Design of High Speed Neural Networks for Fast Pattern Detection by using Cross Correlation and Matrix Decomposition

Hazem M. El-Bakry
Faculty of Computer Science & Information Systems, Mansoura University, Egypt

1. Introduction

Fast pattern detection and identification is a fundamental problem for many applications of real-time systems (Bruce & Veloso 2003). 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 (Rowley et al. 1998; Feraud et al. 2000; Anifantis et al. 1999; Lang et al. 1988; El-Bakry 2001). Among other techniques (Schneiderman & Kanade 1998; Zhu et al. 2000; Srisuk & Kurutach 2002; Bao et al. 2006), neural networks are efficient pattern detectors (Rowley et al. 1998; Feraud et al. 2000; El-Bakry 2002,a; El-bakry 2002,b; Essannouni and Ibn Elhaj 2006; Roth et al. 2006; Ramasubramanian & Kannan 2006). 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 (Zhu et al. 2000; Srisuk & Kurutach 2002; Yang et al. 2002). The main objective of this paper is to reduce the detection time using neural networks. The idea is to accelerate 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 fast neural networks. Compared to conventional neural networks, fast neural networks show a significant reduction in the number of computation steps required to detect a certain pattern in a given image under test. Furthermore, another idea to increase the speed of these fast neural networks through image decomposition is presented. Moreover, the problem of sub-image (local) normalization in the Fourier space which presented in (Feraud et al. 2000) 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 sub-image normalization. Moreover, for neural networks, normalization of weights can be easily done off line before starting the search process.

In section 2, high speed neural networks for pattern detection are described. The details of conventional neural networks, high speed neural networks, and the speed up ratio of
pattern detection are given. A faster searching algorithm for pattern detection which reduces the number of the required computation steps through image decomposition is presented in section 3. Accelerating the new approach using parallel processing techniques is also introduced. Sub-image normalization in the frequency domain through normalization of weights is introduced in section 4. The effect of weight normalization on the speed up ratio is presented in section 5.

2. Fast pattern detection using MLP and FFT

Here, we are interested only in increasing the speed of neural networks during the test phase. By the words “High speed 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 face/object. 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 face/object 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 face/object 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 in the frequency domain point by point and then transform this product into spatial domain via the inverse Fourier transform (Klette&Zamperon 1996). 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 (El-Bakry 2005; El-Bakry 2006; El-Bakry 2007; El-Bakry 2009).

In the detection phase, a sub-image 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 sub-image 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:

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

(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 sub-image I. It can be computed for the whole image $\Psi$ as follows:

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

(2)
Eq.(2) represents a cross correlation operation. Given any two functions $f$ and $g$, their cross correlation can be obtained by (Gonzalez & Woods 2002):
\[
g(x,y) \ast f(x,y) = \sum_{m=-\infty}^{\infty} \sum_{z=-\infty}^{\infty} g(m,z)f(x+m,y+z) \tag{3}
\]

Therefore, Eq.(2) can be written as follows:
\[
h_i = g(W_i \ast \Psi + b_i) \tag{4}
\]

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 $\Psi$ and $(u,v) \in [P-m+1,T-n+1]$.

Now, the above cross correlation can be expressed in terms of the Fourier Transform:
\[
W_i \ast \Psi = F^{-1}\left(F(\Psi)F^*(W_i)\right) \tag{5}
\]

(*) 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:
\[
O(u,v) = g\left(\sum_{i=1}^{q} W_0(i)h_i(u,v)+b_0\right) \tag{6}
\]

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 $\Psi$. $W_o$ is the weight matrix between hidden and output layer. $b_o$ is the bias of the output neuron.

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

1. For a tested image of $N \times N$ pixels, the 2D-FFT requires a number equal to $N^2 \log_2 N^2$ 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 $(2q+1)N^2 \log_2 N^2$.

