International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:8, 2008

A New High Speed Neural Model for Fast Character Recognition Using Cross Correlation and Matrix Decomposition

Hazem M. El-Bakry

Abstract**—**Neural processors have shown good results for detecting a certain character in a given input matrix. In this paper, a new idead to speed up the operation of neural processors for character detection is presented. Such processors are designed based on cross correlation in the frequency domain between the input matrix and the weights of neural networks. This approach is developed to reduce the computation steps required by these faster neural networks for the searching process. The principle of divide and conquer strategy is applied through image decomposition. Each image is divided into small in size sub-images and then each one is tested separately by using a single faster neural processor. Furthermore, faster character detection is obtained by using parallel processing techniques to test the resulting sub-images at the same time using the same number of faster neural networks. In contrast to using only faster neural processors, the speed up ratio is increased with the size of the input image when using faster neural processors and image decomposition. Moreover, the problem of local subimage normalization in the frequency domain is solved. The effect of image normalization on the speed up ratio of character detection is discussed. Simulation results show that local subimage normalization through weight normalization is faster than subimage normalization in the spatial domain. The overall speed up ratio of the detection process is increased as the normalization of weights is done off line.

Keywords**—**Fast Character Detection, Neural Processors, Cross Correlation, Image Normalization, Parallel Processing.

I. INTRODUCTION

CHARACTER detection is a fundamental step before character recognition. Its reliability and performance have a major influence in a whole character recognition system. Nowadays, neural networks have shown very good results for detecting a certain pattern in a given image [2,4,6,8,9,10,12]. Among other techniques [3,5,7], neural networks are efficient pattern detectors [2,4,6,9].

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].

Manuscript received December 1, 2004.
H. M. El-Bakry is assistant lecturer with Faculty of Computer Science and Information Systems – Mansoura University – Egypt. Now, he is PhD student in University of Aizu, Aizu Wakamatsu City, Japan 965-8580 (phone: +81-242-37-2760, fax: +81-242-37-2743, e-mail: d8071106@u-aizu.ac.jp).

The main objective of this paper is to reduce the detection time using neural networks. Our idea is to fast the operation of neural networks by performing the testing process in the frequency domain instead of spatial domain. Then, cross correlation between the input image and the weights of neural networks is performed in the frequency domain. This model is called faster neural networks. Compared to conventional neural networks, faster neural networks show a significant reduction in the number of computation steps required to detect a certain character in a given image under test. Furthermore, another idea to increase the speed of these faster neural networks through image decomposition is presented. Moreover, the problem of subimage (local) normalization in the Fourier space which presented in [4] is solved. The number of computation steps required for weight normalization is proved to be less than that needed for image normalization. Also, 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, faster neural networks for character detection are described. The details of conventional neural networks, faster neural networks, and the speed up ratio of character detection are given. A faster searching algorithm for character detection which reduces the number of the required computation steps through image decomposition is presented in section III. Accelerating the new approach using parallel processing techniques is also introduced. Subimage normalization in the frequency domain through normalization of weights is introduced in section IV. The effect of weight normalization on the speed up ratio is presented in section V.

II. FAST CHARACTER DETECTION USING MLP AND FFT

Here, we are interested only in increasing the speed of neural networks during the test phase. By the words “Faster Neural Networks” we mean reducing the number of computation steps required by neural networks in the detection phase. First neural networks are trained to classify face from non face examples and this is done in the spatial domain. In the test phase, each sub-image in the input image (under test) is tested for the presence or absence of the required character . At each pixel position in the input image each sub-image is multiplied by a window of weights, which has the same size as the sub-image. This multiplication is done in the spatial domain. The outputs of neurons in the hidden layer are multiplied by the weights of the output layer. When the final output is high this means that the sub-image under test contains the required character and vice versa. Thus, we may conclude that this searching problem is cross correlation in the spatial domain between the image under test and the input weights of neural networks.

In this section, a fast algorithm for character detection based on two dimensional cross correlations that take place between the tested image and the sliding window (20x20 pixels) is described. Such window is represented by the neural network weights situated between the input unit and the hidden layer. 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^* (conjugate of 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 [6,8,9,10,11,12,13,14,15,16].

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 mxz and can be represented as mxn matrix. The output of hidden neurons $h(i)$ can be calculated as follows:

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

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:

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

Eq. (2) represents a cross correlation operation. Given any two functions f and g, their cross correlation can be obtained by [1]:

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

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

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

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:

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

(*) 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:

$$O(u,v) = g\left(\sum_{i=1}^{q} W_o(i)\, h_i(u,v) + b_o\right) \qquad (6)$$

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 $2D\text{-}FFT$ requires a number equal to $N^2\log_2N^2$ of complex computation steps. Also, the same number of complex computation steps is required for computing the $2D\text{-}FFT$ of the weight matrix for each neuron in the hidden layer.

2. At each neuron in the hidden layer, the inverse $2D\text{-}FFT$ 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 $2D\text{-}FFT$ to compute is $(2q+1)N^2\log_2N^2$.

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 $N^2\log_2N^2$ complex additions [20,21]. 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 $2D\text{-}FFT$ of an NxN image is:

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

which may be simplified to:

$$\rho=5N^2log_2N^2 \quad (8)$$

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. Assume that the input object/face has a size of (nxn) dimensions. So, the search process will be done over subimages of (nxn) dimensions and the weight matrix will have the same size. Therefore, 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 character 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 I. Practical speed up ratio for manipulating images of different sizes and different in size weight matrices is listed in Table II using 700 MHz processor and *MATLAB ver 5.3*. An interesting property with faster neural networks is that the number of computation steps does not depend on eith the size of the input subimage or the size of the weighth matrix (n). The effect of (n) on the the number of computation steps is very small and can be ignored. This is incontrast to conventional networks networks in which the number of computation steps is increased with the size of both the input subimage and the weight matrix (n).

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 relation between the number of multiplication steps required by conventional and faster neural networks:

$$\eta_m=\frac{qn^2(N-n+1)^2}{(2q+1)(3N^2log_2N^2)+6qN^2} \quad (12)$$

The results listed in Table III 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 (η_g) is in the following form:

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

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 (η_g) defined by Eq. (13) is shown in Table IV. Compared with *MATLAB* cross correlation function *(xcorr2),* experimental results show that the proposed algorithm is faster than this function as shown in Table V.

The authors in [17-19] 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 [22-42]. 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 [17-19] 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 $(q+1)N^2(log_2N)^2$ [17,19]. 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 [17,18].

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 [17]:

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

The speed up factor introduced in [17] 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 [17,18]. Also, this is not a typing error as the curve in Fig.2 in [17] realizes Eq.7, and the curves in Fig.15 in [18] realizes Eq.31 and Eq.32 in [18].

b) Also, the speed up ratio presented in [17] 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 [18] 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 [17,19].

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 [17-19] 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 [17-19] 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 [17-19] 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 is critical error in the activity of hidden neurons given in section 3.1 in [19] and also by Eq.(2) in [17]. Such activity given by those authors in [17,19] as follows:

$$h_i = g\left(\Psi \otimes W_i + b_i\right) \tag{15}$$

is not correct and should be written as Eq.(4) given here in this paper. This is because the fact that the operation of cross correlation is not commutative $(W\otimes\Psi \neq \Psi\otimes W)$. A practical example is shown in appendix ("A"). As a result, Eq.(15) (Eq.(2) in their paper [17]) does not give the same correct results as conventional neural networks. This error leads the researchers who consider the references [17,19] to think about how to modify the operation of cross correlation so that Eq.(15) (Eq.(2) in their paper [17]) can give the same correct results as conventional neural networks. Therefore, errors in these equations must be cleared to all the researchers. In [23-29], the authors proved that a symmetry condition must be found in input matrices (images and the weights of neural networks) so that fast 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. A practical proof for this achievement is explained by examples shown in appendix "A". As presented in [23-29], 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 [17-19] stated that the activity of each neuron in the hidden layer Eq. 16 (Eq.4 in their paper [17]) 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 [23-29] the two results will be the same, only when the two matrices are symmetric or at least the weight matrix is symmetric. A practical example which proves that for any two matrices the result of their cross correlation is different from their convolution unless that they are symmetric or at least the second matrix is symmetric as shown in appendix "B".

