Volume 25, issue 11, 1 November 2007 ISSN 0262-8856
ELSEVIER
image AND vision COMPUTING
Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com
ScienceDirect
Image and Vision Computing 25 (2007) 1767–1784
image and vision COMPUTING
www.elsevier.com/locate/imavis

New fast normalized neural networks for pattern detection

Hazem M. El-Bakry [a,*], Nikos Mastorakis [b]

[a] *Faculty of Computer Science & Information Systems, Mansoura University, Egypt*
[b] *Department of Computer Science, Military Institutions of University Education (MIUE) – Hellenic Naval Academy, Greece*

Received 10 June 2005; received in revised form 2 June 2006; accepted 2 February 2007

Abstract

Neural networks have shown good results for detecting a certain pattern in a given image. In this paper, fast neural networks for pattern detection are presented. Such processors are designed based on cross correlation in the frequency domain between the input image and the input weights of neural networks. This approach is developed to reduce the computation steps required by these fast neural networks for the searching process. The principle of divide and conquer strategy is applied through image decomposition. Each image is divided into small in size sub-images and then each one is tested separately by using a single fast neural processor. Furthermore, faster pattern detection is obtained by using parallel processing techniques to test the resulting sub-images at the same time using the same number of fast neural networks. In contrast to fast neural networks, the speed up ratio is increased with the size of the input image when using fast neural networks and image decomposition. Moreover, the problem of local sub-image normalization in the frequency domain is solved. The effect of image normalization on the speed up ratio of pattern detection is discussed. Simulation results show that local sub-image normalization through weight normalization is faster than sub-image normalization in the spatial domain. The overall speed up ratio of the detection process is increased as the normalization of weights is done offline.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Fast pattern detection; Neural networks; Cross correlation; Image normalization; Parallel processing

1. Introduction

Pattern detection is a fundamental step before pattern recognition. Its reliability and performance have a major influence in a whole pattern recognition system. Nowadays, neural networks have shown very good results for detecting a certain pattern in a given image [2,4,6,8–10,12]. Among other techniques [3,5,7], neural networks are efficient pattern detectors [2,4,6,9]. But the problem with neural networks is that the computational complexity is very high because the networks have to process many small local windows in the images [5,7]. The main objective of this paper is to reduce the detection time using neural networks. Our idea is to fast the operation of neural networks by performing the testing process in the frequency domain instead of spatial domain. Then, cross correlation between the input image and the weights of neural networks is performed in the frequency domain. This model is called 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 [4] is solved. The number of computation steps required for weight normalization is proved to be less than that needed for image normalization. Also, the effect of weight normalization on the speed up ratio is theoretically and practically discussed. Mathematical calculations prove that the new idea of weight normalization, instead of image normalization, provides good results and increases the speed up ratio. This is because weight normalization requires fewer com-

* Corresponding author. Tel.: +2 050 2317356; fax: +2 050 2290631.
E-mail address: helbakry20@yahoo.com (H.M. El-Bakry).

0262-8856/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2007.02.001

putation 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, fast neural networks for pattern detection are described. The details of conventional neural networks, fast neural networks, and the speed up ratio of pattern detection are given. In addition, a clearer explanation more than those presented in [44–46] for the fast neural theory is 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 "Fast 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 (20×20 pixels) is described. Such window is represented by the neural network weights situated between the input unit and the hidden layer. The cross correlation theorem in mathematical analysis says that a cross correlation between f and 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. Compute the conjugate of H (H^*). Multiply F and H^* in the frequency domain point by point and then transform this product into spatial domain via the inverse Fourier transform [1]. As a result, these cross correlations can be represented by a product in the frequency domain. Thus, by using cross correlation in the frequency domain a speed up in an order of magnitude can be achieved during the detection process [6,8–16].