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 $(N^2/2) \log_2 N^2$ complex multiplications and $N^2 \log_2 N^2$ complex additions (Cooley&Tukey 1965). 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 $N \times N$ image is:
\[
\rho=6((N^2/2) \log_2 N^2) + 2(N^2 \log_2 N^2) \tag{7}
\]

which may be simplified to:
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 sub-images 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 high speed neural networks becomes:

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

which can be reformulated as:

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

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 \(\eta\) can be evaluated as follows:

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

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 2.7 GHz processor and MATLAB ver 5.3. An interesting property with high speed neural networks is that the number of computation steps does not depend on either the size of the input sub-image or the size of the weight matrix \((n)\). The effect of \((n)\) on the number of computation steps is very small and can be ignored. This is in contrast to conventional networks in which the number of computation steps is increased with the size of both the input sub-image 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 high speed neural networks. In order to clear this, the following equation \((\eta_m)\) describes relation between the number of multiplication steps required by conventional and high speed neural networks:

www.intechopen.com
### 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                   |
| 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 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                   |
| 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 3. 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 ($\eta_m$) |
|------------|--------------------------|-------------------|---------------------------|
| 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       | 2.9868                    |
| 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                    |

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 high speed 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 ($\eta_g$) is in the following form:

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

(13)

where $\tau$ 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 ($\eta_g$) defined by Eq.(13) is shown in Table 4. Compared with MATLAB cross correlation function (xcorr2), experimental results show that the proposed algorithm is high speed than this function as shown in Table 5.

(Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) 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 high speed neural networks. Then, they established an equation for the
### Table 4. 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                     |

### Table 5. 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                    |
speed up ratio. Unfortunately, these formulas contain many errors which lead to invalid speed up ratio. Recently, other authors developed their work based on these incorrect equations (Ishak et al. 2004). 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 waste their time and effort doing research based on wrong equations.

The authors (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) 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 $(q+1)N^2(\log_2N)^2$ (Ben-Yacoub et al. 1999; Ben-Yacoub 1997). 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 (Ben-Yacoub et al. 1999; Fasel 1998).

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 $\eta$ as follows (Fasel 1998; Ben-Yacoub 1997):

$$\eta = \frac{qn^2}{(q+1)\log_2^2 N} \quad (14)$$

The speed up factor introduced in (Ben-Yacoub et al. 1999) and given by Eq.14 is not correct for the following reasons:

a. The number of computation steps required for the 2D-FFT is $O(N^2\log_2N^2)$ and not $O(N^2\log^2 N)$ as presented in (Fasel 1998; Ben-Yacoub 1997). Also, this is not a typing error as the curve in Fig.2 in (Ben-Yacoub et al. 1999) realizes Eq.(7), and the curves in Fig.15 in (Fasel 1998) realizes Eq.(31) and Eq.(32) in (Fasel 1998).

b. Also, the speed up ratio presented in (Ben-Yacoub et al. 1999) not only contains an error but also is not precise. This is because for high speed 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 (Fasel 1998) 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 (Ben-Yacoub et al. 1999; Ben-Yacoub 1997).

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 (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) 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 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 100x1000 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 (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) for conventional neural networks is for real operations while that is required by the high speed neural networks is for complex operations. To obtain the speed up ratio, the authors in (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) have divided the two formulas directly without converting the number of computation steps required by the high speed neural networks into a real version.

f. Furthermore, there are critical errors in the activity of hidden neurons given in section 3.1 in (Ben-Yacoub 1997) and also by Eq.(2) in (Ben-Yacoub et al. 1999). Such activity given by those authors in (Ben-Yacoub et al. 1999; Ben-Yacoub 1997) as follows:

\[ h_i = g(\Psi \otimes W_i + b_i) \]  (15)

is not correct and should be written as Eq.(4) given here in this chapter. 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.(15) (Eq.(2) in their paper (Ben-Yacoub et al. 1999)) does not give the exact correct results as conventional neural networks. This error leads the researchers who consider the references (Ben-Yacoub et al. 1999; Ben-Yacoub 1997) to think about how to modify the operation of cross correlation so that Eq.(15) (Eq.(2) in their paper (Ben-Yacoub et al. 1999)) can give the exact correct results as conventional neural networks. Therefore, errors in these equations must be cleared to all the researchers. In (El-Bakry 2003), 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. As presented in (El-Bakry 2003), 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, fast 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 (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) stated that the activity of each neuron in the hidden layer Eq.(16) (Eq.(4) in their paper (Ben-Yacoub et al. 1999)) 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 (El-Bakry 2003) 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 “A”.