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 [17-19], 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:

$$a(x,y)=z(2x,2y) \tag{16}$$

$$Z(u,v)=FT(z(x,y)) \tag{17}$$

$$FT(a(x,y))=A(u,v)=\frac{1}{4}Z\left(\frac{u}{2},\frac{v}{2}\right) \tag{18}$$

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.18 is valid only for images shown in the form presented in Eq. 19. In which each block of pixels consists of 4 pixels located beside each other and have the same value as shown in Eq. 19. Certainly, there no guarantee that the input image will be in that form. Of course, it may have another form different from that one presented in Eq. 19.

$$\Psi=\begin{bmatrix} A\ A\ B\ B\ C\ C\ldots\ldots\ldots\ldots \\ A\ A\ B\ B\ C\ C\ldots\ldots\ldots\ldots \\ . \\ . \\ . \\ . \\ S\ S\ X\ X\ Y\ Y\ldots\ldots\ldots\ldots \\ S\ S\ X\ X\ Y\ Y\ldots\ldots\ldots\ldots \end{bmatrix} \tag{19}$$

In [17], 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 [17]:

$$\eta=\frac{qn^2}{(q+2)log^2N} \tag{20}$$

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.20 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:

$$\varphi=(2q+1)(5N^2log_2N^2)+(2q)5(N/2)^2log_2(N/2)^2 \tag{21}$$

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. A New Faster Algorithm for Character Detection based on Image Decomposition

In this section, a new faster algorithm for character detection is presented. The number of computation steps required for faster neural networks with different image sizes is listed in Tables VI and VII. From these tables, we may notice that as the image size is increased, the number of computation steps required by faster neural networks is much increased. For example, the number of computation steps required for an image of size (50x50 pixels) is much less than that needed for an image of size (100x100 pixels). Also, the number of computation steps required for an image of size (500x500 pixels) is much less than that needed for an image of size (1000x1000 pixels). As a result, for example, if an image of size (100x100 pixels) is decomposed into 4 sub-images of size (50x50 pixels) and each sub-image is tested separately, then a speed up factor for character detection can be achieved. The number of computation steps required by faster neural networks to test an image after decomposition can be calculated as follows:

1. Assume that the size of the image under test is (NxN pixels).
2. Such image is decomposed into α (*LxL* pixels) sub-images. So, α can be computed as:

$$\alpha=(N/L)^2 \tag{22}$$

3. Assume that, the number of computation steps required for testing one (*LxL* pixels) sub-image is β. So, the total number of computation steps (T) required for testing these sub-images resulting after the decomposition process is:

$$T=\alpha\beta \tag{23}$$

The speed up ratio in this case (η_d) can be computed as follows:

$$\eta_d = \frac{q(2n^2 - 1)(N - n + 1)^2}{(q(\alpha + 1) + \alpha)(5N_s^2 \log_2 N_s^2) + \alpha q(8N_s^2 - n^2) + N_s + \Delta} \tag{24}$$

where,

Ns: is the size of each small sub-image.

Δ: is a small number of computation steps required to obtain the results at the boundaries between subimages and depends on the size of the subimage.

To detect a character of size 20x20 pixels in an image of any size by using faster neural networks after image decomposition into sub-images, the optimal size of these sub-images must be computed. From Table VI, we may conclude that, the most suitable size for the sub-image which requires the smallest number of computation steps is 25x25 pixels. Also, the fastest speed up ratio can be achieved using this sub-image size (25x25) as shown in Figure 1. It is clear that the speed up ratio is reduced when the size of the sub-image *(L)* is increased. A comparison between the speed up ratio for faster neural networks and faster neural networks after image decomposition with different sizes of the tested images is listed in Tables VIII and IX. It is clear that the speed up ratio is increased with the size of the input image when using faster neural networks and image decomposition. This is in contrast to using only faster neural networks. As shown in Figure 2, the number of computation steps required by faster neural networks is increased rapidly with the size of the input image. Therefore the speed up ratio is decreased with the size of the input image. While in case of using faster neural networks and image decomposition, the number of computation steps required by faster neural networks is increased smoothly. Thus, the linearity of the computation steps required by faster neural networks in this case is better. As a result, the speed up ratio is increased. Increasing the speed up ratio with the size of the input image is considered an important achievement. Furthermore, for very large size matrices, while the speed up ratio for faster neural networks is decreased, the speed up ratio still increase in case of using faster neural networks and matrix decomposition as listed in Table X. Moreover, as shown in Figure 3, the speed up ratio in case of faster neural networks and image decomposition is increased with the size of the weight matrix which has the same size *(n)* as the input window. For example, it is clear that the speed up ratio is for window size of 30x30 is larger than that of size 20x20. Simulation results for the speed up ratio in case of using fast neural networks and image decomposition is listed in Table XI. It is clear that simulation results confirm the theoretical computations and the practical speed up ratio after image decomposition is faster than using only fast neural networks. In addition, the practical speed up ratio is increased with the size of the input image.

Also, to detect small in size matrices such as 5x5 or 10x10 using only faster neural networks, the speed ratio becomes less than one as shown in Tables XII,XIII,XIV, and XV. On the other hand, from the same tables it is clear that using fast neural and image decomposition, the speed up ratio becomes higher than one and increased with the dimensions of the input image. The dimensions of the new subimage after image decomposition *(L)* must not be less than the dimensions of the character which is required to be detected and has the same size as the weight matrix. Therefore, the following equation controls the relation between the subimage and the size of weight matrix (character to be detected) in order not to loss any information in the input image.

$$L \geq n \tag{25}$$

For example, in case of detecting 5x5 characters, the image must be decomposed into subimages of size not less than 5x5.

To further reduce the running time as well as increase the speed up ratio of the detection process, a parallel processing technique is used. Each sub-image is tested using a faster neural network simulated on a single processor or a separated node in a clustered system. The number of operations (ω) performed by each processor / node (sub-images tested by one processor/node) =

$$\omega = \frac{The\ total\ number\ of\ sub-images}{Number\ of\ Processors\ /\ nodes} \tag{26}$$

$$\omega = \frac{\alpha}{Pr} \tag{27}$$

where, *Pr* is the number of processors or nodes.

The total number of computation steps (γ) required to test an image by using this approach can be calculated as:

$$\gamma = \omega\beta \tag{28}$$

By using this algorithm, the speed up ratio in this case (η_{dp}) can be computed as follows:

$$\eta_{dp} = \frac{q(2n^2 - 1)(N - n + 1)^2}{ceil(((q(\alpha + 1) + \alpha)(5N_s^2 \log_2 N_s^2) + \alpha q(8N_s^2 - n^2) + N_s)/pr)} \tag{29}$$

where, *ceil(x)* is a *MATLAB* function rounds the elements of x to the nearest integers towards infinity.

As shown in Tables XVI and XVII, using a symmetric multiprocessing system with 16 parallel processors or 16 nodes in either a massively parallel processing system or a clustered system, the speed up ratio (with respect to conventional neural networks) for character detection is increased. A further reduction in the computation steps can be obtained by dividing each sub-image into groups. For each group, the neural operation (multiplication by weights and summation) is performed for each group by using a single processor. This operation is done for all of these groups as well as other groups in all of the sub-images at the same time. The best case is achieved when each group consists of only one element. In this case, one operation is needed for multiplication of the one element by its weight and also a small number of operations (ε) is required to obtain the over all summation for each sub-image. If the sub-image has n^2 elements, then the required number of processors will be n^2. As a result, the number of computation steps will be $\alpha q(1+\varepsilon)$, where ε is a small number depending on the value of n. For example, when $n=20$, then $\varepsilon=6$ and if $n=25$, then $\varepsilon=7$. The speed up ratio can be calculated as:

$$\eta=(2n^2-1)(N-n+1)^2/\alpha(1+\varepsilon) \tag{30}$$

Moreover, if the number of processors = αn^2, then the number of computation steps will be $q(1+\varepsilon)$, and the speed up ratio becomes:

$$\eta=(2n^2-1)(N-n+1)^2/\ (1+\varepsilon) \tag{31}$$

Furthermore, if the number of processors = $q\alpha n^2$, then the number of computation steps will be $(1+\varepsilon)$, and the speed up ratio can be calculated as:

$$\eta=q(2n^2-1)(N-n+1)^2/\ (1+\varepsilon) \tag{32}$$

In this case, as the length of each group is very small, then there is no need to apply cross correlation between the input image and the weights of the neural network in frequency domain.

