Proceedings of the 7th WSEAS International Conference on Neural Networks, Cavtat, Croatia, June 12-14, 2006 (pp144-151)

A Novel Model of Neural Networks for Fast Data Detection

Hazem M. El-Bakry

Faculty of Computer Science & Information Systems,
Mansoura University, EGYPT

Nikos Mastorakis

Department of Computer Science,
Military Institutions of University Education (MIUE) -
Hellenic Naval Academy, Greece

Abstract- Neural networks have shown good results for detection of a certain pattern in a given image. In our previous paper, a fast algorithm for object/face detection was presented. Such algorithm was designed based on cross correlation in the frequency domain between the input image and the weights of neural networks. In this paper, a simple design for solving the problem of local subimage normalization in the frequency domain is presented. Furthermore, the effect of image normalization on the speed up ratio of pattern detection is presented. Simulation results show that local subimage normalization through weight normalization is faster than subimage normalization in the spatial domain. Moreover, the overall speed up ratio of the detection process is increased as the normalization of weights is done off line.

Keywords: Fast Neural Networks, Cross Correlation, Frequency Domain, Subimage Normalization

I. Introduction

Pattern detection is a fundamental step before pattern recognition. Its reliability and performance have a major influence in a whole pattern recognition system. Nowadays, neural networks have shown very good results for detecting a certain pattern in a given image [3,6,10]. But the problem with neural networks is that the computational complexity is very high because the networks have to process many small local windows in the images [5,7]. In our pervious papers, we presented fast neural networks based on applying cross correlation in the frequency domain between the input image and the input weights of neural networks. It was proved that the speed of these networks is much faster than conventional neural networks [1-4]. It was also proved that fast neural networks introduced by previous authors [9,11,12] are not correct. The reasons for this were given in [2].

The problem of subimage (local) normalization in the Fourier space was presented in [8]. Here, a simple method for solving this problem is presented. By using the proposed algorithm, the number of computation steps required for weight normalization becomes less than that needed for image normalization. Furthermore, the effect of weight normalization on the speed up ratio is theoretically and practically discussed. Mathematical calculations prove that the new idea of weight normalization, instead of image normalization, provides good results and increases the speed up ratio. This is because weight normalization requires fewer computation steps than subimage normalization. Moreover, for neural networks, normalization of weights can be easily done off line before starting the search process.

In section II, fast neural networks for pattern detection are described. Subimage normalization in the frequency domain through normalization of weights is presented in section III. The effect of weight normalization on the speed up ratio is presented in section IV.

II. Fast Neural Networks

Finding a certain pattern in the input image is a search problem. Each subimage in the input image is tested for the presence or absence of the required pattern. At each pixel position in the input image each subimage is multiplied by a window of weights, which has the same size as the subimage. The outputs of neurons in the hidden layer are multiplied by the weights of the output layer. A high output implies that the tested subimage contains the required pattern and vice versa. Thus, we may conclude that this searching problem is cross correlation between the image under test and the weights of the hidden neurons.

The convolution theorem in mathematical analysis says that a convolution of f with h is identical to the result of the following steps: let F and H be the results of the Fourier transformation of f and h in the frequency domain. Multiply F and H in the frequency domain point by point and then transform this product into spatial domain via the inverse Fourier transform [1]. As a result, these cross correlations can be represented by a product in the frequency domain. Thus, by using cross correlation in the frequency domain a speed up in an order of magnitude can be achieved during the detection process [1-4].