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 (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997), 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:

\[
a(x, y) = z(2x, 2y) \quad (16)
\]

\[
Z(u, v) = FT(z(x, y)) \quad (17)
\]

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

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.(16) 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).

\[
\Psi = \begin{bmatrix}
A & A & B & B & C & \ldots \\
A & A & B & B & C & \ldots \\
\vdots & \vdots & \ddots & \vdots \\
S & S & X & X & Y & \ldots \\
S & S & X & X & Y & \ldots 
\end{bmatrix} \quad (19)
\]

In (Ben-Yacoub et al. 1999), the author claimed that the processing needs \( O((q+2)N^2\log^2 N) \) additional number of computation steps. Thus the speed up ratio will be (Ben-Yacoub et al. 1999):

\[
\eta = \frac{qn^2}{(q + 2)\log^2 N} \quad (20)
\]

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 high speed neural networks will be:

\[
\varphi=(2q+1)(5N^2\log_2N^2)+(2q)5(N/2)^2\log_2(N/2)^2
\]

(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 high speed neural networks.

3. A new faster algorithm for pattern detection based on image decomposition

In this section, a new faster algorithm for face/object detection is presented. The number of computation steps required for faster neural networks with different image sizes is listed in Tables 6 and 7. From these tables, we may notice that as the image size is increased, the number of computation steps required by high speed 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 face/object detection can be achieved. The number of computation steps required by high speed neural networks to test an image after decomposition can be calculated as follows:

1. Assume that the size of the image under test is \((N\times N)\) pixels.
2. Such image is decomposed into \(\alpha\) \((L\times L)\) sub-images. So, \(\alpha\) can be computed as:

\[
\alpha=(N/L)^2
\]

(22)

3. Assume that the number of computation steps required for testing one \((L\times L)\) sub-image is \(\beta\). So, the total number of computation steps \((T)\) required for testing these sub-images resulting after the decomposition process is:

\[
T=\alpha\beta
\]

(23)

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

\[
\eta_d = \frac{q(2n^2-1)(N-n+1)^2}{(q(a+1)+a)\left[5N^2\log_2N^2\right]+aq\left[8N^2-n^2\right]+N^2+\Delta}
\]

where,

\(N_s\): is the size of each small sub-image.

\(\Delta\): is a small number of computation steps required to obtain the results at the boundaries between sub-images and depends on the size of the sub-image.

To detect a face/object of size 20x20 pixels in an image of any size by using high speed neural networks after image decomposition into sub-images, the optimal size of these sub-images must be computed. From Table 7, we may conclude that, the most suitable size for
| 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 6. 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 |
|----------------|---------------------------------------------|
| 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                                |

Table 7. The number of computation steps required by FNN for images of sizes (1050x1050 - 2000x2000 pixels), q=30, n=20
Design of High Speed Neural Networks for Fast Pattern Detection by using Cross Correlation and Matrix Decomposition

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.

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 high speed neural networks and high speed neural networks after image decomposition with different sizes of the tested images is listed in Tables 8 and 9. It is clear that the speed up ratio is increased with the size of the input image when using high speed neural networks and image decomposition. This is in contrast to using only high speed neural networks. As shown in Figure 2, the number of computation steps required by high speed 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 high speed neural networks and image decomposition, the
| 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 8. 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 |
|------------|-----------------------------------|----------------------------------------------------------|
| 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 9. 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)
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 10. 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) number of computation steps required by high speed neural networks is increased smoothly. Thus, the linearity of the computation steps required by high speed 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 high speed neural networks is decreased, the speed up ratio still increase in case of using high speed neural networks and matrix decomposition as listed in Table 10. Moreover, as shown in Figure 3, the speed up ratio in case of high speed 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 11. 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 high speed neural networks, the speed ratio becomes less than one as shown in Tables 12,13,14, and 15. 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 sub-image after image decomposition (L) must not be less than the dimensions of the face/object which is required to be detected and has the same size as the weight matrix. Therefore, the following equation controls the relation.
between the sub-image and the size of weight matrix (face/object to be detected) in order not to loss any information in the input image.

