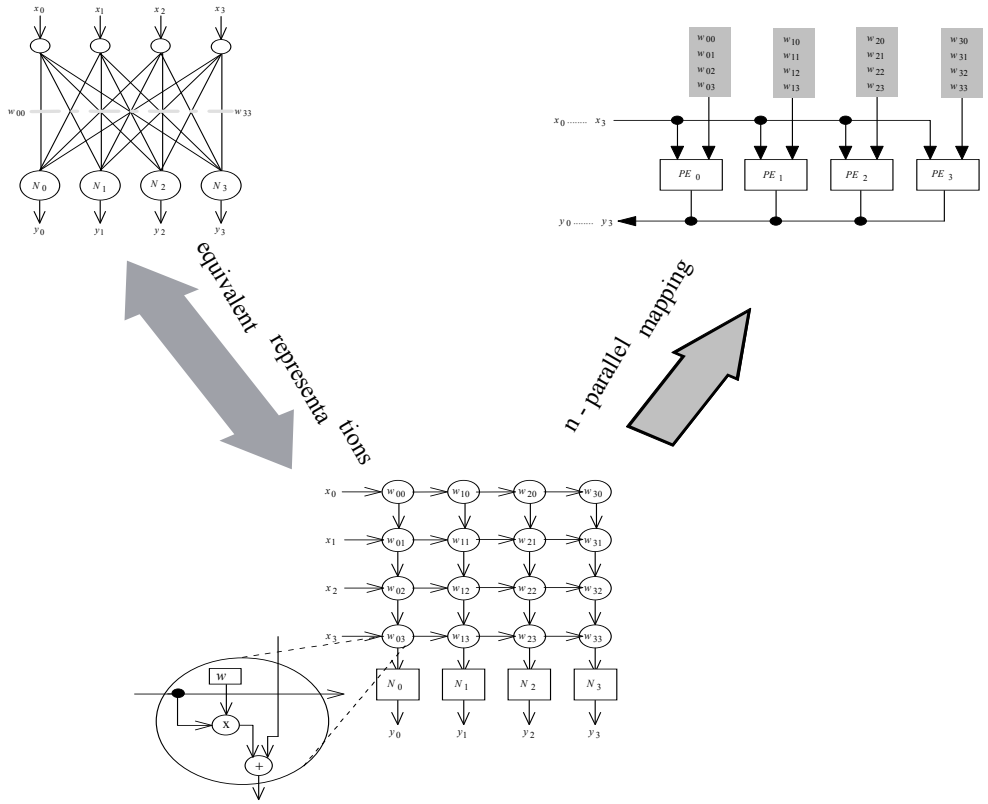


Fig. 8. Presentation of a neural network: conventional (left), weight matrix (middle), mapped N-parallel (right)

form of $y=ax + b$ which is used to approximate the sigmoid function. Note that if the coefficients for the lines are chosen to be powers of two, the sigmoid functions can be realized by a series of shift and add operations. Many implementations of neuron activation functions use such piecewise linear approximations one of them is. The second method is *lookup tables*, in which uniform samples taken from the center of sigmoid function can be stored in a table for look up. The regions outside the center of the sigmoid function are still approximated in a piece-wise linear fashion.

6. MNEO design and implementation

As mentioned previosly, the MNEO convolutional neural network model is to be implemented. This model is formed by an even number of layers, the odd ones which are simple layers have adaptable input connections and the even ones which are complex layers have fixed input connections. Each pair of layers has an equal number of planes. The number of planes is increased in a consequent manner from input to output layer in order to detect more specific features of higher complexity while the spatial resolution is decreasing. According to these properties, a special strategy is to be followed to implement the

architecture of the MNEO in hardware. In this chapter, a parallel digital hardware implementation of the MNEO in FPGA platform is presented in details.

6.1 Network parameters

The size and parameters of the MNEO neural network is dependent on the application. Therefore, a specified application should be determined beforehand. As they contain a great deal of information and many complex features which can vary significantly depending on the acquisition conditions, faces are considered one of the more challenging problems in image analysis and object recognition.