In the detection phase, a subimage X of size mxn (sliding window) is extracted from the tested image, which has a size PxT, and fed to the neural network. Let W_i be the vector of weights between the input subimage and the hidden layer. This vector has a size of mxn and can be represented as mxn matrix. The output of hidden neurons $h(i)$ can be calculated as follows:

$$h_i = g\left(\sum_{j=1}^{m} \sum_{k=1}^{n} W_i(j,k)X(j,k) + b_i \right) \qquad (1)$$

where g is the activation function and $b(i)$ is the bias of each hidden neuron (i). Eq.1 represents the output of each hidden neuron for a particular subimage I. It can be computed for the whole image Ψ as follows:

$$h_i(u,v)=g\left(\sum_{j=-m/2}^{m/2}\sum_{k=-n/2}^{n/2} W_i(j,k)\Psi(u+j,v+k)+b_i\right) \quad (2)$$

Eq. (2) represents a cross correlation operation. Given any two functions *f* and *g*, their cross correlation can be obtained by:

$$f(x,y)\otimes g(x,y)= \left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f(x+m,y+n)g(m,n)\right) \quad (3)$$

Therefore, Eq. (2) can be written as follows:

$$h_i = g\left(\Psi\otimes W_i + b_i\right) \quad (4)$$

where h_i is the output of the hidden neuron *(i)* and $h_i(u,v)$ is the activity of the hidden unit *(i)* when the sliding window is located at position *(u,v)* in the input image Ψ and $(u,v)\in[P-m+1,T-n+1]$.

Now, the above cross correlation can be expressed in terms of the Fourier Transform:

$$\Psi\otimes W_i = F^{-1}\left(F(\Psi)\bullet F^*(W_i)\right) \quad (5)$$

(*) means the conjugate of the *FFT* for the weight matrix. Hence, by evaluating this cross correlation, a speed up ratio can be obtained comparable to conventional neural networks. Also, the final output of the neural network can be evaluated as follows:

$$O(u,v)=g\left(\sum_{i=1}^{q} W_O(i)\,h_i(u,v)+b_O\right) \quad (6)$$

where *q* is the number of neurons in the hidden layer. *O(u,v)* is the output of the neural network when the sliding window located at the position *(u,v)* in the input image Ψ. W_o is the weight matrix between hidden and output layer.

The complexity of cross correlation in the frequency domain can be analyzed as follows:

1. For a tested image of *NxN* pixels, the *2D-FFT* requires a number equal to $N^2log_2N^2$ of complex computation steps. Also, the same number of complex computation steps is required for computing the *2D-FFT* of the weight matrix for each neuron in the hidden layer.

2. At each neuron in the hidden layer, the inverse *2D-FFT* is computed. So, *q* backward and *(1+q)* forward transforms have to be computed. Therefore, for an image under test, the total number of the *2D-FFT* to compute is $(2q+1)N^2log_2N^2$.

3. The input image and the weights should be multiplied in the frequency domain. Therefore, a number of complex computation steps equal to qN^2 should be added.

4. The number of computation steps required by the faster neural networks is complex and must be converted into a real version. It is known that the two dimensions Fast Fourier Transform requires $(N^2/2)log_2N^2$ complex multiplications and $N^2log_2N^2$ complex additions [13,14]. Every complex multiplication is realized by six real floating point operations and every complex addition is implemented by two real floating point operations. So, the total number of computation steps required to obtain the *2D-FFT* of an *NxN* image is:

$$\rho=6((N^2/2)log_2N^2) + 2(N^2log_2N^2) \quad (7)$$

which may be simplified to:

$$\rho=N^2log_2N^2 \quad (8)$$

Performing complex dot product in the frequency domain also requires $6qN^2$ real operations.

5. In order to perform cross correlation in the frequency domain, the weight matrix must have the same size as the input image. So, a number of zeros = (N^2-n^2) must be added to the weight matrix. This requires a total real number of computation steps = $q(N^2-n^2)$ for all neurons. Moreover, after computing the *2D-FFT* for the weight matrix, the conjugate of this matrix must be obtained. So, a real number of computation steps $=qN^2$ should be added in order to obtain the conjugate of the weight matrix for all neurons. Also, a number of real computation steps equal to *N* is required to create butterflies complex numbers $(e^{-jk(2\Pi n/N)})$, where $0<K<L$. These (N/2) complex numbers are multiplied by the elements of the input image or by previous complex numbers during the computation of the *2D-FFT*. To create a complex number requires two real floating point operations. So, the total number of computation steps required for the faster neural networks becomes:

$$\sigma=(2q+1)(5N^2log_2N^2) +6qN^2+q(N^2-n^2)+qN^2 +N \quad (9)$$

which can be reformulated as:

$$\sigma=(2q+1)(5N^2log_2N^2) +q(8N^2-n^2) +N \quad (10)$$

6. Using a sliding window of size nxn for the same image of *NxN* pixels, $q(2n^2-1)(N-n+1)^2$ computation steps are required when using traditional neural networks for face/object detection process. The theoretical speed up factor η can be evaluated as follows:

$$\eta=\frac{q(2n^2-1)(N-n+1)^2}{(2q+1)(5N^2log_2N^2)+q(8N^2-n^2)+N} \quad (11)$$

The theoretical speed up ratio (Eq. 11) with different sizes of the input image and different in size weight matrices is listed in Table 1. Practical speed up ratio for manipulating images of different sizes and different in size weight matrices is listed in Table 2 using 700 MHz processor and *MATLAB ver 5.3*.

In practical implementation, the multiplication process consumes more time than the addition one. The effect of the number of multiplications required for conventional neural networks in the speed up ratio (Eq. 11) is more than the number of of

multiplication steps required by the faster neural networks. In order to clear this, the following equation (η_m) describes the relation between the number of multiplication steps required by conventional and faster neural networks:

$$\eta_m = \frac{qn^2(N-n+1)^2}{(2q+1)(3N^2 log_2 N^2)+6qN^2} \quad (12)$$

The results listed in Table 3 prove that the effect of the number of multiplication steps in case of conventional neural networks is more than faster neural networks and this the reason why practical speed up ratio is larger than theoretical speed up ratio.

For general fast cross correlation the speed up ratio becomes in the following form:

$$\eta = \frac{q(2n^2-1)N^2}{(2q+1)(5(N+\tau)^2 log_2(N+\tau)^2)+q(8(N+\tau)^2 - n^2)+(N+\tau)} \quad (13)$$

where τ is a small number depends on the size of the weight matrix. General cross correlation means that the process starts from the first element in the input matrix. The theoretical speed up ratio for general fast cross correlation Eq. (13) is shown in Table 4. Compared with *MATLAB* cross correlation function *(xcorr2),* experimental results show that the our proposed algorithm is faster than this function as shown in Table 5.

The authors in [9,11,12] have proposed a multilayer perceptron (MLP) algorithm for fast face/object detection. The same authors claimed incorrect equation for cross correlation between the input image and the weights of the neural networks. They introduced formulas for the number of computation steps needed by conventional and faster neural networks. Then, they established an equation for the speed up ratio. Unfortunately, these formulas contain many errors which lead to invalid speed up ratio. Other authors developed their work based on these incorrect equations [1,2]. So, the fact that these equations are not valid must be cleared to all researchers. It is not only very important but also urgent to notify other researchers not to do research based on wrong equations.

The authors in [9,11,12] analyzed their proposed fast neural network as follows: For a tested image of NxN pixels, the *2D-FFT* requires $O(N^2(log_2N)^2)$ computation steps. For the weight matrix W_i, the *2D-FFT* can be computed off line since these are constant parameters of the network independent of the tested image. The *2D-FFT* of the tested image must be computed. As a result, q backward and one forward transforms have to be computed. Therefore, for a tested image, the total number of the *2D-FFT* to compute is $(q+1)N^2(log_2N)^2$ [9,12]. In addition, the input image and the weights should be multiplied in the frequency domain. Therefore, computation steps of (qN^2) should be added. This yields a total of $O((q+1)N^2(log_2N)^2+qN^2)$ computation steps for the fast neural network [9,11].