| 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 11. 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)

![Graph](image)

Fig. 3. The speed up ratio in case of image decomposition and different window size (n), (L=25x25).
### Table 12. 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 13. 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                                                      |
| 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 14. 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 |
|-----------|-----------------------------------|-------------------------------------------------------------|
| 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 15. 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)
Design of High Speed Neural Networks for Fast Pattern Detection
by using Cross Correlation and Matrix Decomposition

\[ L \geq n \] (25)

For example, in case of detecting 5x5 pattern, the image must be decomposed into sub-images 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 high speed neural network simulated on a single processor or a separated node in a clustered system. The number of operations \((\omega)\) performed by each processor / node (sub-images tested by one processor/node) =

\[ \omega = \frac{\text{The total number of sub-images}}{\text{Number of processors/nodes}} \] (26)

\[ \omega = \frac{\alpha}{P_r} \] (27)

where, \(P_r\) is the number of processors or nodes.

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

\[ \gamma = \omega \beta \] (28)

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

\[ \eta_{dp} = \frac{q(2n^2-1)(N-n+1)^2}{\text{ceil}((q(a+1) + a)(5N^2\log_2 N^2) + aq(8N^2 - n^2) + N^2)/pr)} \] (29)

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

As shown in Tables 16 and 17, 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 face/object 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 \((\epsilon)\) 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 to multiply each element in the sub-image matrix by the relevant element in the weight matrix; at the same time; will be \(n^2\). As a result, the number of computation steps will be \(\alpha q(1+\epsilon)\), where \(\epsilon\) is a small number depending on the value of \(n\). For example, when \(n=20\), then \(\epsilon=6\) and if \(n=25\), then \(\epsilon=7\). The speed up ratio can be calculated as:

\[ \eta = (2n^2-1)(N-n+1)^2/\alpha(1+\epsilon) \] (30)
### Table 16. 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       |
| 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   | 354.7127       |

### Table 17. 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  | 354.7127       |
Moreover, if the number of processors = $\alpha n^2$, then the number of computation steps will be $q(1+\epsilon)$, and the speed up ratio becomes:

$$\eta = (2n^2-1)(N-n+1)^2 / (1+\epsilon)$$  \hspace{1cm} (31)

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

$$\eta = q(2n^2-1)(N-n+1)^2 / (1+\epsilon)$$  \hspace{1cm} (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.

4. Sub-image centering and normalization in the frequency domain

(Feraud et al. 2000) 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 (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997), the authors stated that centering and normalizing the image can be obtained by centering and normalizing the weights as follows (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997):

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

$$\bar{X}_{rc} = X_{rc} - \bar{x}_{rc}$$  \hspace{1cm} (33)

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

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - \bar{x}_{rc} \otimes W_i$$  \hspace{1cm} (34)

where,

$$\bar{x}_{rc} = \frac{X_{rc}}{n^2}$$  \hspace{1cm} (35)

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

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - \frac{X_{rc}}{n^2} \otimes W_i$$  \hspace{1cm} (36)

which is the same as:

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - X_{rc} \otimes \frac{1}{n^2} \otimes W_i$$  \hspace{1cm} (37)

The centered zero mean weights are given by:
\[
\overline{W_i} = W_i - \frac{W_i}{n^2}
\]  

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

\[
\overline{X_{rc}} \otimes W_i = X_{rc} \otimes \left( W_i - \frac{W_i}{n^2} \right)
\]

(39)

So, it can be concluded that:

\[
\overline{X_{rc}} \otimes W_i = X_{rc} \otimes \overline{W_i}
\]

(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 (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) . However, this proof which presented in (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) 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 (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) ) 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:

\[
\overline{X_{rc}} \otimes W_i \neq X_{rc} \otimes \overline{W_i}
\]

(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 “B”.