IV. Subimage Centering and Normalization in the Frequency Domain

In [4], 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. In [17-19], the authors stated that centering and normalizing the image can be obtained by centering and normalizing the weights as follows [17-19]:

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

$$\overline{X}_{rc} = X_{rc} - \bar{x}_{rc} \tag{33}$$

where, $\overline{X}_{rc}$ is the mean value of the sub-image located at (r,c). We are interested in computing the cross correlation between the sub-image $\overline{X}_{rc}$ and the weights W_i that is:

$$\overline{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - \bar{x}_{rc} \otimes W_i \tag{34}$$

where,

$$\bar{x}_{rc} = \frac{X_{rc}}{n^2} \tag{35}$$

Combining (34) and (35), the following expression can be obtained:

$$\overline{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - \frac{X_{rc}}{n^2} \otimes W_i \tag{36}$$

which is the same as:

$$\overline{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - X_{rc} \otimes \frac{W_i}{n^2} \tag{37}$$

The centered zero mean weights are given by:

$$\overline{W}_i = W_i - \frac{W_i}{n^2} \tag{38}$$

Also, Eq. (37) can be written as:

$$\overline{X}_{rc} \otimes W_i = X_{rc} \otimes \left(W_i - \frac{W_i}{n^2} \right) \tag{39}$$

So, it can be concluded that:

$$\overline{X}_{rc} \otimes W_i = X_{rc} \otimes \overline{W}_i \tag{40}$$

which means that cross-correlating a normalized sub-image with the weight matrix is equal to the cross-correlation of the non – normalized sub-image with the normalized weight matrix [17-19]. However, this proof which presented in [17-19] is not correct at all because it is proved here mathematically and practically that cross-correlating a normalized sub-image with the weight matrix is not equal to the cross-correlation of the non – centered image with the normalized weight matrix

During the test phase, each sub-image in the input image is multiplied (dot multiplication) by the weight matrix and this operation is repeated for all possible sub-images in the input image. Repeating this process for all sub-images in the input image is equivalent to the cross correlation operation. Therefore, there is no cross correlation between each sub-image and the weight matrix. The cross correlation is done between the weight matrix and the whole input image. Thus, this proves that there is no need to the proof of Eq.(40) (presented in [17-19]) which is mathematically wrong. The result of Eq.(40) is correct only for the center value which equals to the dot product between the two matrices (sub-image and weight matrices). For all other values except the center value:

$$\overline{X}_{rc} \otimes W_i \neq X_{rc} \otimes \overline{W}_i \tag{41}$$

This fact is true for all types and values of matrices except symmetric matrices and our new technique of image decomposition presented in last section III. A practical example is given in appendix "C".

Furthermore, the definition of the mean value, Eq. (35) presented in [17-19] is not correct and must be :

$$\bar{x}_{rc} = \frac{\sum_{i,j=1}^{n} X_{rc}(i,j)}{n^2} \tag{42}$$

which makes the proof of Eq.(40) (presented in [17-19]) not correct.

Moreover, the operation performed between the weight matrix and each sub-image is dot multiplication. Our new idea is to normalize each sub-image in the frequency domain by normalizing the weight matrix. The dot product of two matrices is defined as follows:

$$X \bullet W = \sum_{i,j=1}^{n^2} X_{ij} W_{ij} \tag{43}$$

The result of dot product is only one value. We have also the following definitions:

$$1_{nxn} \bullet X = X \bullet 1_{nxn} = \sum_{i,j=1}^{n^2} X_{ij} \tag{44}$$

Where, 1_{nxn} is a nxn matrix where every element is 1.

$$1_{nxn} \bullet W = W \bullet 1_{nxn} = \sum_{i,j=1}^{n^2} W_{ij} \tag{45}$$

Lemma : $\bar{w} 1_{nxn} \bullet X = \bar{x} 1_{nxn} \bullet W$

Proof:

From Eqs. 42,43,44,and 45, we can conclude that:

$$\bar{w} 1_{nxn} \bullet X = \bar{w} \sum_{i,j=1}^{n^2} X_{ij} = \frac{1}{n^2} \sum_{i,j=1}^{n^2} W_{ij} \bullet \sum_{i,j=1}^{n^2} X_{ij} \tag{46}$$

Which can be reformulated as:

$$\bar{w} 1_{nxn} \bullet X = \frac{1}{n^2} \sum_{i,j=1}^{n^2} W_{ij} \bullet \sum_{i,j=1}^{n^2} X_{ij} \tag{47}$$

Also,

$$\bar{x} 1_{nxn} \bullet W = \bar{x} \sum_{i,j=1}^{n^2} W_{ij} = \frac{1}{n^2} \sum_{i,j=1}^{n^2} X_{ij} \bullet \sum_{i,j=1}^{n^2} W_{ij} \tag{48}$$

Which is the same as:

$$\bar{x} 1_{nxn} \bullet W = \frac{1}{n^2} \sum_{i,j=1}^{n^2} X_{ij} \bullet \sum_{i,j=1}^{n^2} W_{ij} \tag{49}$$

It is clear that Eq.(47) is the same as Eq.(49), which means:

$$\therefore \bar{w} 1_{nxn} \bullet X = \bar{x} 1_{nxn} \bullet W \tag{50}$$

Theorem:

$$\overline{X} \bullet W = \overline{W} \bullet X$$

Proof:

$$\overline{X} \bullet W = (X - \bar{x} 1_{nxn}) \bullet W$$

$$= X \bullet W - \bar{x} 1_{nxn} \bullet W$$

$$= X \bullet W - X \bullet 1_{nxn} \bar{w}$$

$$= X(W - \bar{w} \bullet 1_{nxn})$$

$$= X \bullet \overline{W}$$

So, we may conclude that:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet \overline{W}_i \tag{51}$$

which means that multiplying a normalized sub-image with a non-normalized weight matrix dot multiplication is equal to the dot multiplication between the non – normalized sub-image and the normalized weight matrix. The validation of Eq. (51) and a practical example is given in appendix "D".

As proved in our previous paper [8], the relation defined by Eq. (40) is true only for the resulting middle value. This is under two conditions. The first is to apply the technique of faster neural networks and image decomposition. In this case, the cross correlation is performed between each input sub-image and the weight matrix which has the same size as the resulting sub-image after image decomposition. The resulting middle value equals to the dot product between the input sub-image and the weight matrix (the value which we interested in). The second is that the required face/object is completely located in one of these sub-images (not between two sub-images). However applying cross correlation consumes more computation steps than applying dot product which makes Eq. (40) useful less.

V. Effect of Weight Normalization on the Speed up Ratio

Normalization of subimages in the spatial domain (in case of using traditional neural networks) requires $2n^2(N\text{-}n+1)^2$ 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:

$$\Gamma = \frac{(N-n+1)^2}{q} \tag{52}$$

The speed up ratio of the normalization process for images of different sizes is listed in Table XVIII. 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:

$$\eta_c = \frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{q(2n^2-1)(N-n+1)^2} \tag{53}$$

which can be simplified to:

$$\eta_c = 1 + \frac{2n^2}{q(2n^2-1)} \tag{54}$$

b) For Fasr 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:

$$\eta_o = \frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{(2q+1)(5N^2\log_2 N^2) + q(8N^2-n^2) + N} \tag{55}$$

which can be simplified to:

$$\eta_o = \frac{(N-n+1)^2\,(\,q(2n^2-1) + 2n^2\,)}{(2q+1)(5N^2\log_2 N^2) + q(8N^2-n^2) + N} \tag{56}$$

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

$$\eta_o = \eta + \frac{2n^2(N-n+1)^2}{(2q+1)(5N^2\log_2 N^2) + q(8N^2-n^2) + N} \tag{57}$$

The overall speed up ratio (Eq. 57) with images of different sizes and different sizes of windows is listed in Table XIX. 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 I. This means that the search process with normalized fast neural networks is done faster than conventional neural networks with or without normalization of the input image. The overall practical speed up ratio (Eq. 57) after normalization of weights off line is listed in Table XX.