In the detection phase, a sub-image X of size $m \times n$ (sliding window) is extracted from the tested image, which has a size $P \times T$, and fed to the neural network as shown in Fig. 1. Let W_i be the vector of weights between the input sub-image and the hidden layer. This vector has a size of $m \times z$ and can be represented as $m \times n$ matrix. The output of hidden neurons h_i can be calculated as follows:

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

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 Ψ as follows:

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

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

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

Therefore, Eq. (2) can be written as follows [10–14]:

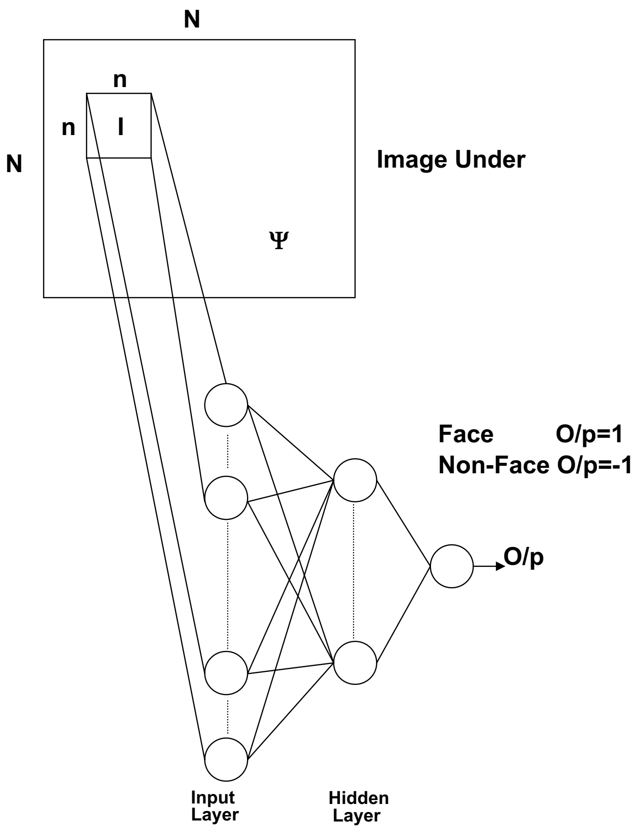

Fig. 1. Pattern detection using conventional neural networks.

$$h_i = g(W_i \otimes \Psi + b_i) \tag{4}$$

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 Ψ and $(u,v) \in [P-m+1, T-n+1]$.

Now, the above cross correlation can be expressed in terms of the Fourier Transform [1]:

$$W_i \otimes \Psi = F^{-1}(F(\Psi) \cdot F^*(W_i)) \tag{5}$$

$(^*)$ 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:

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

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 Ψ. W_o is the weight matrix between hidden and output layer.

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

1. For a tested image of $N \times N$ 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 fast 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 [20,21]. Every complex multiplication is realized by six real floating point operations and every complex addition is implemented by two real floating point operations. So, the total number of computation steps required to obtain the *2D-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:

$$\rho = 5N^2 \log_2 N^2 \tag{8}$$

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

5. In order to perform cross correlation in the frequency domain, the weight matrix must have the same size as the input image. Assume that the input object/face has a size of $(n \times n)$ dimensions. So, the search process will be done over sub-images of $(n \times n)$ 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 fast neural networks becomes:

$$\sigma = (2q+1)(5N^2\log_2 N^2) + 6qN^2 + q(N^2 - n^2) + qN^2 + N \tag{9}$$

which can be reformulated as:

$$\sigma = (2q+1)(5N^2\log_2 N^2) + q(8N^2 - n^2) + N \tag{10}$$

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

$$\eta = \frac{q(2n^2-1)(N-n+1)^2}{(2q+1)(5N^2\log_2 N^2) + q(8N^2 - n^2) + N} \tag{11}$$

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

The authors in [17–19] have proposed a multilayer perceptron (MLP) algorithm for fast face/object detection. The same authors claimed incorrect equation for cross correlation between the input image and the weights of the neural networks. They introduced formulas for the number of computation steps needed by conventional and fast neural networks. Then, they established an equation for the