Using sliding window of size nxn, for the same image of NxN pixels, qN^2n^2 computation steps are required when using traditional neural networks for the face detection process. They evaluated theoretical speed up factor η as follows [9]:

$$\eta = \frac{qn^2}{(q+1)log^2 N} \quad (14)$$

The speed up factor introduced in [9] and given by Eq.14 is not correct for the following reasons:

a) The number of computation steps required for the *2D-FFT* is $O(N^2log_2N^2)$ and not $O(N^2log^2N)$ as presented in [9,11]. Also, this is not a typing error as the curve in Fig.2 in [9] realizes Eq.7, and the curves in Fig.15 in [11] realize Eq.31 and Eq.32 in [11].

b) Also, the speed up ratio presented in [9] not only contains an error but also is not precise. This is because for faster neural networks, the term $(6qN^2)$ corresponds to complex dot product in the frequency domain must be added. Such term has a great effect on the speed up ratio. Adding only qN^2 as stated in [11] is not correct since a one complex multiplication requires six real computation steps.

c) For conventional neural networks, the number of operations is $(q(2n^2-1)(N-n+1)^2)$ and not (qN^2n^2). The term n^2 is required for multiplication of n^2 elements (in the input window) by n^2 weights which results in another new n^2 elements. Adding these n^2 elements, requires another (n^2-1) steps. So, the total computation steps needed for each window is $(2n^2-1)$. The search operation for a face in the input image uses a window with *nxn* weights. This operation is done at each pixel in the input image. Therefore, such process is repeated $(N-n+1)^2$ times and not N^2 as stated in [9,12].

d) Before applying cross correlation, the *2D-FFT* of the weight matrix must be computed. Because of the dot product, which is done in the frequency domain, the size of weight matrix should be increased to be the same as the size of the input image. Computing the *2D-FFT* of the weight matrix off line as stated in [9,11,12] is not practical. In this case, all of the input images must have the same size. As a result, the input image will have only a one fixed size. This means that, the testing time for an image of size 50x50 pixels will be the same as that image of size 1000x1000 pixels and of course, this is unreliable.

e) It is not valid to compare number of complex computation steps by another of real computation steps directly. The number of computation steps given by pervious authors [9,11,12] for conventional neural networks is for real operations while that is required by the faster neural networks is for complex operations. To obtain the speed up ratio, the authors in [9,11,12] have divided the two formulas directly without converting the number of computation steps required by the faster neural networks into a real version.

f) Furthermore, there are critical errors in Eq.4 (equation of cross correlation) in [9] and also Eq.13 in [11]. Eq. 4 in [9] which was defined by:

$$f(x,y)\otimes g(x,y)=\left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}f(m,n)g(x+m,y+n)\right) \quad (15)$$

is not correct because the definition of cross correlation is as given by Eq.3 here in this paper. Then, Eq.4 given by those authors in [9,11,12] which is:

$$h_i = g\left(W_i \otimes \Psi + b_i\right) \quad (16)$$

is also not correct and should be written as Eq. 4 given here in this paper. Therefore, the cross correlation in the frequency domain given by (Eq.5 in their paper [9]) does not represent Eq. 16 (Eq.4 in their paper [9]) This is because the fact that the operation of cross correlation is not commutative *($W\otimes\Psi \neq \Psi\otimes W$).* As a result, Eq. 16 (Eq.4 in their paper [9]) does not give the same correct results as conventional neural networks. This error leads the researchers who consider the references [9,11,12] to think about how to modify the operation of cross correlation so that Eq. 16 (Eq.4 in their paper [9]) can give the same correct results as conventional neural networks. Therefore, errors in these equations must be cleared to all the researchers. In [2], the authors proved that a symmetry condition must be found in input matrices (images and the weights of neural networks) so that faster neural networks can give the same results as conventional neural networks. In case of symmetry $W\otimes\Psi=\Psi\otimes W$, the cross correlation becomes commutative and this is a valuable achievement. In this case, the cross correlation is performed without any constrains on the arrangement of matrices. As presented in [1], this symmetry condition is useful for reducing the number of patterns that neural networks will learn. This is because the image is converted into symmetric shape by rotating it down and then the up image and its rotated down version are tested together as one (symmetric) image. If a pattern is detected in the rotated down image, then, this means that this pattern is found at the relative position in the up image. So, if conventional neural networks are trained for up and rotated down examples of the pattern, faster neural networks will be trained only to up examples. As the number of trained examples is reduced, the number of neurons in the hidden layer will be reduced and the neural network will be faster in the test phase compared with conventional neural networks.