Furthermore, the definition of the mean value, Eq. (35) presented in (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) is not correct and must be:

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

(42)

which makes the proof of Eq.(40) (presented in (Ben-Yacoub et al. 1999; Fasel 1998; and Ben-Yacoub 1997) ) 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:

\[
X \cdot W = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij} W_{ij}
\]  

(43)

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

\[
1_{nxn} \cdot X = X \cdot 1_{nxn} = \sum_{i,j=1}^{n^2} X_{ij}
\]  

(44)

where, \(1_{nxn}\) is a \(nxn\) matrix where every element is 1.

\[
l_{nxn} \cdot W = W \cdot l_{nxn} = \sum_{i,j=1}^{n^2} W_{ij}
\]  

(45)

Lemma: \(\bar{w}l_{nxn} \cdot X = xl_{nxn} \cdot W\)

Proof:

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

\[
\bar{w}l_{nxn} \cdot X = \frac{1}{n} \sum_{i,j=1}^{n^2} X_{ij} = \frac{1}{n^2} \sum_{i,j=1}^{n^2} W_{ij} \cdot \sum_{i,j=1}^{n^2} X_{ij}
\]  

(46)

which can be reformulated as:

\[
\bar{w}l_{nxn} \cdot X = \frac{1}{n^2} \sum_{i,j=1}^{n^2} W_{ij} \cdot \sum_{i,j=1}^{n^2} X_{ij}
\]  

(47)

also,

\[
xl_{nxn} \cdot W = x \sum_{i,j=1}^{n^2} W_{ij} = \frac{1}{n^2} \sum_{i,j=1}^{n^2} X_{ij} \cdot \sum_{i,j=1}^{n^2} W_{ij}
\]  

(48)

which is the same as:

\[
xl_{nxn} \cdot W = \frac{1}{n^2} \sum_{i,j=1}^{n^2} X_{ij} \cdot \sum_{i,j=1}^{n^2} W_{ij}
\]  

(49)

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

\[
\bar{w}l_{nxn} \cdot X = xl_{nxn} \cdot W
\]  

(50)
Theorem:

\[ \overline{X} \cdot W = \overline{W} \cdot X \]

Proof:

\[ \overline{X} \cdot W = (X - \overline{x}_{nxn}) \cdot W \]
\[ = X \cdot W - \overline{x}_{nxn} \cdot W \]
\[ = X \cdot W - X \cdot \overline{1}_{nxn} \cdot w \]
\[ = X(W - \overline{w} \cdot l_{nxn}) \]
\[ = X \cdot \overline{W} \]

So, we may conclude that:

\[ \overline{X_{rc}} \cdot W_i = X_{rc} \cdot \overline{W_i} \]  \hspace{1cm} (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 "C".

As proved in the previous paper (El-Bakry 2002,a), 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 high speed 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.

5. Effect of weight normalization on the speed up ratio

Normalization of sub-images 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 sub-images 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 \(\Gamma\) can be calculated as:

\[ \Gamma = \frac{(N-n+1)^2}{q} \]  \hspace{1cm} (52)

The speed up ratio of the normalization process for images of different sizes is listed in Table 18. 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.
Table 18. The speed up ratio of the normalization process for images of different sizes (n = 20, q = 100)

2. Normalization of an image through normalization of weights is faster than normalization of each sub-image.
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 \( \eta_c \) in this case can be given by:

\[
\eta_c = \frac{q(2n^2 - 1)(N - n + 1)^2 + 2n^2(N - n + 1)^2}{q(2n^2 - 1)(N - n + 1)^2}
\]

which can be simplified to:

\[
\eta_c = 1 + \frac{2n^2}{q(2n^2 - 1)}
\]

b) For High speed neural networks:

The overall speed up ratio equals the number of computation steps required by conventional neural networks with image normalization divided by the number of computation steps...
computation steps needed by high speed neural networks with weight normalization, which is done off line. The overall speed up ratio $\eta_o$ can be given by:

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

The relation between the speed up ratio before ($\eta$) and after ($\eta_o$) the normalization process can be summed up as:

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

The overall speed up ratio (Eq. (56)) with images of different sizes and different sizes of windows is listed in Table 19. 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 high speed neural networks is done faster than conventional neural networks with or without normalization of the input image. The overall practical speed up ratio (Eq. (56)) after normalization of weights off line is listed in Table 20.

| 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 19. Theoretical Results for the Speed up Ratio in case of Image Normalization by Normalizing the Input Weights
Design of High Speed Neural Networks for Fast Pattern Detection by using Cross Correlation and Matrix Decomposition

| 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                 |

Table 20. The theoretical speed up ratio for images with different sizes

6. 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 high speed 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 new high speed neural network approach for pattern 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 pattern 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. In addition, by using high speed neural networks and image decomposition, the speed up ratio has been increased with the size of the input image. Moreover, the problem of local sub-image 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 sub-image 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 Different from their Convolution

Let \( X = \begin{bmatrix} 5 \\ 3 \\ 7 \end{bmatrix} \), and \( W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \).

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

\[
W \otimes X = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 \times 5 & 8 \times 1 + 9 \times 5 \\ 5 \times 5 + 8 \times 3 & 6 \times 5 + 5 \times 1 + 9 \times 3 + 8 \times 7 \\ 5 \times 3 & 6 \times 3 + 5 \times 7 \\ 6 \times 7 \end{bmatrix} = \begin{bmatrix} 89 \\ 56 \end{bmatrix}
\]

The convolution between \( W \) and \( X \) can be obtained as follows:

\[
W \odot X = \begin{bmatrix} 8 \\ 5 \end{bmatrix} \odot \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \times 5 & 5 \times 5 + 6 \times 1 \\ 9 \times 5 + 6 \times 3 & 8 \times 5 + 9 \times 1 + 5 \times 3 + 6 \times 7 \\ 9 \times 3 & 8 \times 3 + 9 \times 7 \end{bmatrix} = \begin{bmatrix} 30 & 31 \\ 63 & 106 \\ 27 & 87 \end{bmatrix}
\]

which proves that \( W \otimes X \neq W \odot X \).

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

\[
W \otimes X = \begin{bmatrix} 8 \\ 9 \end{bmatrix} \otimes \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} = \begin{bmatrix} 8 \times 5 & 9 \times 5 + 8 \times 1 \\ 9 \times 5 + 8 \times 3 & 8 \times 5 + 9 \times 1 + 5 \times 3 + 9 \times 7 \\ 9 \times 3 & 8 \times 3 + 9 \times 7 \end{bmatrix} = \begin{bmatrix} 89 \\ 79 \\ 45 \end{bmatrix}
\]
while the convolution can be between $W$ and $X$ can be obtained as follows:

$$W \ast X = \begin{bmatrix} 8 & 9 \\ 9 & 8 \end{bmatrix} \ast \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} = \begin{bmatrix} 8 \times 5 & 9 \times 5 + 8 \times 1 & 9 \times 1 \\ 9 \times 5 + 8 \times 3 & 8 \times 5 + 9 \times 3 + 9 \times 1 + 8 \times 7 & 8 \times 1 + 7 \times 9 \\ 9 \times 3 & 8 \times 3 + 9 \times 7 & 8 \times 7 \end{bmatrix} = \begin{bmatrix} 40 & 87 & 9 \\ 79 & 106 & 71 \\ 45 & 53 & 56 \end{bmatrix}$$

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 “B”**

*A cross correlation Example between a normalized matrix and other non-normalized one and Vise versa*

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

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

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

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

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

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

which means that $\tilde{X} \otimes \tilde{W} \neq \tilde{X} \otimes \tilde{W}$. 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 “C”.

www.intechopen.com
Appendix “C”