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 ($n = 30$)
100 × 100	3.67	5.04	6.34
200 × 200	4.01	5.92	8.05
300 × 300	4.00	6.03	8.37
400 × 400	3.95	6.01	8.42
500 × 500	3.89	5.95	8.39
600 × 600	3.83	5.88	8.33
700 × 700	3.78	5.82	8.26
800 × 800	3.73	5.76	8.19
900 × 900	3.69	5.70	8.12
1000 × 1000	3.65	5.65	8.05
1100 × 1100	3.62	5.60	7.99
1200 × 1200	3.58	5.55	7.93
1300 × 1300	3.55	5.51	7.93
1400 × 1400	3.53	5.47	7.82
1500 × 1500	3.50	5.43	7.77
1600 × 1600	3.48	5.43	7.72
1700 × 1700	3.45	5.37	7.68
1800 × 1800	3.43	5.34	7.64
1900 × 1900	3.41	5.31	7.60
2000 × 2000	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 ($n = 30$)
100 × 100	7.88	10.75	14.69
200 × 200	6.21	9.19	13.17
300 × 300	5.54	8.43	12.21
400 × 400	4.78	7.45	11.41
500 × 500	4.68	7.13	10.79
600 × 600	4.46	6.97	10.28
700 × 700	4.34	6.83	9.81
800 × 800	4.27	6.68	9.60
900 × 900	4.31	6.79	9.72
1000 × 1000	4.19	6.59	9.46
1100 × 1100	4.24	6.66	9.62
1200 × 1200	4.20	6.62	9.57
1300 × 1300	4.17	6.57	9.53
1400 × 1400	4.13	6.53	9.49
1500 × 1500	4.10	6.49	9.45
1600 × 1600	4.07	6.45	9.41
1700 × 1700	4.03	6.41	9.37
1800 × 1800	4.00	6.38	9.32
1900 × 1900	3.97	6.35	9.28
2000 × 2000	3.94	6.31	9.25

speed up ratio. Unfortunately, these formulas contain many errors which lead to invalid speed up ratio. Other authors developed their work based on these incorrect equations [47]. So, the fact that these equations are not valid must be cleared to all researchers. It is not only very important but also urgent to notify other researchers not to do research based on wrong equations. Some of these mistakes were corrected in [22–46]. In this paper, complete corrections are given.

The authors in [17–19] analyzed their proposed fast neural network as follows: for a tested image of $N \times N$ pixels, the *2D-FFT* requires $O(N^2(\log_2 N)^2)$ computation steps. For the weight matrix W_i, the *2D-FFT* can be computed offline 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 $(q+1)N^2(\log_2 N)^2$ [17,19]. In addition, the input image and the weights should be multiplied in the frequency domain. Therefore, computation steps of (qN^2) should be added. This yields a total of $O((q+1)N^2(\log_2 N)^2 + qN^2)$ computation steps for the fast neural network [17,18].

Using sliding window of size $n \times n$, for the same image of $N \times N$ pixels, qN^2n^2 computation steps are required when using traditional neural networks for the face detection process. They evaluated theoretical speed up factor η as follows [17]:

$$\eta = \frac{qn^2}{(q+1)\log^2 N} \tag{12}$$

The speed up factor introduced in [17] and given by Eq. (14) is not correct for the following reasons:

(a) The number of computation steps required for the *2D-FFT* is $O(N^2\log_2 N^2)$ and not $O(N^2\log^2 N)$ as presented in [17,18]. Also, this is not a typing error as the curve in Fig. 2 in [17] realizes Eq. (7), and the curves in Fig. 15 of [18] realizes Eqs. (31) and (32) in [18].

(b) Also, the speed up ratio presented in [17] not only contains an error but also is not precise. This is because for fast neural networks, the term $(6qN^2)$ corresponds to complex dot product in the frequency domain must be added. Such term has a great effect on the speed up ratio. Adding only qN^2 as stated in [18] is not correct since a one complex multiplication requires six real computation steps.