VI. Conclusion

A novel high speed neural model for fast character 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. A new general formulas for fast cross correlation as well as the speed up ratio have been given. A faster neural network approach for character detection has been introduced. Such approach has decomposed the input image under test into many small in size sub-images. Furthermore, a simple algorithm for fast character detection based on cross correlations in the frequency domain between the sub-images and the weights of the neural net has been presented in order to speed up the execution time. Simulation results have shown that, using a parallel processing technique, large values of speed up ratio could be achieved. Moreover, by using faster neural networks and image decomposition, the speed up ratio has been increased with the size of the input image. Also, the problem of local subimage normalization in the frequency space has been solved. 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 theoretical computations by using MATLAB. The proposed approach can be applied to detect the presence/absence of any other object in an image.

APPENDIX "A"

AN EXAMPLE PROVES THAT THE CROSS CORRELATION BETWEEN ANY TWO MATRICES IS NOT COMMUTATIVE

Let $X = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}$, and $W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix}$

Then, the cross correlation between X and W can be obtained as follows:

$$W \otimes X = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 8\times5 & 8\times1+9\times5 & 9\times1 \\ 5\times5+8\times3 & 6\times5+5\times1+9\times3+8\times7 & 6\times1+9\times7 \\ 5\times3 & 6\times3+5\times7 & 6\times7 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 53 & 9 \\ 49 & 118 & 69 \\ 15 & 53 & 42 \end{bmatrix}$$

On the other hand, the cross correlation the cross correlation between W and X can be computed as follows:

$$X \otimes W = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} \otimes \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 7\times6 & 3\times6+7\times5 & 3\times5 \\ 1\times6+7\times9 & 5\times6+1\times5+3\times9+7\times8 & 5\times5+3\times8 \\ 1\times9 & 5\times9+1\times8 & 5\times8 \end{bmatrix}$$

$$= \begin{bmatrix} 42 & 53 & 15 \\ 69 & 118 & 49 \\ 9 & 53 & 40 \end{bmatrix}$$

which proves that $X \otimes W \neq W \otimes X$.

Also, when one of the two matrices is symmetric the cross correlation between the two matrices is non commutative as shown in the following example:

Let $X = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$, and $W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix}$

Then, the cross correlation between X and W can be obtained as follows:

$$X \otimes W = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} \otimes \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 5\times6 & 3\times6+5\times5 & 3\times5 \\ 3\times6+5\times9 & 5\times6+3\times5+3\times9+5\times8 & 5\times5+3\times8 \\ 3\times9 & 5\times9+3\times8 & 5\times8 \end{bmatrix}$$

$$= \begin{bmatrix} 30 & 43 & 15 \\ 63 & 112 & 49 \\ 27 & 69 & 40 \end{bmatrix}$$

On the other hand, the cross correlation the cross correlation between W and X can be computed as follows:

$$W \otimes X = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} 8\times5 & 8\times3+9\times5 & 9\times3 \\ 5\times5+8\times3 & 6\times5+5\times3+9\times3+8\times5 & 6\times3+9\times5 \\ 5\times3 & 6\times3+5\times5 & 6\times5 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 69 & 27 \\ 49 & 112 & 63 \\ 15 & 43 & 30 \end{bmatrix}$$

which proves that $X \otimes W \neq W \otimes X$.

The cross correlation between any two matrices is commutative only when the two matrices are symmetric as shown in the following example.

Let $X = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$, and $W = \begin{bmatrix} 8 & 9 \\ 9 & 8 \end{bmatrix}$

Then, the cross correlation between X and W can be obtained as follows:

$$X \otimes W = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} \otimes \begin{bmatrix} 8 & 9 \\ 9 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 5\times 8 & 9\times 5+8\times 3 & 9\times 3 \\ 9\times 5+8\times 3 & 8\times 5+9\times 3+9\times 3+8\times 5 & 8\times 3+9\times 5 \\ 9\times 3 & 8\times 3+9\times 5 & 8\times 5 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 69 & 27 \\ 69 & 122 & 69 \\ 27 & 69 & 40 \end{bmatrix}$$

On the other hand, the cross correlation between W and X can be computed as follows:

$$W \otimes X = \begin{bmatrix} 8 & 9 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} 8\times 5 & 9\times 5+8\times 3 & 9\times 3 \\ 9\times 5+8\times 3 & 8\times 5+9\times 3+9\times 3+8\times 5 & 9\times 5+8\times 3 \\ 9\times 3 & 5\times 9+3\times 8 & 8\times 5 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 69 & 27 \\ 69 & 122 & 69 \\ 27 & 69 & 40 \end{bmatrix}$$

which proves that the cross correlation is commutative ($X \otimes W = W \otimes X$) only under the condition when the two matrices X and W are symmetric.

APPENDIX "B"

AN EXAMPLE PROVES THAT THE CROSS CORRELATION BETWEEN ANY TWO MATRICES IS DIFFERENT FROM THEIR CONVOLUTION

$$Let \quad X = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}, \quad and \quad W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \quad ,$$

the result of their cross correlation can be computed as illustrated from the previous example (first result) in appendix "A". The convolution between W and X can be obtained as follows:

$$W \Diamond X = \begin{bmatrix} 8 & 9 \\ 5 & 6 \end{bmatrix} \Diamond \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 6\times 5 & 5\times 5+6\times 1 & 5\times 1 \\ 9\times 5+6\times 3 & 8\times 5+9\times 1+5\times 3+6\times 7 & 8\times 1+5\times 7 \\ 9\times 3 & 8\times 3+9\times 7 & 8\times 7 \end{bmatrix}$$

$$= \begin{bmatrix} 30 & 31 & 5 \\ 63 & 106 & 43 \\ 27 & 87 & 56 \end{bmatrix}$$

which proves that $W \otimes X \neq W \Diamond X$.

When the second matrix W is symmetric, the cross correlation between W and X can be computed as follows:

$$W \otimes X = \begin{bmatrix} 8 & 9 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 8\times 5 & 9\times 5+8\times 1 & 9\times 1 \\ 9\times 5+8\times 3 & 8\times 5+9\times 3+9\times 1+8\times 7 & 8\times 1+7\times 9 \\ 9\times 3 & 8\times 3+9\times 7 & 8\times 7 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 87 & 9 \\ 79 & 106 & 71 \\ 45 & 53 & 56 \end{bmatrix}$$

while the convolution can be between W and X can be obtained as follows:

$$W \Diamond X = \begin{bmatrix} 8 & 9 \\ 9 & 8 \end{bmatrix} \Diamond \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 8\times 5 & 9\times 5+8\times 1 & 9\times 1 \\ 9\times 5+8\times 3 & 8\times 5+9\times 3+9\times 1+8\times 7 & 8\times 1+7\times 9 \\ 9\times 3 & 8\times 3+9\times 7 & 8\times 7 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 87 & 9 \\ 79 & 106 & 71 \\ 45 & 53 & 56 \end{bmatrix}$$

which proves that under the condition that the second matrix is symmetric (or the two matrices are symmetric) the cross correlation between any the two matrices equals to their convolution.

APPENDIX "C"

A CROSS CORRELATION EXAMPLE BETWEEN A NORMALIZED MATRIX AND OTHER NON-NORMALIZED ONE AND VISE VERSA

$$\text{Let} \quad X = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} \quad , \text{and} \quad W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix}$$

Then the normalized matrices $\bar{X}$, and $\bar{W}$ can be computed as:

$$\bar{X} = \begin{bmatrix} 1 & -3 \\ -1 & 3 \end{bmatrix} \text{ , and } \bar{W} = \begin{bmatrix} -1 & -2 \\ 2 & 1 \end{bmatrix}$$

Now, the cross correlation between a normalized matrix and the other non-normalized one can be computed as follows:

$$\bar{X} \otimes W = \begin{bmatrix} 1 & -3 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} = \begin{bmatrix} 18 & 9 & -5 \\ 9 & 6 & -3 \\ -27 & -15 & 8 \end{bmatrix}$$

$$X \otimes \bar{W} = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} -1 & -2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} -7 & -17 & -6 \\ 13 & 6 & -7 \\ 2 & 11 & 5 \end{bmatrix}$$

which means that $\overline{X} \otimes W \neq X \otimes \overline{W}$.

However, the two results are equal only at the center element which equals to the dot product between the two matrices. The value of the center element *(2,2) =6* as shown above and also in appendix "D".