g) Moreover, the authors in [9,11,12] stated that the activity of each neuron in the hidden layer Eq. 16 (Eq.4 in their paper [9]) can be expressed in terms of convolution between a bank of filter (weights) and the input image. This is not correct because the activity of the hidden neuron is a cross correlation between the input image and the weight matrix. It is known that the result of cross correlation between any two functions is different from their convolution. As we proved in [1,2] the two results will be the same, only when the two matrices are symmetric or at least the weight matrix is symmetric.

h) Images are tested for the presence of a face (object) at different scales by building a pyramid of the input image which generates a set of images at different resolutions. The face detector is then applied at each resolution and this process takes much more time as the number of processing steps will be increased. In [9,11,12], the authors stated that the Fourier transforms of the new scales do not need to be computed. This is due to a property of the Fourier transform. If *z(x,y)* is the original and *a(x,y)* is the sub-sampled by a factor of 2 in each direction image then:

$$a(x,y)=z(2x,2y) \quad (17)$$

$$Z(u,v)=FT(z(x,y)) \quad (18)$$

$$FT(a(x,y))=A(u,v)=\frac{1}{4}Z\left(\frac{u}{2},\frac{v}{2}\right) \quad (19)$$

This implies that we do not need to recompute the Fourier transform of the sub-sampled images, as it can be directly obtained from the original Fourier transform. But experimental results have shown that Eq.17 is valid only for images in the following form:

$$\Psi=\begin{bmatrix} A\ A\ B\ B\ C\ C\ \ldots\ldots\ldots\ldots \\ A\ A\ B\ B\ C\ C\ \ldots\ldots\ldots\ldots \\ . \\ . \\ . \\ . \\ S\ S\ X\ X\ Y\ Y\ldots\ldots\ldots\ldots \\ S\ S\ X\ X\ Y\ Y\ldots\ldots\ldots\ldots \end{bmatrix} \quad (20)$$

In [9], the author claimed that the processing needs $O((q+2)N^2log^2N)$ additional number of computation steps. Thus the speed up ratio will be [9]:

$$\eta=\frac{qn^2}{(q+2)log^2N} \quad (21)$$

Of course this is not correct, because the inverse of the Fourier transform is required to be computed at each neuron in the hidden layer (for the resulted matrix from the dot product between the Fourier matrix in two dimensions of the input image and the Fourier matrix in two dimensions of the weights, the inverse of the Fourier transform must be computed). So, the term *(q+2)* in Eq.21 should be *(2q+1)* because the inverse *2D-FFT* in two dimensions must be done at each neuron in the hidden layer. In this case, the number of computation steps required to perform *2D-FFT* for the faster neural networks will be:

$$\varphi=(2q+1)(5N^2log_2N^2)+(2q)5(N/2)^2log_2(N/2)^2 \quad (22)$$

In addition, a number of computation steps equal to $6q(N/2)^2+q((N/2)^2-n^2)+q(N/2)^2$ must be added to the number of computation steps required by the faster neural networks.

III. Subimage Normalization in the Frequency Domain

In [6], the authors stated that image normalization to avoid weak or strong illumination could not be done in the frequency space. This is because the image normalization is local and not easily computed in the Fourier space of the whole image. Here, a simple method for image normalization is presented. Normalizing the image can be obtained by centering and normalizing the weights as follows:

Let $\overline{X}_{rc}$ be the zero-mean centered subimage located at (r,c) in the input image ψ:

$$\overline{X}_{rc} = X_{rc} - \bar{x}_{rc} \tag{23}$$

where, $\bar{x}_{rc}$ is the mean value of the sub image located at position (r,c). We are interested in computing the dot multiplication between the subimage $\overline{X}_{rc}$ and the weights W_i the of hidden layer as follows:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet W_i - \bar{x}_{rc} \bullet W_i \tag{24}$$

where,

$$\bar{x}_{rc} = \frac{\sum_{k,j=1}^{n} X_{rc}(k,j)}{n^2} \tag{25}$$

The dot multiplication denoted by (•) is not a matrix multiplication but is done element-wise (multiply each element in the first matrix by its corresponding element at the same position in the second matrix and sum up the results to obtain a one final value).