(c) For conventional neural networks, the number of operations is $(q(2n^2-1)(N-n+1)^2)$ and not (qN^2n^2). The term n^2 is required for multiplication of n^2 elements (in the input window) by n^2 weights which results in another new n^2 elements. Adding these n^2 elements, requires another (n^2-1) steps. So, the total computation steps needed for each window is $(2n^2-1)$. The search operation for a face in the input image uses a window with $n \times n$ weights. This operation is done at each pixel in the input image. Therefore, such process is repeated $(N-n+1)^2$ times and not N^2 as stated in [17,19].

(d) Before applying cross correlation, the *2D-FFT* of the weight matrix must be computed. Because of the dot product, which is done in the frequency domain, the size of weight matrix should be increased to be the same as the size of the input image. Computing

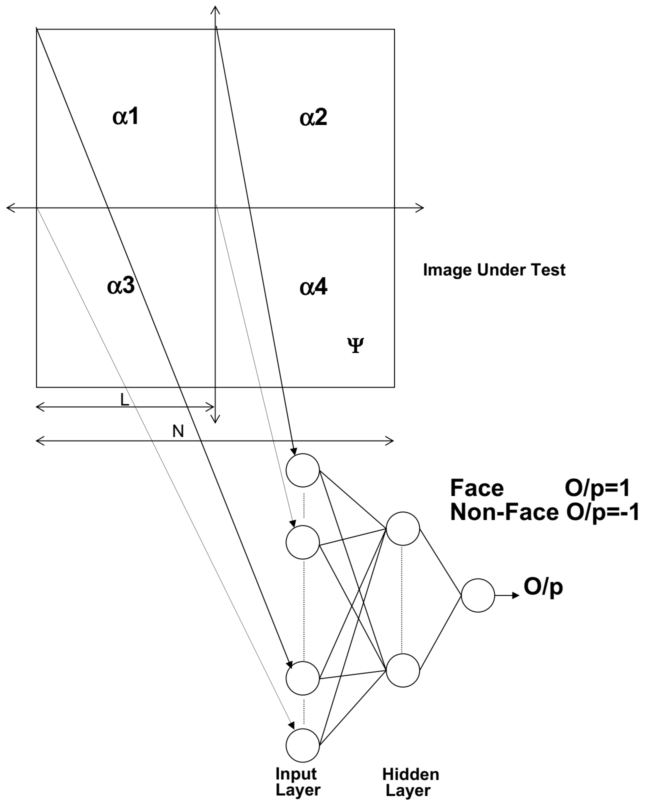

Fig. 2. Image decomposition into four sub-images and testing each sub-image separately.

the *2D-FFT* of the weight matrix off line as stated in [17–19] is not practical. In this case, all of the input images must have the same size. As a result, the input image will have only a one fixed size. This means that, the testing time for an image of size 50×50 pixels will be the same as that image of size 1000×1000 pixels and of course, this is unreliable.

(e) It is not valid to compare number of complex computation steps by another of real computation steps directly. The number of computation steps given by pervious authors [17–19] for conventional neural networks is for real operations while that is required by fast neural networks is for complex operations. To obtain the speed up ratio, the authors in [17–19] have divided the two formulas directly without converting the number of computation steps required by fast neural networks into a real version.