APPENDIX "D"

A DOT PRODUCT EXAMPLE BETWEEN A NORMALIZED MATRIX AND OTHER NON-NORMALIZED ONE AND VISE VERSA

This is to validate the correctness of Eq. (51). The left hand side of Eq. 51 can be expresseded as follows:

$$\overline{X} \bullet W = \begin{bmatrix} X_{1,1} - \overline{X} \ldots\ldots\ldots\ldots X_{1,n} - \overline{X} \\ . \\ . \\ . \\ . \\ X_{n,1} - \overline{X} \ldots\ldots\ldots\ldots X_{n,n} - \overline{X} \end{bmatrix} \bullet \begin{bmatrix} W_{1,1} \ldots\ldots\ldots\ldots\ldots W_{1,n} \\ .. \\ . \\ . \\ . \\ . \\ W_{n,1} \ldots\ldots\ldots\ldots\ldots W_{n,n} \end{bmatrix} \tag{58}$$

and also the right hand side of the same can be repressented as:

$$X \bullet \bar{W} = \begin{bmatrix} X_{1,1} \ldots\ldots\ldots\ldots\ldots X_{1,n} \\ .. \\ . \\ . \\ . \\ . \\ X_{n,1} \ldots\ldots\ldots\ldots\ldots X_{n,n} \end{bmatrix} \bullet \begin{bmatrix} W_{1,1} - \bar{W} \ldots\ldots\ldots\ldots W_{1,n} - \bar{W} \\ . \\ . \\ . \\ . \\ W_{n,1} - \bar{W} \ldots\ldots\ldots\ldots W_{n,n} - \bar{W} \end{bmatrix} \tag{59}$$

$\bar{X}$ and $\bar{W}$ are defined as follows:

$$\bar{X} = \frac{X_{1,1} + X_{1,2} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots + X_{n,n}}{n^2} \tag{60}$$

$$\bar{W} = \frac{W_{1,1} + W_{1,2} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots + W_{n,n}}{n^2}$$

By substituting from Eq.(60) in Eq.(58) and Eq.(59), then simplifying the results we can easily conclude that $\overline{X}_{rc} \bullet W_i = X_{rc} \bullet \overline{W}_i$.

Here is also a practical example:

Let $X = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}$, and $W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix}$

Then the normalized matrices $\bar{X}$, and $\bar{W}$ can be computed as:

$$\bar{X} = \begin{bmatrix} 1 & -3 \\ -1 & 3 \end{bmatrix} \text{ , and } \bar{W} = \begin{bmatrix} -1 & -2 \\ 2 & 1 \end{bmatrix}$$

Now, the dot product between a normalized matrix and the other non-normalized one can be performed as follows:

$$\bar{X} \bullet W = \begin{bmatrix} 1 & -3 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} = 6 - 15 - 9 + 24 = 6$$

$$X \bullet \bar{W} = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} -1 & -2 \\ 2 & 1 \end{bmatrix} = -5 - 2 + 6 + 7 = 6$$

which means generally that the dot product between a normalized matrix X and non-normalized matrix W equals to the dot product between the normalized matrix W and non-normalized matrix X. On the other hand, the cross correlation results are different as proved in appendix "C".

REFERENCES

[1] R. Klette, and Zamperon, Handbook of image processing operators, John Wiley & Sonsltd, 1996.

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

[3] H. Schneiderman and T. Kanade, "Probabilistic modeling of local appearance and spatial relationships for object recognition, " *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 45-51, SantaBarbara, CA, 1998.

[4] 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.

[5] Y. Zhu, S. Schwartz, and M. 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.

[6] H. 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] 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, Washington D.C., USA, May 20-21, 2002, pp. 306-311.

[8] H. M. El-Bakry, "Face detection using fast neural networks and image decomposition," *Neurocomputing Journal*, Vol. 48, 2002, pp. 1039-1046.

[9] H. 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.

[10] Hazem M. El-Bakry, and Qiangfu Zhao, "Fast Object/Face Detection Using Neural Networks and Fast Fourier Transform," *International Journal of Signal Processing*, Vol.1, No.3, pp. 182-187, 2004.

[11] Hazem M. El-Bakry, and Qiangfu Zhao, "A Modified Cross Correlation in the Frequency Domain for Fast Pattern Detection Using Neural Networks," *International Journal of Signal Processing*, Vol.1, No.3, pp. 188-194, 2004.

[12] Hazem M. El-Bakry, and Qiangfu Zhao, "Face Detection Using Fast Neural Processors and Image Decomposition," *International Journal of Computational Intelligence*, Vol.1, No.4, pp. 313-316, 2004.

[13] Hazem M. El-Bakry, and Qiangfu Zhao, "Fast Complex Valued Time Delay Neural Networks," *International Journal of Computational Intelligence*, Vol.2, No.1, pp. 16-26, 2005.

[14] Hazem M. El-Bakry, and Qiangfu Zhao, "A Fast Neural Algorithm for Serial Code Detection in a Stream of Sequential Data," *International Journal of Information Technology*, Vol.2, No.1, pp. 71-90, 2005.

[15] H. M. El-Bakry, and Q. Zhao, "Speeding-up Normalized Neural Networks For Face/Object Detection," *Machine Graphics & Vision Journal (MG&V)*, Vol. 14, No.1, 2005, pp. 29-59.

[16] H. M. El-Bakry, and Q. Zhao, "Fast Pattern Detection using Normalized Neural Networks and Cross Correlation in the Frequency Domain," *EURASIP Journal on Applied Signal Processing*,. Special Issue on Advances in Intelligent Vision Systems: Methods and Applications—Part I, Vol. 2005, No. 13, 1 August 2005, pp. 2054-2060.

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

[18] B. Fasel, "Fast Multi-Scale Face Detection," *IDIAP-Com 98-04*, 1998.

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

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

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

[22] H. 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.

[23] H. M. El-Bakry, and H. Stoyan, "Fast Neural Networks for Object/Face Detection," *Proc. of the 30th Anniversary SOFSEM Conference on Current Trends in Theory and Practice of Computer Science*, 24-30 January, 2004, Hotel VZ MERIN, Czech Republic.

[24] H. M. El-Bakry, and H. Stoyan, "Fast Neural Networks for Sub-Matrix (Object/Face) Detection," *Proc. of IEEE International Symposium on Circuits and Systems*, Vancouver, Canada, 23-26 May, 2004.

[25] H. M. El-Bakry, "Fast Sub-Image Detection Using Neural Networks and Cross Correlation in Frequency Domain," *Proc. of IS 2004: 14th Annual Canadian Conference on Intelligent Systems*, Ottawa, Ontario, 6-8 June, 2004.

[26] H. M. El-Bakry, and H. Stoyan, "Fast Neural Networks for Code Detection in a Stream of Sequential Data," *Proc. of CIC 2004 International Conference on Communications in Computing*, Las Vegas, Nevada, USA, 21-24 June, 2004.

[27] H. M. El-Bakry, "Fast Neural Networks for Object/Face Detection," *Proc. of 5th International Symposium on Soft Computing for Industry with Applications of Financial Engineering*, June 28 - July 4, 2004, Sevilla, Andalucia, Spain.

[28] H. M. El-Bakry, and H. Stoyan, "A Fast Searching Algorithm for Sub-Image (Object/Face) Detection Using Neural Networks," *Proc. of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, 18-21 July, 2004, Orlando, USA.

[29] H. M. El-Bakry, and H. Stoyan, "Fast Neural Networks for Code Detection in Sequential Data Using Neural Networks for Communication Applications," *Proc. of the First International Conference on Cybernetics and Information Technologies*, Systems and Applications: CITSA 2004, 21-25 July, 2004. Orlando, Florida, USA, Vol. IV, pp. 150-153.

[30] H. M. El-bakry, M. A. Abo-elsoud, and M. S. Kamel, "Fast Modular Neural Networks for Human Face Detection," *Proc. of IEEE-INNS-ENNS International Joint Conference on Neural Networks*, Como, Italy, Vol. III, pp. 320-324, 24-27 July, 2000.

[31] H. M. El-bakry, "Fast Iris Detection using Cooperative Modular Neural Nets," *Proc. of the 6th International Conference on Soft Computing*, 1-4 Oct., 2000, Japan.