6.1.1 Still-to-Still image database

A resolution of 32x32 pixels can be considered for the task of face recognition since a face is primarily characterized by existence of eyes, nose and mouth together with their geometrical relationship all of which can be recognized at low spatial resolution (Neubauer, 1998). The Oracle Research Labs database (see Fig. 9, available in http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html), which has 10 different images of 40 distinct subjects is used in this work. The images are grayscaled with a resolution of 92x112, but the resolution is reduced to 32x32. The influence of the resolution reduction in hardware field is to reduce the probability of FPGA over-fitting, and to lower the information content that has to be learned by the networks and consequently reduce the required hardware resources. The designed network can classify 12 out of the 40 subjects. The experiments were performed on five training images and five testing images per person. A total of 60 training images and 60 testing images are used to adjust the parameters which presented in the next section. The training is wholly implemented in software.

6.1.2 Parameters setting of the MNEO

Parameters setting of the MNEO such as the choice of the number of layers, neurons, cell planes, and so on, is a complex process. In fact, this process requires a lot of 'fine-tuning' effort and can be obtained by multiple simulation run of networks with different parameters. Then the most precise network is selected and its parameters are adapted. This selection is based on the evaluation of the network with respect to the recognition rate. The parameters selection strategy used in MNEO can simulate the selection of good selectivity parameter in the original Neocognitron. The network which to be implemented in this chapter is depicted in Fig. 10. The network structure consists of five layers, first hidden layer is a simple layer of four convolutional planes. The second hidden layer is a complex layer of also four convolutional planes. The third and fourth hidden layers are simple and complex layers respectively, each is of 16 convolutional planes. There are 12 output neurons in the last layer (fully connected feedforward layer), according to 12 different subjects (faces). Receptive fields sizes are chosen as 5x5, with 4 overlapped pixels(each field is overlapped over another by four pixels in both horizontal and vertical directions). The features in the hidden layers are organized as (4x4)x4, with 2 overlapped pixels,(4x4)x4, with 3 overlapped pixels,(4x4)x16, with 2 overlapped pixels, and (4x4)x16.

6.1.2.1 Hardware implementation of the S-cell

In return to equation(8), the response of S-cell is simply a function of input vector $\hat{x}$ (receptive field) and weight vector $\hat{w}$ which can be written as (Cios & Shin, 1995):

Fig. 9. Three subjects of the ORL face database (There are 10 images for each subject).

$$u = \phi(\sum_{i=0}^{N-1} x_i.w_i) = \phi(\hat{x} \bullet \hat{w}) = \phi\left(\frac{|\hat{x}|^2 + |\hat{w}|^2 - d^2}{2}\right) \tag{11}$$

where

$d^2 = |\hat{x} - \hat{w}|^2 = |\hat{x}|^2 + |\hat{w}|^2 - 2\hat{x} \bullet \hat{w}.$

It can be seen from the above equation that the neuron response depends on the distance between $\hat{x}$ and $\hat{w}$ .Thus the smaller the distance between them, the greater the response of the neuron. Now considering that of one specific feature each receptive field has to be