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

$$h_i = g(\Psi \otimes W_i + b_i) \tag{13}$$

is not correct and should be written as Eq. (4) given here in this paper. This is because the fact that the operation of cross correlation is not commutative ($W \otimes \Psi \neq \Psi \otimes W$). As a result, Eq. (13) (Eq. (4) of [17]) does not give the same correct results as conventional neural networks. This error leads the researchers who consider the references [17–19] to think about how to modify the operation of cross correlation so that Eq. (13) (Eq. (4) of [17]) can give the same correct results as conventional neural networks. Therefore, errors in these equations must be cleared to all the researchers. In [23–30], the authors proved that a symmetry condition must be found in input matrices (images and the weights of neural networks) so that fast neural networks can give the same results as conventional neural networks. In case of symmetry $W \otimes \Psi = \Psi \otimes W$, the cross correlation becomes commutative and this is a valuable achievement. In this case, the cross correlation is performed without any constrains on the arrangement of matrices. A practical proof for this achievement is explained by examples shown in Appendix A. As presented in [24–30], 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 of [17–19] stated that the activity of each neuron in the hidden layer defined by Eq. (13) (Eq. (4) of [17]) can be expressed in terms of convolution between a bank of filter (weights) and the input image. This is not correct because the activity of the hidden neuron is a cross correlation between the input image and the weight matrix. It is known that the result of cross correlation between any two functions is different from their convolution. As we proved in [24–30] the two results will be the same, only when the two matrices are symmetric or at least the weight matrix is symmetric. A practical example which proves that for any two matrices the result of their cross correlation is different from their convolution unless that they are symmetric or at least the second matrix is symmetric as shown in Appendix B.

(h) Images are tested for the presence of a face (object) at different scales by building a pyramid of the input image which generates a set of images at different resolutions. The face detector is then applied at each resolution and this process takes much more time as the number of processing steps will be increased. In [17–19], the authors stated that the Fourier transforms of the new scales do not need to be computed. This is due to a property of the Fourier transform. If $z(x, y)$ is the original and $a(x, y)$ is the sub-sampled by a factor of 2 in each direction image then:

$$a(x, y) = z(2x, 2y) \tag{14}$$

$$Z(u, v) = \mathrm{FT}(z(x, y)) \tag{15}$$

$$\mathrm{FT}(a(x, y)) = A(u, v) = \frac{1}{4} Z\left(\frac{u}{2}, \frac{v}{2}\right) \tag{16}$$

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 in the following form:

$$\Psi = \begin{bmatrix} A & A & B & B & C & C & \dots \\ A & A & B & B & C & C & \dots \\ \vdots & & & & & & \\ S & S & X & X & Y & Y & \dots \\ S & S & X & X & Y & Y & \dots \end{bmatrix} \tag{17}$$

In [17], the authors 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 [17]:

$$\eta = \frac{qn^2}{(q+2)\log^2 N} \tag{18}$$

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. (18) 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 fast neural networks will be:

$$\varphi = (2q+1)(5N^2\log_2 N^2) + (2q)5(N/2)^2\log_2(N/2)^2 \tag{19}$$

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 fast 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 fast neural networks with different image sizes is listed in Tables 3 and 4. From these tables, we may notice that as the image size is increased, the number of computation steps required by fast neural networks is much increased. For example, the number of computation steps required for an image of size (50 × 50 pixels) is much less than that needed for an image of size (100 × 100 pixels). Also, the number of computation steps required for an image of size (500 × 500 pixels) is much less than that needed for an image of size (1000 × 1000 pixels). As a result, for example, if an image of size (100 × 100 pixels) is decomposed into 4 sub-images of size (50 × 50 pixels) and each sub-image is tested separately as shown in Fig. 2, then a speed up factor for face/object detection can be achieved. The number of computation steps required by fast neural networks to test an image after decomposition can be calculated as follows:

Table 3
The number of computation steps required by fast neural networks (FNN) for images of sizes (25 × 25–1000 × 1000 pixels), $q = 30$, $n = 20$

Image size	No. of computation steps in case of using FNN
25 × 25	1.9085e+006
50 × 50	9.1949e+006
100 × 100	4.2916e+007
150 × 150	1.0460e+008
200 × 200	1.9610e+008
250 × 250	3.1868e+008
300 × 300	4.7335e+008
350 × 350	6.6091e+008
400 × 400	8.8203e+008
450 × 450	1.1373e+009
500 × 500	1.4273e+009
550 × 550	1.7524e+009
600 × 600	2.1130e+009
650 × 650	2.5096e+009
700 × 700	2.9426e+009
750 × 750	3.4121e+009
800 × 800	3.9186e+009
850 × 850	4.4622e+009
900 × 900	5.0434e+009
950 × 950	5.6623e+009
1000 × 1000	6.3191e+009