Combining Eq. (24) and Eq. (25), we get the following expression:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet W_i - \frac{\sum_{k,j=1}^{n} X_{rc}(k,j)}{n^2} \bullet W_i \tag{26}$$

For any two matrices with the same size, multiplying the first matrix dot by the mean of the second and summing the results the same as multiplying the second matrix dot by the mean of the first one and summing the results of multiplication. Therefore, Eq. (26) can be written as:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet W_i - X_{rc} \bullet \frac{\sum_{k,j=1}^{n} W_i(k,j)}{n^2} \tag{27}$$

The zero mean weights are given by:

$$\overline{W}_i = W_i - \frac{\sum_{k,j=1}^{n} W_i(k,j)}{n^2} \tag{28}$$

Also, Eq. (27) can be written as:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet \left(W_i - \frac{\sum_{k,j=1}^{n} W_i(k,j)}{n^2} \right) \tag{29}$$

So, we may conclude that:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet \overline{W}_i \tag{30}$$

which means that multiplying a normalized image with a non-normalized weight matrix dot multiplication is equal to the dot multiplication of the non – normalized image with the non-normalized weight matrix.

IV. Effect of Weight Normalization on the Speed up Ratio

Normalization of subimages in the spatial domain (in case of using traditional neural networks) requires $2n^2(N-n+1)^2$ computation steps. On the other hand, normalization of subimages in the frequency domain through normalizing the weights of the neural networks requires $2qn^2$ operations. This proves that local image normalization in the frequency domain is faster than that in the spatial one. By using weight normalization, the speed up ratio for image normalization Γ can be calculated as:

$$\Gamma = \frac{(N-n+1)^2}{q} \tag{31}$$

The speed up ratio of the normalization process for images of different sizes is listed in Table 6. As a result, we may conclude that:

1- Using this technique, normalization in the frequency domain can be done through normalizing the weights in spatial domain.
2- Normalization of an image through normalization of weights is faster than normalization of each subimage.
3- Normalization of weights can be done off line. So, the speed up ratio in the case of weight normalization can be calculated as follows:

a) For Conventional Neural Networks:

The speed up ratio equals the number of computation steps required by conventional neural networks with image normalization divided by the number of computation steps needed by conventional neural networks with weight normalization, which is done off line. The speed up ratio η_c in this case can be given by:

$$\eta_c = \frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{q(2n^2-1)(N-n+1)^2} \tag{32}$$

which can be simplified to:

$$\eta_c = 1 + \frac{2n^2}{q(2n^2 - 1)} \quad (33)$$

b) For Fast Neural Networks:

The over all speed up ratio equals the number of computation steps required by conventional neural networks with image normalization divided by the number of computation steps needed by fast neural networks with weight normalization, which is done off line. The over all speed up ratio η_o can be given by:

$$\eta_o = \frac{q(2n^2 - 1)(N - n + 1)^2 + 2n^2(N - n + 1)^2}{(2q + 1)(5N^2 \log_2 N^2) + q(8N^2 - n^2) + N} \quad (34)$$

which can be simplified to:

$$\eta_o = \frac{(N - n + 1)^2 (q(2n^2 - 1) + 2n^2)}{(2q + 1)(5N^2 \log_2 N^2) + q(8N^2 - n^2) + N} \quad (35)$$

The relation between the speed up ratio before (η) and after (η_o) the normalization process can be summed up as:

$$\eta_o = \eta + \frac{2n^2(N - n + 1)^2}{(2q + 1)(5N^2 \log_2 N^2) + q(8N^2 - n^2) + N} \quad (36)$$

The overall speed up ratio (Eq. 36) with images of different sizes and different sizes of windows is listed in Table 7. We can easily note that the speed up ratio in case of image normalization through weight normalization is larger than the speed up ratio (without normalization) listed in Table 1. This means that the search process with normalized faster neural networks is done faster than conventional neural networks with or without normalization of the input image. The overall practical speed up ratio (Eq. 36) after normalization of weights off line is listed in Table 8.