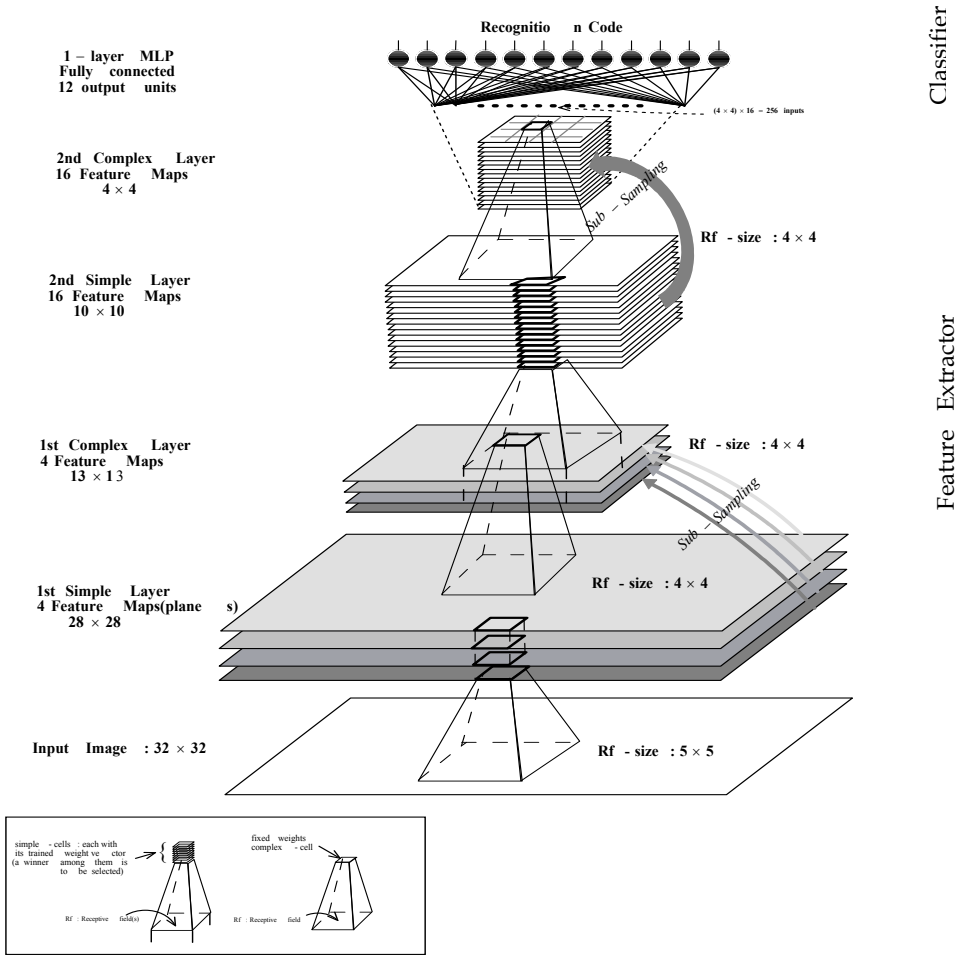


Fig. 10. The modified Neocognitron (MNEO) structure with its layer's parameters used for face recognition

detected by a number of S-cells distributed over different plans. This means, the sigmoid activation functions defined in equation (8) for those S-cells (spatially localized cells) can be replaced by a competitive function. This function sets the cell that has a minimum distance for that receptive field and resets the other cells.

The above calculation for the S-cell responses is implemented primarily in software during the MNEO training phase. For the MNEO propagation phase, the same approach is used but achieved in hardware. This ensures that the cell itself that is stimulated during the learning phase will also be stimulated during the propagation phase when the same input is applied. In this approach, since the value of the output cell is either '0' or '1', then only one storage element is required which simplifies the successive operations and their hardware. This is because there is no need to deal with real numbers that are usually produced from the

sigmoid function. This simplification also plays a positive role when implementing the down-sampling operation done by complex layers.
The selection of the similarity measure is another factor that influences on the hardware implementation of the S-cell. Manhattan distance is used as a measure of similarity between $\hat{x}$ and $\hat{w}$ (Dawwd Sh., Mahmood B., 2009). It measures the features that are detected by the S-cell either in training or in propagation phase. In particular, the Manhattan distance is used to avoid multiplications that are required in the calculation of Euclidean distance (the most critical operation in hardware). Also dot product between $\hat{x}$ and $\hat{w}$ is avoided when using Manhattan distance. Manhattan distance is defined as:

$$d = |\hat{x} - \hat{w}| = \sum_{i=o}^{N-1} |x_i - w_i| \tag{12}$$

It can be seen from the above equation that the implementation of Manhattan distance in hardware requires computational units for integer subtraction, accumulation and the determination of absolute unsigned values.

6.1.2.2 Hardware implementation of the C-cell

As it is done in simplifying the hardware implementation of the S-cell by representing its output by only one bit, the same is done with the C-cell. From equation(3) it can be seen that if the parameter a is chosen equal to 0, the output of the C-cell will be either one or zero as shown in (9). Here, also reduction of storage elements and simplification of the hardware connected to the output of the C-cell are achieved. Depending on the representation of the S-cell and the C-cell activation functions, the C-cell can be built by only using OR function. The benefits from these representations not only influence on the implementation of simple and complex layers, but also they play an important and essential role for implementing the final layer of the MNEO(the fully connected feedforwared layer).