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 expressed as follows:

$$
\overline{X} \cdot W = \begin{bmatrix}
X_{1,1} - \overline{X} & \cdots & X_{1,n} - \overline{X} \\
\vdots & \ddots & \vdots \\
X_{n,1} - \overline{X} & \cdots & X_{n,n} - \overline{X}
\end{bmatrix} \cdot \begin{bmatrix}
W_{1,1} & \cdots & W_{1,n} \\
\vdots & \ddots & \vdots \\
W_{n,1} & \cdots & W_{n,n}
\end{bmatrix}
$$

(57)

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

$$
X \cdot \overline{W} = \begin{bmatrix}
X_{1,1} & \cdots & X_{1,n} \\
\vdots & \ddots & \vdots \\
X_{n,1} & \cdots & X_{n,n}
\end{bmatrix} \cdot \begin{bmatrix}
W_{1,1} - \overline{W} & \cdots & W_{1,n} - \overline{W} \\
\vdots & \ddots & \vdots \\
W_{n,1} - \overline{W} & \cdots & W_{n,n} - \overline{W}
\end{bmatrix}
$$

(58)

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

$\overline{X} = \frac{X_{1,1} + X_{1,2} + \cdots + X_{n,n}}{n^2}$

$\overline{W} = \frac{W_{1,1} + W_{1,2} + \cdots + W_{n,n}}{n^2}$

(59)

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

Here is also a practical example:

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

Then the normalized matrices $\overline{X}$, and $\overline{W}$ can be computed as:
\[
X = \begin{bmatrix} 1 & -3 \\ -1 & 3 \end{bmatrix}, \quad \text{and} \quad W = \begin{bmatrix} -1 & -2 \\ 2 & 1 \end{bmatrix}
\]

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

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

\[
X \cdot \bar{W} = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} -1 & -2 \\ 2 & 1 \end{bmatrix} = -5 - 2 + 6 + 7 = 6
\]

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”.

7. References

Anifantis D., Dermatas E., Kokkinakis G. (1999) A neural network method for accurate face detection on arbitrary images. In: Proceedings of 6th IEEE international conference on electronics .Circuits and systems, Paphos, Cyprus, 5–8 September, pp. 109–112.

Bao J-P, Shen J-Y, Liu H-Y, Liu X-D (2006) A fast document copy detection model. Soft Comput J Fusion Found Methodol Appl, vol.10, no. 1, 2006, pp. 41–46.

Ben-Yacoub S. (1997) Fast object detection using MLP and FFT, IDIAP-RR 11, IDIAP.

Ben-Yacoub S., Fasel B., Luettin J. (1999) Fast face detection using MLP and FFT. In: Proceedings of the second international conference on audio and video-based biometric person authentication (AVBPA’99)

Bruce J., Veloso M. (2003) Fast and accurate vision-based pattern detection and identification. In: Proceedings of ICRA’03, the 2003, IEEE International Conference on Robotics and Automation, Taiwan, May 2003, pp. 1-6.

El-Bakry HM. (2002,a) Face detection using fast neural networks and image decomposition, Neurocomputing Journal, vol. 48, 2002, pp. 1039-1046.

El-Bakry HM. (2002,b) 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.

El-Bakry HM. (2003) Comments on Using MLP and FFT for Fast Object/Face Detection, Proc. of IEEE IJCNN’03, Portland, Oregon, 20-24 July, 2003, pp. 1284-1288.

El-Bakry HM. (2005) Human Face Detection Using New High Speed Modular Neural Networks, Lecture Notes in Computer Science, Springer, Vol. 3696, September 2005, pp. 543-550.

El-Bakry HM. (2006) New Fast Time Delay Neural Networks Using Cross Correlation Performed in the Frequency Domain, Neurocomputing Journal, Vol. 69, pp. 2360-2363.
El-Bakry HM. (2007) New Fast Principal Component Analysis for Face Detection, *International Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol.11, no.2, 2007, pp. 195-201.