V. Conclusion

Normalized neural networks for fast pattern detection in a given image have been presented. It has been proved mathematically and practically that the speed of the detection process becomes faster than conventional neural networks. This has been accomplished by applying cross correlation in the frequency domain between the input image and the normalized input weights of the neural networks. Furthermore, a new general formulas for fast cross correlation as well as the speed up ratio have been given. Also, the problem of local subimage normalization in the frequency space has been solved. Moreover, it has been generally proved that the speed up ratio in the case of image normalization through normalization of weights is faster than subimage normalization in the spatial domain. This speed up ratio is faster than the one obtained without normalization. Simulation results have confirmed the theoretical computations by using *MATLAB*. The proposed approach can be applied to detect the presence/absence of any other object in an image.

References

[1] Hazem M. El-Bakry, "Fast Sub-Image Segmentation Using Parallel Neural Processors and Image Decomposition," Proc. of the SPIE Vol. 5960, the International Symposium on Visual Communication and Image Processing (VCIP), 12-15 July, 2005, Beijing, China**,** pp. 1712-1722.

[2] Hazem M. El-Bakry, "Comments on Using MLP and FFT for Fast Object/Face Detection," Proc. of IEEE IJCNN'03, Portland, Oregon, pp. 1284-1288, July, 20-24, 2003.

[3] Hazem M. El-Bakry, "Human Iris Detection Using Fast Cooperative Modular Neural Networks and Image Decomposition," Machine Graphics & Vision Journal (MG&V), vol. 11, no. 4, 2002, pp. 498-512.

[4] Hazem M. El-Bakry, "Face detection using fast neural networks and image decomposition," Neurocomputing Journal, vol. 48, 2002, pp. 1039-1046.

[5] S. Srisuk and W. Kurutach, "A New Robust Face Detection in Color Images", Proc. of IEEE Computer Society International Conference on Automatic Face and Gesture Recognition (AFGR'02), Washington D.C., USA, May 20-21, 2002, pp. 306-311.

[6] Hazem M. El-Bakry, "Automatic Human Face Recognition Using Modular Neural Networks," Machine Graphics & Vision Journal (MG&V), vol. 10, no. 1, 2001, pp. 47-73.

[7] Ying Zhu, Stuart Schwartz, and Michael Orchard, "Fast Face Detection Using Subspace Discriminate Wavelet Features," Proc. of IEEE Computer Society International Conference on Computer Vision and Pattern Recognition (CVPR'00)**,** South Carolina, June 13 - 15, 2000, vol.1, pp. 1636-1643.

[8] R. Feraud, O. Bernier, J. E. Viallet, and M. Collobert, "A Fast and Accurate Face Detector for Indexation of Face Images," Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition, Grenoble, France, 28-30 March, 2000.

[9] S. Ben-Yacoub, B. Fasel, and J. Luettin**,** "Fast Face Detection using MLP and FFT," in Proc. of the Second International Conference on Audio and Video-based Biometric Person Authentication (AVBPA'99)", 1999.

[10] S. Baluja, H. A. Rowley, and T. Kanade, "Neural Network - Based Face Detection," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 20, No. 1, pp. 23-38, 1998.

[11] Beat Fasel, "Fast Multi-Scale Face Detection," IDIAP-Com 98-04, 1998.

[12] S. Ben-Yacoub, "Fast Object Detection using MLP and FFT," IDIAP-RR 11, IDIAP, 1997.

[13] James W. Cooley and John W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput*. 19, 297–301 (1965).