6.1.2.3 Hardware implementation of the Feedforward-cell

In feedforward layer, the dot product calculations among the weight vectors and the receptive field vectors of the last complex layer in the MNEO require multiplications. But the mentioned modification of the MNEO network does not need this multiplication operation. Feedforward layer only needs accumulation operation and the number of required accumulators equals to the number of the feedforward cells. Each accumulator accumulates the scalar weight of a cell if its input is '1'. The equation for feedforward cell is:

$$P = \theta(\sum_{i=0}^{N-1} x_i.w_i) \tag{13}$$

θ is the sigmoid transfer function, P is the cell output , x_i and w_i are the input and weight vectors respectively. For example assume the feedforward cell receives 3 weighted inputs as $\hat{x}$ =(1,0,1) and its trained weights are $\hat{w}$ =(0.98,0.13,0.22), then the accumulator produces (0.98+0.22) which equal to 1.2. Along with the accumulator, conditioning circuitry (mainly AND gates) is used to select which value of $\hat{w}$ vector is to be accumulated. To produce cell's output, activation function (θ) (sigmoid usually used in feedforward) should be implemented and as will be shown in section 6.2, its implementation will require only one multiplication operation.

6.2 Implementation of the processing flow

Some transfer functions like sigmoid function need some modifications to simplify the hardware of the function. In this case, the sigmoid function has been substituted by a piecewise linear function like *satlin* function (Chapman, 1997). The substitution is based on the selection of a linear *satlin* equation that has a minimum least square error with the original sigmoid function. Only one multiplier and one adder are required to implement the approximated linear transfer function. Using one multiplier and one adder for each feedforward node may be also perceived as a critical problem if the number of output nodes exceeds the number of embedded multipliers in the FPGA chip. To solve this problem, only one *satlin* computational unit is built and made common to all feedforward nodes, such that the output nodes use this unit sequentially in a pipelined manner.

As it is not worthwhile to use pipelining for the successive layers of the convolutional network, then the processing units required and used for one layer can be used for the other layers. Thus, hardware is minimized and fully time utilized. Layers hardware implementation cycle is shown in Fig. 11. All receptive fields of view are processed in parallel. The reconfigurable processing elements are those responsible to generate the node's outputs for each layer. The PEs are built with adders for Manhatan distance calculations, buffers, comparators accumulators and activation function emulators which contain multipliers. All these components are fully pipelined. Each layer begins its calculation according to a control mechanism (Dawwd Sh., Mahmood B., 2009).

7. Video face recognition

Real time frame processing of video image should be implemented in the period of 1/30 second (30 frame/sec) or less than this period. To achieve this goal, detection and recognition should be achieved in fastest possible time. Therefore the original Neocognitron is modified and the MNEO is presented in this work to deal with this challenge.

Face recognition is challenging visual classification task. Reasonable deviations from a three-dimensional face shape have to be detected, also it is necessary to normalize the face with respect to the size and position and orientation. Using the Neocognitron simplifies this task very much, because the Neocognitron can recognize stimulus pattern correctly without being affected by shifts in position or by affine scaling and rotation. In this work, the MNEO is trained by using images that extracted from different positions, scales, rotations and orientations. If a trained image is applied to the system in recognition cycle, then the system should recognize it. If an image for the same class is applied but in variant pose, the system can also recognize it according to the mentioned properties of Neocognitron(or MNEO). If more training samples are used, then more generalization is achieved.

The recognition is performed in the region of interest (the segmented area). The MNEO consider the problem of face recognition under pose variations. Once the segmentation process of the colored frame is complete in the detection stage, the recognition process begins. The face detection block in Fig. 2 can be removed in our proposed system. This is achieved by increasing the training samples with variant poses for each face class as mentioned in the last paragraph. If different consequent face index frames (see Fig. 1) are recognized according to predefined statistic criteria, then the recognized face can be considered as a valid class. If the threshold value is less than the acceptable, then the