[32] H. 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.

[33] H. M. El-bakry, "Fast Iris Detection Using Cooperative Modular Neural Networks," *Proc. of the 5th International Conference on Artificial Neural Nets and Genetic Algorithms*, pp. 201-204, 22-25 April, 2001, Prague, Czech Republic.

[34] H. M. El-bakry, "Fast Iris Detection Using Neural Nets," *Proc. of the 14th Canadian Conference on Electrical and Computer Engineering*, pp.1409-1415, 13-16 May, 2001, Canada.

[35] H. M. El-bakry, "Human Iris Detection Using Fast Cooperative Modular Neural Nets," *Proc. of INNS-IEEE International Joint Conference on Neural Networks*, pp. 577-582, 14-19 July, 2001, Washington, DC, USA.

[36] H. M. El-bakry, "Human Iris Detection for Information Security Using Fast Neural Nets," *Proc. of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, 22-25 July, 2001, Orlando, Florida, USA.

[37] H. M. El-bakry, "Human Iris Detection for Personal Identification Using Fast Modular Neural Nets," *Proc. of the 2001 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences,* pp. 112-118, 25-28 July, 2001, Monte Carlo Resort, Las Vegas, Nevada, USA.

[38] H. M. El-bakry, "Human Face Detection Using Fast Neural Networks and Image Decomposition," *Proc. the fifth International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, 6-8 September 2001, Osaka-kyoiku University, Kashiwara City, Japan, pp. 1330-1334.

[39] H. M. El-Bakry, "Fast Iris Detection for Personal Verification Using Modular Neural Networks," *Proc. of the International Conference on*

Computational Intelligence, 1-3 Oct., 2001, Dortmund, Germany, pp.-269-283.

[40] H. M. El-bakry, "Fast Cooperative Modular Neural Nets for Human Face Detection," *Proc. of IEEE International Conference on Image Processing*, 7-10 Oct., 2001, Thessaloniki, Greece.

[41] H. M. El-Bakry, "Fast Face Detection Using Neural Networks and Image Decomposition," *Proc. of the 6th International Computer Science Conference*, Active Media Technology, Dec. 18-20, 2001, Hong Kong – China, pp. 205-215, 2001.

[42] H. M. El-Bakry, "Face Detection Using Fast Neural Networks and Image Decomposition," *Proc. of INNS-IEEE International Joint Conference on Neural Networks*, 14-19 May, 2002, Honolulu, Hawaii, USA.

Eng. Hazem Mokhtar El-Bakry (Mansoura, EGYPT 20-9-1970) received B.Sc. degree in Electronics Engineering, and M.Sc. in Electrical Communication Engineering from the Faculty of Engineering, Mansoura University – Egypt, in 1992 and 1995 respectively. Since 1997, he has been an assistant lecturer at the Faculty of Computer Science and Information Systems – Mansoura University – Egypt. Currently, he is a doctoral student at the Multimedia device laboratory, University of Aizu - Japan. In 2004, he got a Research Scholarship from Japanese Government based on a recommendation from University of Aizu.

His research interests include neural networks, pattern recognition, image processing, biometrics, cooperative intelligent systems and electronic circuits. In these areas, he has published more than 44 papers as a single author in major international journals and conferences. He is the first author in 31 refereed international journal papers and more than 72 refereed international conference papers.

Eng. El-Bakry has the patent No. 2003E 19442 DE HOL / NUR, Magnetic Resonance, SIEMENS Company, Erlangen, Germany, 2003. Furthermore, he is associate editor for journal of computer science and network security (IJCSNS). In addition, he is a referee for IEEE Transactions on Signal Processing, Journal of Applied Soft Computing, the International Journal of Machine Graphics & Vision, the International Journal of Computer Science and Network Security, WASET Journals, WSEAS Journals and many different international conferences organized by IEEE. Moreover, he has been awarded the Japanese Computer & Communication prize in April 2006.

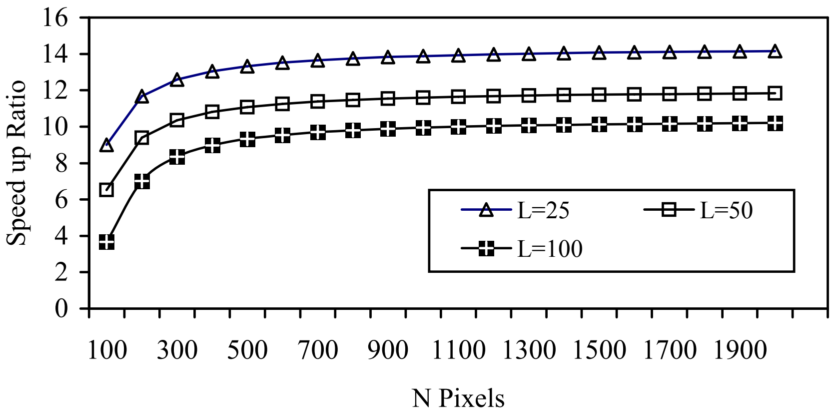

Fig. 1 The speed up ratio for images decomposed into different in size sub-images (L)

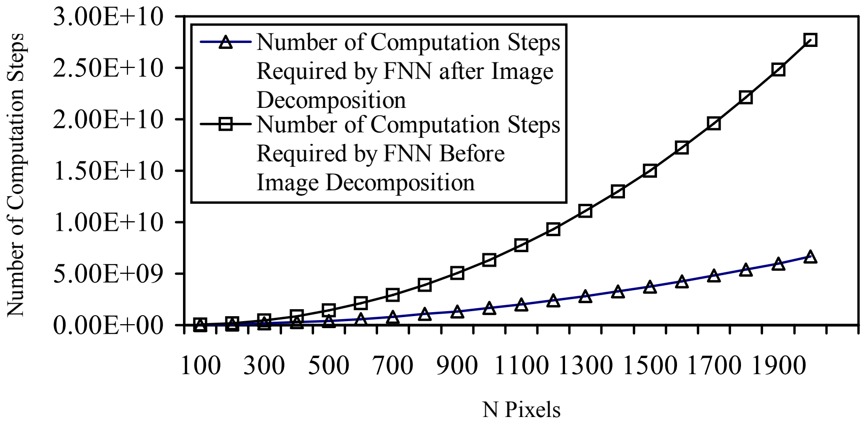

Fig. 2 A comparison between the number of computation steps required by FNN before and after Image decomposition

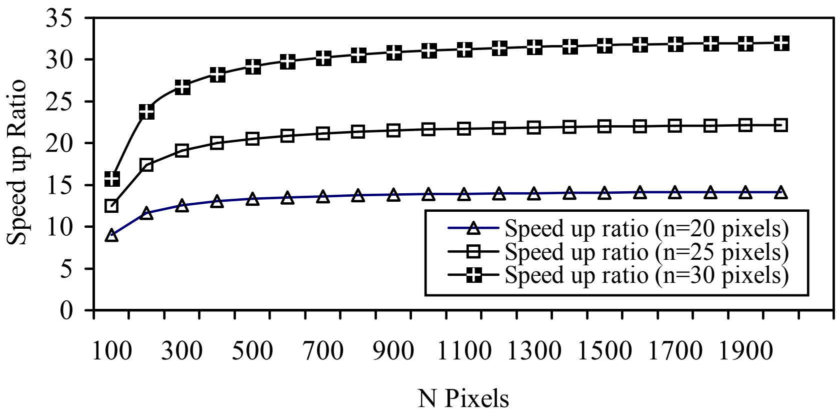

Fig. 3 The speed up ratio in case of image decomposition and different window size (n), (L=25x25)

TABLE I
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
1100x1100	3.62	5.60	7.99
1200x1200	3.58	5.55	7.93
1300x1300	3.55	5.51	7.93
1400x1400	3.53	5.47	7.82
1500x1500	3.50	5.43	7.77
1600x1600	3.48	5.43	7.72
1700x1700	3.45	5.37	7.68
1800x1800	3.43	5.34	7.64
1900x1900	3.41	5.31	7.60
2000x2000	3.40	5.28	7.56