[14] J.P. Lewis, "Fast Normalized Cross Correlation", Available from:<http://www.idiom.com/~zilla/ Papers/nvisionInterface/nip.html >

Table 1: The theoretical speed up ratio for images with different sizes.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	3.67	5.04	6.34
200x200	4.01	5.92	8.05
300x300	4.00	6.03	8.37
400x400	3.95	6.01	8.42
500x500	3.89	5.95	8.39
600x600	3.83	5.88	8.33
700x700	3.78	5.82	8.26
800x800	3.73	5.76	8.19
900x900	3.69	5.70	8.12
1000x1000	3.65	5.65	8.05

Table 2: Practical Speed up ratio for images with different sizes Using MATLAB ver 5.3.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	7.88	10.75	14.69
200x200	6.21	9.19	13.17
300x300	5.54	8.43	12.21
400x400	4.78	7.45	11.41
500x500	4.68	7.13	10.79
600x600	4.46	6.97	10.28
700x700	4.34	6.83	9.81
800x800	4.27	6.68	9.60
900x900	4.31	6.79	9.72
1000x1000	4.19	6.59	9.46

Table 3: A comparison between the number of multiplication steps required for conventional and faster neural networks to manipulate images with different sizes (n=20, q=30).

Image size	Conventional Neural Nets	Fast Neural Nets	Speed up ratio (η_m)
100x100	7.8732e+007	2.6117e+007	3.01
200x200	3.9313e+008	1.1911e+008	3.30
300x300	9.4753e+008	2.8726e+008	3.29
400x400	1.7419e+009	5.3498e+008	3.26
500x500	2.7763e+009	8.6537e+008	3.21
600x600	4.0507e+009	1.2808e+009	3.16
700x700	5.5651e+009	1.7832e+009	3.12
800x800	7.3195e+009	2.3742e+009	3.08
900x900	9.3139e+009	3.0552e+009	3.05
1000x1000	1.1548e+010	3.8275e+009	3.02

Table 4: The theoretical speed up ratio for the general fast cross correlation algorithm.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	5.59	8.73	11.95
200x200	4.89	7.64	10.75
300x300	4.56	7.12	10.16
400x400	4.35	6.80	9.68
500x500	4.20	6.56	9.37
600x600	4.08	6.38	9.13
700x700	4.00	6.24	8.94
800x800	3.92	6.12	8.77
900x900	3.85	6.02	8.63
1000x1000	3.79	5.93	8.51

Table 5: Simulation results of the speed up ratio for the general fast cross correlation compared with the MATLAB cross correlation function (xcorr2).

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	10.14	13.05	16.49
200x200	9.17	11.92	14.33
300x300	8.25	10.83	13.41
400x400	7.91	9.62	12.65
500x500	6.77	9.24	11.77
600x600	6.46	8.89	11.19
700x700	5.99	8.47	10.96
800x800	5.48	8.74	10.32
900x900	5.31	8.43	10.66
1000x1000	5.91	8.66	10.51

Table 6: The speed up ratio of the normalization process for images of different sizes (n=20,q=30).

Image size	Speed up ratio
100x100	219
200x200	1092
300x300	2632
400x400	4839
500x500	7712
600x600	11252
700x700	15459
800x800	20332
900x900	25872
1000x1000	32079

Table 7: Theoretical results for the speed up ratio in case of image normalization by normalizing the input weights.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	3.79	5.21	6.55
200x200	4.14	6.12	8.32
300x300	4.13	6.23	8.65
400x400	4.08	6.21	8.70
500x500	4.02	6.15	8.67
600x600	3.96	6.08	8.61
700x700	3.90	6.01	8.53
800x800	3.86	5.95	8.46
900x900	3.81	5.89	8.39
1000x1000	3.77	5.84	8.32

Table 8: Simulation results for the speed up ratio in case of image normalization by normalizing the input weights.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	8.91	12.03	16.74
200x200	7.43	10.42	15.39
300x300	6.72	9.72	14.45
400x400	5.99	8.61	13.59
500x500	5.75	8.32	12.94
600x600	5.61	8.09	11.52
700x700	5.49	7.97	11.04
800x800	5.41	7.83	10.74
900x900	5.32	7.71	10.56
1000x1000	5.29	7.58	10.45