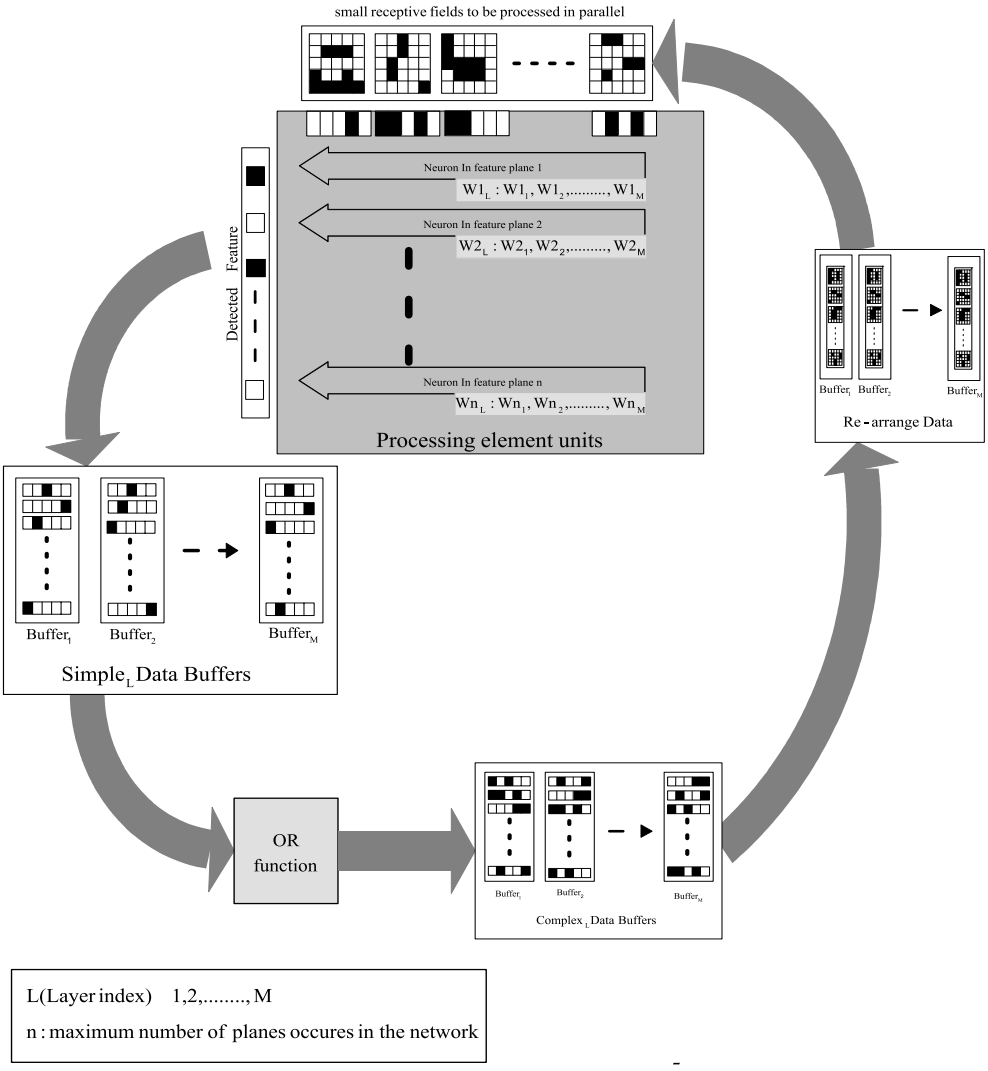


Fig. 11. MNEO layers hardware implementation cycle. All *n* feature neurons are executed in parallel, each tuned to a specific feature. Between adjacent layers, many memories act as data buffers. A special stage is necessary to re-arrange the layer output into field of views serving as input for the next layer. After implementation of all simple and complex layers, feedforward layer also uses the same processing element units to generate the final recognition code.

recognized face is not a valid class. The latter may be happened when the region of interest (the segmented region) belongs to parts other than face regions (may be hand or other color close to the color of the human skin).

8. Result

In this chapter, Xilinx Spartan-3E FPGAs are used for implementations because these devices come with large internal SRAM blocks. Each block can be used for internal weight storage and for buffering the data vectors of the segmented image.