Table 4
The number of computation steps required by FNN for images of sizes (1050 × 1050–2000 × 2000 pixels), $q = 30$, $n = 20$

Image size	No. of computation steps in case of using FNN
1050 × 1050	7.0142e+009
1100 × 1100	7.7476e+009
1150 × 1150	8.5197e+009
1200 × 1200	9.3306e+009
1250 × 1250	1.0180e+010
1300 × 1300	1.1070e+010
1350 × 1350	1.1998e+010
1400 × 1400	1.2966e+010
1450 × 1450	1.3973e+010
1500 × 1500	1.5021e+010
1550 × 1550	1.6108e+010
1600 × 1600	1.7236e+010
1650 × 1650	1.8404e+010
1700 × 1700	1.9612e+010
1750 × 1750	2.0861e+010
1800 × 1800	2.2150e+010
1850 × 1850	2.3480e+010
1900 × 1900	2.4851e+010
1950 × 1950	2.6263e+010
2000 × 2000	2.7716e+010
2050 × 2050	2.9211e+010

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

$$\alpha = (N/L)^2 \tag{20}$$

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

$$T = \alpha\beta \tag{21}$$

The speed up ratio in this case (η_{d}) can be computed as follows:

$$\eta_{\mathrm{d}} = \frac{q(2n^2-1)(N-n+1)^2}{(q(\alpha+1)+\alpha)(5N_s^2\log_2 N_s^2)+\alpha q(8N_s^2-n^2)+N_s^2+\Delta} \tag{22}$$

where, N_s, is the size of each small sub-image.

Δ, is a small number of computation steps required to obtain the results at the boundaries between sub-images and depends on the size of the sub-image.

The results of the detection before image decomposition (presented in Section 2) and after image decomposition are the same. A practical example which proves that the results of cross correlation before and after the decomposition are the same is listed in Appendix C. To detect a face/object of size 20×20 pixels in an image of any size by using fast neural networks after image decomposition into sub-images, the optimal size of these sub-images must be computed. From Table 3, we may conclude that, the most suitable size for the sub-image which requires the smallest number of computation steps is 25×25 pixels. Also, the fastest speed up ratio can be achieved using this sub-image size (25×25) as shown in Fig. 3. 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 fast neural networks and fast neural networks after image decomposition with different sizes of the tested images is listed in Tables 5 and 6. It is clear that the speed up ratio is increased with the size of the input image when using fast neural networks and image decomposition. This is in contrast to using only fast neural networks. As shown in Fig. 4, the number of computation steps required by fast 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 fast neural networks and image decomposition, the number of computation steps required by fast neural networks is

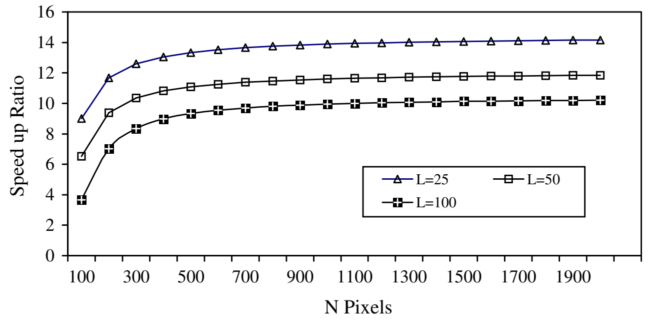

Fig. 3. The speed up ratio for images decomposed into different in size sub-images (L).