El-Bakry HM. (2009) New Fast Principal Component Analysis for Real-Time Face Detection, *Machine Graphics & Vision Journal (MG&V)*, vol. 18, no. 4, pp. 405-426.

Essannouni L., Ibn Elhaj E. (2006) Face identification of video sequence. In: *Proceedings of 2006 the second European international symposium on communications, control and signal processing*, 13–15 March 2006, Marrakech, Morocco, 4 p

Fasel B. (1998) Fast multi-scale face detection, IDIAP-Com 98-04

Feraud R., Bernier O., Viallet JE. & Collobert M. (2000) A fast and accurate face detector for indexation of face images. In: *Proceedings of the fourth IEEE international conference on automatic face and gesture recognition*, Grenoble, France, pp. 28–30, March 2000.

Gonzalez RC. & Woods RE. (2002) *Digital image processing*. Prentice-Hall, USA

Ishak KA, Samad SA, Hussain A, Majlis BY (2004) A fast and robust face detection using neural networks. In: *Proceedings of the international symposium on information and communication technologies*. Multimedia University, Putrajaya, Malaysia, Vol 2, 7–8 October, pp 5–8

Cooley JW. & Tukey JW. (1965) An algorithm for the machine calculation of complex Fourier series, *Math. Comput.* 19, 297–301.

Klette R. & Zamperon P. (1996) Handbook of image processing operators. Wiley, New York

Lewis JP (1988) Fast Normalized Cross Correlation. Available from, http://www.idiom.com/*zilla/Papers/nvisionInterface/nip.html

Lang KJ, Hinton GE (1988) The development of time-delay neural network architecture for speech recognition, *Technical Report CMU-CS-88-152*. Carnegie-Mellon University, Pittsburgh, PA

Ramasubramanian P. & Kannan A. (2006) A genetic-algorithm based neural network short-term forecasting framework for database intrusion prediction system. Soft Comput J Fusion Found Methodol Appl, Vol. 10, No. 8, 2006, pp. 699-714.

Roth S., Gepperth A. & Igel C. (2006) Multi-objective neural network optimization for visual object detection, Vol 16. Springer, Berlin

Rowley HA., Baluja S. & Kanade T. (1998) Neural network-based face detection. *IEEE Trans Pattern Anal Mach Intell*, Vol. 20, No.1, 1998, pp.23–38.

Schneiderman H. & Kanade T. (1998) Probabilistic modeling of local appearance and spatial relationships for object recognition, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Santa Barbara, CA, pp 45–51.

Srisuk S. & Kurutach W. (2002) A new robust face detection in color images In: *Proceedings of IEEE Computer Society International Conference on Automatic Face and Gesture Recognition*, Washington DC, USA, 20–21 May, 2002, pp 306–311.

Yang M., Kriegman DJ. & Huja N. (2002) Detecting faces in images: a survey. *IEEE Trans Pattern Anal Mach Intell* 24(1):34–58.

Zhu Y., Schwartz S. & Orchard M. (2000) Fast face detection using subspace discriminate wavelet features, *Proceedings of IEEE Computer Society International Conference on Computer Vision and Pattern Recognition (CVPR’00)*, South Carolina, Vol. 1, 13–15 June, pp 1636–1643.
This book covers 27 articles in the applications of artificial neural networks (ANN) in various disciplines which includes business, chemical technology, computing, engineering, environmental science, science and nanotechnology. They modeled the ANN with verification in different areas. They demonstrated that the ANN is very useful model and the ANN could be applied in problem solving and machine learning. This book is suitable for all professionals and scientists in understanding how ANN is applied in various areas.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Hazem M. El-Bakry (2011). Design of High Speed Neural Networks for Fast Pattern Detection by using Cross Correlation and Matrix Decomposition, Artificial Neural Networks - Application, Dr. Chi Leung Patrick Hui (Ed.), ISBN: 978-953-307-188-6, InTech, Available from: http://www.intechopen.com/books/artificial-neural-networks-application/design-of-high-speed-neural-networks-for-fast-pattern-detection-by-using-cross-correlation-and-matri

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

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821
© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.