8.1 CNN performance evaluation

In the CNN_SIMD_FPGA system, since the system does not support learning, the Connection Per Second (CPS) and Connection-bytes-per-second (CBS) are considered to evaluate the system speed performance. The most common performance rating is Connection Per Second (CPS) (Lindsey & Lindblad, 1994) which is defined as the number of multiply and accumulate operations per second during the recall or execution phase. The speed performance of the FPGA based system among other FPGA systems depends on the operation frequency of the FPGA model used. The operation frequency of the FPGA model is 50 MHz, the number of input vectors that processed in parallel are five, in each clock cycle, 5 input connections of (9bits≈1byte) are evaluated by 4 weight connections of the same bit precision, then the maximum CPS and CBS(= *bytes(weight). bytes(input). CPS)* achieved from the designed system are:

$$CPS=5\times4\times50\times10^{6}=1GCPS$$

$$CBS=1\times1\times CPS \quad =1GCBS$$

The above performance seems reasonable and comparable with the available neural network hardware (Dias et al., 2003).

8.2 CNN system and face recognition

The goal of the CNN system is to identify particular person in real-time or to allow access to a group of people and deny access to all. Multiple images per person are often available for training and real-time recognition is required.

The CNN hardware system can recognize face's image with the same recognition accuracy that achieved when using the software version. This is due to the use of efficient model, its parameters setting, functions approximations and the hardware implementation such as convolution node that is based on the realization of a competition unit. The system is trained to recognize 12 different classes. The recognition rate achieved from both software and hardware versions were equal to 93% when 60(12x5) training image and 60(12x5) testing images were used. Further recognition rate improvements can be obtained by performing more fine tuning to the CNN parameters during learning which is implemented in software. Some techniques for fine tuning improvements can be found in (Chapman ,1997).

8.3 Speedup achieved in H/W CNN

From Fig. 12, one can see that the overall time required for processing one complete image on Xilinx Spartan-3 200,000 / Spartan 3E 500,000-gates Platform FPGAs of 50MHz is equal

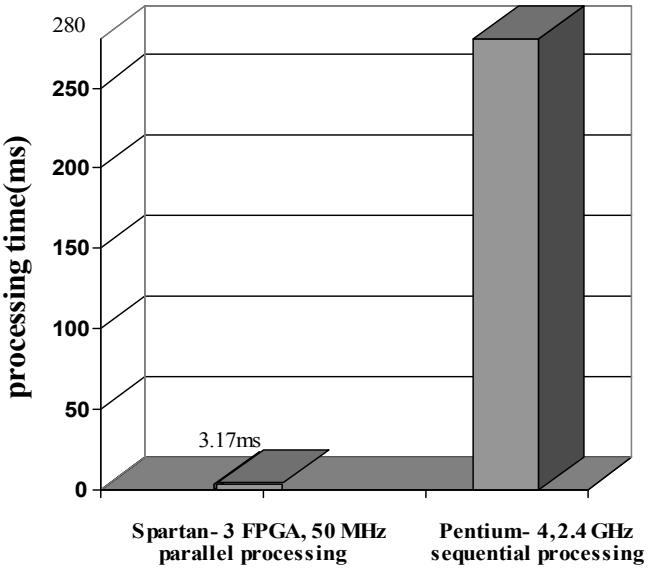


Fig. 12. An image processing time of parallel and sequential processors

to (3.17)ms, while the same model needs (280) ms when implemented primarily in software, resulting a speedup of (88).

8.4 Area consumption in H/W CNN

When the word length of the calculation unit (the input and weight value) is set to 9 bits and the word length of the accumulator is set to 13 bits, the CNN hardware required 1488 slices. This number utilizes 77% of the total number of slices in the Spartan-3 200,000 FPGA and 30% of the total number of slices in the Spartan-3E 500,000 FPGA. This means that the CNN system can be synthesized in a cheap FPGA chip. If larger system of large input image is required, the CNN system can be also synthesized in a chip of a reasonable cost.

9. Conclusion

In this work we have succeeded in mapping one of the most complex neural networks (the Neocognitron) on an FPGA SIMD architecture. The modifications of the Neocognitron to reduce its complexity give it the possibility of realizing and processing in real-time, then a high speed frame processing is achieved. Using the binary representation of cells outputs in all the network layers highly reduced the hardware resources required to implement the network. Consequently a relatively small FPGA model of 200,000 gates can implement the complex design of the MNEO system.