Table 5
The speed up ratio in case of using FNN and FNN after image decomposition into sub-images (25×25 pixels) for images of different sizes (from $N = 50$ to 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
50×50	2.7568	5.0713
100×100	5.0439	12.4622
150×150	5.6873	15.6601
200×200	5.9190	17.3611
250×250	6.0055	18.4073
300×300	6.0301	19.1136
350×350	6.0254	19.6218
400×400	6.0059	20.0047
450×450	5.9790	20.3034
500×500	5.9483	20.5430
550×550	5.9160	20.7394
600×600	5.8833	20.9032
650×650	5.8509	21.0419
700×700	5.8191	21.1610
750×750	5.7881	21.2642
800×800	5.7581	21.3546
850×850	5.7292	21.4344
900×900	5.7013	21.5054
950×950	5.6744	21.5689
1000×1000	5.6484	21.6260

Table 6
The speed up ratio in case of using FNN and FNN after image decomposition into sub-images (25×25 pixels) for images of different sizes (from $N = 1050$ to 2000, $n = 25$, $q = 30$)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
1050×1050	5.6234	21.6778
1100×1100	5.5994	21.7248
1150×1150	5.5762	21.7678
1200×1200	5.5538	21.8072
1250×1250	5.5322	21.8434
1300×1300	5.5113	21.8769
1350×1350	5.4912	21.9079
1400×1400	5.4717	21.9366
1450×1450	5.4528	21.9634
1500×1500	5.4345	21.9884
1550×1550	5.4168	22.0118
1600×1600	5.3996	22.0338
1650×1650	5.3830	22.0544
1700×1700	5.3668	22.0738
1750×1750	5.3511	22.0921
1800×1800	5.3358	22.1094
1850×1850	5.3209	22.1257
1900×1900	5.3064	22.1412
1950×1950	5.2923	22.1559
2000×2000	5.2786	22.1699

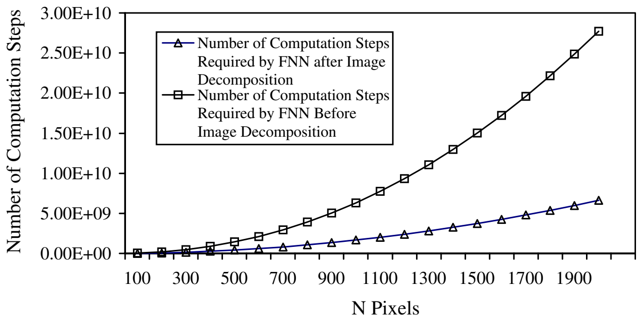

Fig. 4. A comparison between the number of computation steps required by FNN before and after Image decomposition.

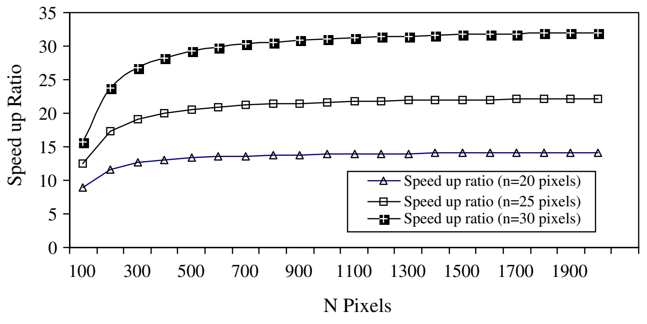

Fig. 5. The speed up ratio in case of image decomposition and different window size (n), ($L = 25 \times 25$).

increased smoothly. Thus, the linearity of the computation steps required by fast 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 fast neural networks is decreased, the speed up ratio still increase in case of using fast neural networks and matrix decomposition as listed in Table 7. Moreover, as shown in Fig. 5, the speed up ratio in case of fast 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 30×30 is larger than that of size 20×20. Simulation results for the speed up ratio in case of using fast neural networks and image decomposition is listed in Table 8. 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 5×5 or 10×10 using only fast neural networks, the speed ratio becomes less than one as shown in Tables 9–12. On the other hand, from the same tables it is clear that using fast neural networks 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.

$$L \geqslant n \tag{23}$$

Table 7
The speed up ratio in case of using FNN and FNN after matrix decomposition into sub-matrices (25×25 elements) for very large matrices (from $N = 100{,}000$ to $2{,}000{,}000$, $n = 25$, $q = 30$)