TABLE II
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
1100x1100	4.24	6.66	9.62
1200x1200	4.20	6.62	9.57
1300x1300	4.17	6.57	9.53
1400x1400	4.13	6.53	9.49
1500x1500	4.10	6.49	9.45
1600x1600	4.07	6.45	9.41
1700x1700	4.03	6.41	9.37
1800x1800	4.00	6.38	9.32
1900x1900	3.97	6.35	9.28
2000x2000	3.94	6.31	9.25

TABLE III
A COMPARISON BETWEEN THE NUMBER OF MULTIPLICATION STEPS REQUIRED FOR CONVENTIONAL AND FASTER NEURAL NETS TO MANIPULATE IMAGES WITH DIFFERENT SIZES (n=20, q=30)

.Image size	Conventional Neural Nets	Faster Neural Nets	Speed up ratio (η_m)
100x100	7.8732e+007	2.6117e+007	3.0146
200x200	3.9313e+008	1.1911e+008	3.3007
300x300	9.4753e+008	2.8726e+008	3.2985
400x400	1.7419e+009	5.3498e+008	3.2560
500x500	2.7763e+009	8.6537e+008	3.2083
600x600	4.0507e+009	1.2808e+009	3.1627
700x700	5.5651e+009	1.7832e+009	3.1209
800x800	7.3195e+009	2.3742e+009	3.0830
900x900	9.3139e+009	3.0552e+009	3.0486
1000x1000	1.1548e+010	3.8275e+009	3.0172
1100x1100	1.4023e+010	4.6921e+009	2.9886
1200x1200	1.6737e+010	5.6502e+009	2.9622
1300x1300	1.9692e+010	6.7026e+009	2.9379
1400x1400	2.2886e+010	7.8501e+009	2.9154
1500x1500	2.6320e+010	9.0935e+009	2.8944
1600x1600	2.9995e+010	1.0434e+010	2.8748
1700x1700	3.3909e+010	1.1871e+010	2.8564
1800x1800	3.8064e+010	1.3407e+010	2.8392
1900x1900	4.2458e+010	1.5041e+010	2.8229
2000x2000	7.8732e+007	2.6117e+007	3.0146

TABLE IV
THE THEORETICAL SPEED UP RATIO FOR THE GENERAL FASTER 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	12.58
200x200	4.89	7.64	11.01
300x300	4.56	7.12	10.26
400x400	4.35	6.80	9.79
500x500	4.20	6.56	9.45
600x600	4.08	6.38	9.20
700x700	3.99	6.24	8.99
800x800	3.91	6.12	8.81
900x900	3.85	6.02	8.67
1000x1000	3.79	5.93	8.54
1100x1100	3.74	5.85	8.43
1200x1200	3.70	5.78	8.33
1300x1300	3.66	5.72	8.24
1400x1400	3.62	5.66	8.16
1500x1500	3.59	5.61	8.08
1600x1600	3.56	5.57	8.02
1700x1700	3.53	5.52	7.95
1800x1800	3.50	5.48	7.89
1900x1900	3.48	5.44	7.84
2000x2000	3.46	5.41	7.79

International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:8, 2008

TABLE V
SIMULATION RESULTS OF THE SPEED UP RATIO FOR THE GENERAL FASTER 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
1100x1100	5.77	8.61	10.46
1200x1200	5.68	8.56	10.40
1300x1300	5.62	8.52	10.35
1400x1400	5.58	8.47	10.31
1500x1500	5.54	8.43	10.26
1600x1600	5.50	8.39	10.22
1700x1700	5.46	8.33	10.18
1800x1800	5.42	8.28	10.14
1900x1900	5.38	8.24	10.10
2000x2000	5.34	8.20	10.06

TABLE VI
THE NUMBER OF COMPUTATION STEPS REQUIRED BY FASTER NEURAL NETWORKS (FNN) FOR IMAGES OF SIZES (25X25 - 1000X1000 PIXELS), q=30, n=20

Image size	No. of computation steps in case of using FNN
25x25	1.9085e+006
50x50	9.1949e+006
100x100	4.2916e+007
150x150	1.0460e+008
200x200	1.9610e+008
250x250	3.1868e+008
300x300	4.7335e+008
350x350	6.6091e+008
400x400	8.8203e+008
450x450	1.1373e+009
500x500	1.4273e+009
550x550	1.7524e+009
600x600	2.1130e+009
650x650	2.5096e+009
700x700	2.9426e+009
750x750	3.4121e+009
800x800	3.9186e+009
850x850	4.4622e+009
900x900	5.0434e+009
950x950	5.6623e+009
1000x1000	6.3191e+009

TABLE VII
THE NUMBER OF COMPUTATION STEPS REQUIRED BY FNN FOR IMAGES OF SIZES (1050X1050 - 2000X2000 PIXELS), q=30, n=20

Image size	No. of computation steps in case of using FNN
1050x1050	7.0142e+009
1100x1100	7.7476e+009
1150x1150	8.5197e+009
1200x1200	9.3306e+009
1250x1250	1.0180e+010
1300x1300	1.1070e+010
1350x1350	1.1998e+010
1400x1400	1.2966e+010
1450x1450	1.3973e+010
1500x1500	1.5021e+010
1550x1550	1.6108e+010
1600x1600	1.7236e+010
1650x1650	1.8404e+010
1700x1700	1.9612e+010
1750x1750	2.0861e+010
1800x1800	2.2150e+010
1850x1850	2.3480e+010
1900x1900	2.4851e+010
1950x1950	2.6263e+010
2000x2000	2.7716e+010
2050x2050	2.9211e+010

TABLE VIII
THE SPEED UP RATIO IN CASE OF USING FNN AND FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (25X25 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=50 TO N=1000, n=25, q=30)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
50x50	2.7568	5.0713
100x100	5.0439	12.4622
150x150	5.6873	15.6601
200x200	5.9190	17.3611
250x250	6.0055	18.4073
300x300	6.0301	19.1136
350x350	6.0254	19.6218
400x400	6.0059	20.0047
450x450	5.9790	20.3034
500x500	5.9483	20.5430
550x550	5.9160	20.7394
600x600	5.8833	20.9032
650x650	5.8509	21.0419
700x700	5.8191	21.1610
750x750	5.7881	21.2642
800x800	5.7581	21.3546
850x850	5.7292	21.4344
900x900	5.7013	21.5054
950x950	5.6744	21.5689
1000x1000	5.6484	21.6260

TABLE IX
THE SPEED UP RATIO IN CASE OF USING FNN AND FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (25X25 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=1050 TO N=2000, n=25, q=30)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
1050x1050	5.6234	21.6778
1100x1100	5.5994	21.7248
1150x1150	5.5762	21.7678
1200x1200	5.5538	21.8072
1250x1250	5.5322	21.8434
1300x1300	5.5113	21.8769
1350x1350	5.4912	21.9079
1400x1400	5.4717	21.9366
1450x1450	5.4528	21.9634
1500x1500	5.4345	21.9884
1550x1550	5.4168	22.0118
1600x1600	5.3996	22.0338
1650x1650	5.3830	22.0544
1700x1700	5.3668	22.0738
1750x1750	5.3511	22.0921
1800x1800	5.3358	22.1094
1850x1850	5.3209	22.1257
1900x1900	5.3064	22.1412
1950x1950	5.2923	22.1559
2000x2000	5.2786	22.1699

TABLE X
THE SPEED UP RATIO IN CASE OF USING FNN AND FNN AFTER MATRIX DECOMPOSITION INTO SUB-MATRICES (25X25 ELEMENTS) FOR VERY LARGE MATRICES (FROM N=100000 TO N=2000000, n=25, q=30)

Matrix size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after matrix decomposition
100000x100000	3.6109	22.7038
200000x200000	3.4112	22.7092
300000x300000	3.3041	22.7110
400000x400000	3.2320	22.7119
500000x500000	3.1783	22.7125
600000x600000	3.1357	22.7128
700000x700000	3.1005	22.7131
800000x800000	3.0707	22.7133
900000x900000	3.0448	22.7134
1000000x1000000	3.0221	22.7136
1100000x1100000	3.0018	22.7137
1200000x1200000	2.9835	22.7138
1300000x1300000	2.9668	22.7138
1400000x1400000	2.9516	22.7139
1500000x1500000	2.9376	22.7139
1600000x1600000	2.9245	22.7140
1700000x1700000	2.9124	22.7140
1800000x1800000	2.9011	22.7141
1900000x1900000	2.8904	22.7141
2000000x2000000	2.8804	22.7141