In the MNEO architecture, the 'sharing' of weights over processing units reduces the number of free variables, increasing the generalization performance of the network. Sharing

the weights over the spatial array, leading to intrinsic insensitivity to translations, rotation and scaling of the input which is considered as attractive feature for video processing where more number of valid faces are detected, afterward, speeding up the video recognition process.

10. References

Browne M., GhidaryS. (2003). Convolutional Neural Networks for Image Processing: An Application in Robot Vision, Lecture Notes in Computer Science, Publisher: Springer Berlin / Heidelberg, Volume 2903, pages 641 – 652.

Chapman R. (1997). Using A Neuroprocessor Model for Describing Artificial Neural Nets, Project Report for EE563, Neuroprocessor Modelin.

Cios K. and Shin I. (1995). Image recognition neural network, Neurocomputing, vol. 7, pages 159-185.

Dawwd Sh. (2000). Face Recognition using Neocognitron Neural Network, M.Sc thesis, Univ of Mosul, Mosul, Iraq.

Dawwd Sh., Mahmood B., Majeed R.(2008). Real time Image segmentation for face detection based on Fuzzy Logic, Nahrain University College of Engineering Journal, IRAQ, vol. 11,No. 2, pages 278-287.

Dawwd Sh. (2009). High Performance Colored Image Segmentation System Based Neural Network, Al-Rafidain Engineering Journal, IRAQ, vol.17, No. 2, 2009, pages 1-10.

Dawwd Sh., Mahmood B.(2009). A reconfigurable interconnectedfilter for face recognition based on convolution neural network, 4th IEEE international Workshop for Design and Test, Riyadh, Saudi Arabia, ISBN: 978-1-4244-5748-9.

Dias F. M., Antunes A. andMota A.(2003). Commercial Hardware for Artificial Neural Networks: a Survey, SICICA - 5th IFAC International Symposium on Inteligent Components and Instruments for Control Applications, Aveiro.

Fukushima K. and Miyake S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position , pattern recognition, vol. 15, no. 6, pages 455-469.

Lindsey, C. and Lindblad T. (1994).Review of Hardware Neural Networks: A User's Perspective, Proceeding of 3rd Workshop on Neural Networks: From Biology to High Energ Physics, Isola d'Elba, Italy, Sept. 26-30.

NeubauerC. (1998). Evaluation of Convolutional Neural Networks for Visual Recognition, IEEE Transactions on Neural Networks. vol. 9, no. 4, pages 685-696.

Schoenauer T., Jahnke A., Roth U., Klar H. (1998). Digital Neurohardware: Principles and Perspectives, Neuronal Networks in Applications - NN'98 – Magdeburg.

Wang H., Wang Y., Cao Y. (2009). Video-based Face Recognition: A Survey, World Academy of Science, Engineering and Technology, December 2009, Issue 60. 1307-6892.

Yang J. and Waibel A. (1996). A real-time face tracker, n Proceedings of the Third IEEE Workshop on Applications of Computer Vision, pages 142-147, Sarasota, FL.

Zhu J. and Sutton P. (2003). FPGA Implementations of Neural Networks – a Survey of a Decade of Progress, International Conference on Field Programmable Logic and Applications (FPL'03),LNCS 2778, Springer Verlag, Berlin, Heidelberg, pages 1062-1066.

New Approaches to Characterization and Recognition of Faces
Edited by Dr. Peter Corcoran

ISBN 978-953-307-515-0
Hard cover, 252 pages
Publisher InTech
Published online 01, August, 2011
Published in print edition August, 2011

As a baby, one of our earliest stimuli is that of human faces. We rapidly learn to identify, characterize and eventually distinguish those who are near and dear to us. We accept face recognition later as an everyday ability. We realize the complexity of the underlying problem only when we attempt to duplicate this skill in a computer vision system. This book is arranged around a number of clustered themes covering different aspects of face recognition. The first section presents an architecture for face recognition based on Hidden Markov Models; it is followed by an article on coding methods. The next section is devoted to 3D methods of face recognition and is followed by a section covering various aspects and techniques in video. Next short section is devoted to the characterization and detection of features in faces. Finally, you can find an article on the human perception of faces and how different neurological or psychological disorders can affect this.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821