TABLE XI
THE PRACTICAL SPEED UP RATIO IN CASE OF USING FNN AND FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (25X25 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=100 TO N=2000, n=25, q=30)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
100x100	10.75	34.55
200x200	9.19	35.65
300x300	8.43	36.73
400x400	7.45	37.70
500x500	7.13	38.66
600x600	6.97	39.61
700x700	6.83	40.56
800x800	6.68	41.47
900x900	6.79	42.39
1000x1000	6.59	43.28
1100x1100	6.66	44.14
1200x1200	6.62	44.95
1300x1300	6.57	45.71
1400x1400	6.53	46.44
1500x1500	6.49	47.13
1600x1600	6.45	47.70
1700x1700	6.41	48.19
1800x1800	6.38	48.68
1900x1900	6.35	49.09
2000x2000	6.31	49.45

TABLE XII
THE SPEED UP RATIO IN CASE OF USING FNN AND FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (5X5 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=50 TO N=1000, n=5, q=30)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
50x50	0.3361	1.3282
100x100	0.3141	1.4543
150x150	0.2985	1.4965
200x200	0.2872	1.5177
250x250	0.2785	1.5303
300x300	0.2716	1.5388
350x350	0.2658	1.5448
400x400	0.2610	1.5493
450x450	0.2568	1.5529
500x500	0.2531	1.5557
550x550	0.2498	1.5580
600x600	0.2469	1.5599
650x650	0.2442	1.5615
700x700	0.2418	1.5629
750x750	0.2396	1.5641
800x800	0.2375	1.5652
850x850	0.2356	1.5661
900x900	0.2339	1.5669
950x950	0.2322	1.5677
1000x1000	0.2306	1.5683

TABLE XIII
THE SPEED UP RATIO IN CASE OF USING FNN AND FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (5X5 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=1050 TO N=2000, n=5, q=30)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
1050x1050	0.2292	1.5689
1100x1100	0.2278	1.5695
1150x1150	0.2265	1.5700
1200x1200	0.2253	1.5704
1250x1250	0.2241	1.5709
1300x1300	0.2230	1.5713
1350x1350	0.2219	1.5716
1400x1400	0.2209	1.5720
1450x1450	0.2199	1.5723
1500x1500	0.2189	1.5726
1550x1550	0.2180	1.5728
1600x1600	0.2172	1.5731
1650x1650	0.2163	1.5733
1700x1700	0.2155	1.5735
1750x1750	0.2148	1.5738
1800x1800	0.2140	1.5740
1850x1850	0.2133	1.5742
1900x1900	0.2126	1.5743
1950x1950	0.2119	1.5745
2000x2000	0.2112	1.5747

TABLE XIV
THE SPEED UP RATIO IN CASE OF USING FNN AND FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (5X5 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=50 TO N=1000, n=10, q=30)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
50x50	1.1202	3.1369
100x100	1.1503	3.9558
150x150	1.1303	4.2397
200x200	1.1063	4.3829
250x250	1.0842	4.4691
300x300	1.0647	4.5267
350x350	1.0474	4.5678
400x400	1.0321	4.5987
450x450	1.0185	4.6228
500x500	1.0063	4.6420
550x550	0.9952	4.6578
600x600	0.9851	4.6709
650x650	0.9758	4.6820
700x700	0.9672	4.6915
750x750	0.9593	4.6998
800x800	0.9519	4.7070
850x850	0.9451	4.7133
900x900	0.9386	4.7190
950x950	0.9325	4.7241
1000x1000	0.9268	4.7286

TABLE XV
THE SPEED UP RATIO IN CASE OF USING FNN AND FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (5X5 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=1050 TO N=2000, n=10, q=30)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
1050x1050	0.9214	4.7328
1100x1100	0.9163	4.7365
1150x1150	0.9114	4.7399
1200x1200	0.9068	4.7431
1250x1250	0.9023	4.7460
1300x1300	0.8981	4.7486
1350x1350	0.8941	4.7511
1400x1400	0.8902	4.7534
1450x1450	0.8865	4.7555
1500x1500	0.8829	4.7575
1550x1550	0.8795	4.7594
1600x1600	0.8762	4.7611
1650x1650	0.8730	4.7628
1700x1700	0.8699	4.7643
1750x1750	0.8669	4.7658
1800x1800	0.8640	4.7672
1850x1850	0.8613	4.7685
1900x1900	0.8586	4.7697
1950x1950	0.8559	4.7709
2000x2000	0.8534	4.7720

TABLE XVI
THE SPEED UP RATIO IN CASE OF USING FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (25X25 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=50 TO N=1000, n=25, q=30) USING 16 PARALLEL PROCESSORS OR 16 NODES

Image size	Speed up ratio
50x50	81.1403
100x100	199.3946
150x150	250.5611
200x200	277.7780
250x250	294.5171
300x300	305.8174
350x350	313.9482
400x400	320.0748
450x450	324.8552
500x500	328.6882
550x550	331.8296
600x600	334.4509
650x650	336.6712
700x700	338.5758
750x750	340.2276
800x800	341.6738
850x850	342.9504
900x900	344.0856
950x950	345.1017
1000x1000	346.0164

International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:8, 2008

TABLE XVII
THE SPEED UP RATIO IN CASE OF USING FNN AFTER IMAGE DECOMPOSITION INTO SUB-IMAGES (25X25 PIXELS) FOR IMAGES OF DIFFERENT SIZES (FROM N=1050 TO N=2000, n=25, q=30) USING 16 PARALLEL PROCESSORS OR 16 NODES

Image size	Speed up ratio
1050x1050	346.8442
1100x1100	347.5970
1150x1150	348.2844
1200x1200	348.9147
1250x1250	349.4946
1300x1300	350.0300
1350x1350	350.5258
1400x1400	350.9862
1450x1450	351.4150
1500x1500	351.8152
1550x1550	352.1896
1600x1600	352.5406
1650x1650	352.8704
1700x1700	353.1808
1750x1750	353.4735
1800x1800	353.7500
1850x1850	354.0115
1900x1900	354.2593
1950x1950	354.4943
2000x2000	354.7177

TABLE XVIII
THE SPEED UP RATIO OF THE NORMALIZATION PROCESS FOR IMAGES OF DIFFERENT SIZES (n =20, q =100)

Image size	Speed up ratio
100x100	62
200x200	328
300x300	790
400x400	1452
500x500	2314
600x600	3376
700x700	4638
800x800	6100
900x900	7762
1000x1000	9624
1100x1100	11686
1200x1200	13948
1300x1300	16410
1400x1400	19072
1500x1500	21934
1600x1600	24996
1700x1700	28258
1800x1800	31720
1900x1900	35382
2000x2000	39244

TABLE XIX
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	3.7869	5.2121	6.5532
200x200	4.1382	6.1165	8.3167
300x300	4.1320	6.2313	8.6531
400x400	4.0766	6.2063	8.7031
500x500	4.0152	6.1467	8.6684
600x600	3.9570	6.0796	8.6054
700x700	3.9039	6.0132	8.5334
800x800	3.8557	5.9502	8.4603
900x900	3.8120	5.8915	8.3891
1000x1000	3.7723	5.8369	8.3212
1100x1100	3.7360	5.7862	8.2568
1200x1200	3.7027	5.7391	8.1961
1300x1300	3.6719	5.6952	8.1389
1400x1400	3.6434	5.6542	8.0849
1500x1500	3.6169	5.6158	8.0340
1600x1600	3.5922	5.5798	7.9858
1700x1700	3.5690	5.5458	7.9403
1800x1800	3.5472	5.5138	7.8971
1900x1900	3.5266	5.4835	7.8560
2000x2000	3.5072	5.4547	7.8169

TABLE XX
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	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
1100x1100	5.41	7.83	10.81
1200x1200	5.36	7.77	10.76
1300x1300	5.32	7.71	10.71
1400x1400	5.28	7.65	10.66
1500x1500	5.24	7.60	10.62
1600x1600	5.21	7.56	10.58
1700x1700	5.18	7.52	10.52
1800x1800	5.14	7.48	10.47
1900x1900	5.11	7.44	10.43
2000x2000	5.08	7.41	10.38