Matrix size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after matrix decomposition
100,000 × 100,000	3.6109	22.7038
200,000 × 200,000	3.4112	22.7092
300,000 × 300,000	3.3041	22.7110
400,000 × 400,000	3.2320	22.7119
500,000 × 500,000	3.1783	22.7125
600,000 × 600,000	3.1357	22.7128
700,000 × 700,000	3.1005	22.7131
800,000 × 800,000	3.0707	22.7133
900,000 × 900,000	3.0448	22.7134
1,000,000 × 1,000,000	3.0221	22.7136
1,100,000 × 1,100,000	3.0018	22.7137
1,200,000 × 1,200,000	2.9835	22.7138
1,300,000 × 1,300,000	2.9668	22.7138
1,400,000 × 1,400,000	2.9516	22.7139
1,500,000 × 1,500,000	2.9376	22.7139
1,600,000 × 1,600,000	2.9245	22.7140
1,700,000 × 1,700,000	2.9124	22.7140
1,800,000 × 1,800,000	2.9011	22.7141
1,900,000 × 1,900,000	2.8904	22.7141
2,000,000 × 2,000,000	2.8804	22.7141

Table 8
The practical speed up ratio in case of using FNN and FNN after image decomposition into sub-images (25×25 pixels) for images of different sizes (from $N = 100$ to 2000, $n = 25$, $q = 30$)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
100 × 100	10.75	34.55
200 × 200	9.19	35.65
300 × 300	8.43	36.73
400 × 400	7.45	37.70
500 × 500	7.13	38.66
600 × 600	6.97	39.61
700 × 700	6.83	40.56
800 × 800	6.68	41.47
900 × 900	6.79	42.39
1000 × 1000	6.59	43.28
1100 × 1100	6.66	44.14
1200 × 1200	6.62	44.95
1300 × 1300	6.57	45.71
1400 × 1400	6.53	46.44
1500 × 1500	6.49	47.13
1600 × 1600	6.45	47.70
1700 × 1700	6.41	48.19
1800 × 1800	6.38	48.68
1900 × 1900	6.35	49.09
2000 × 2000	6.31	49.45

Table 9
The speed up ratio in case of using FNN and FNN after image decomposition into sub-images (5×5 pixels) for images of different sizes (from $N = 50$ to 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
50 × 50	0.3361	1.3282
100 × 100	0.3141	1.4543
150 × 150	0.2985	1.4965
200 × 200	0.2872	1.5177
250 × 250	0.2785	1.5303
300 × 300	0.2716	1.5388
350 × 350	0.2658	1.5448
400 × 400	0.2610	1.5493
450 × 450	0.2568	1.5529
500 × 500	0.2531	1.5557
550 × 550	0.2498	1.5580
600 × 600	0.2469	1.5599
650 × 650	0.2442	1.5615
700 × 700	0.2418	1.5629
750 × 750	0.2396	1.5641
800 × 800	0.2375	1.5652
850 × 850	0.2356	1.5661
900 × 900	0.2339	1.5669
950 × 950	0.2322	1.5677
1000 × 1000	0.2306	1.5683

Table 10
The speed up ratio in case of using FNN and FNN after image decomposition into sub-images (5×5 pixels) for images of different sizes (from $N = 50$ to 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
1050 × 1050	0.2292	1.5689
1100 × 1100	0.2278	1.5695
1150 × 1150	0.2265	1.5700
1200 × 1200	0.2253	1.5704
1250 × 1250	0.2241	1.5709
1300 × 1300	0.2230	1.5713
1350 × 1350	0.2219	1.5716
1400 × 1400	0.2209	1.5720
1450 × 1450	0.2199	1.5723
1500 × 1500	0.2189	1.5726
1550 × 1550	0.2180	1.5728
1600 × 1600	0.2172	1.5731
1650 × 1650	0.2163	1.5733
1700 × 1700	0.2155	1.5735
1750 × 1750	0.2148	1.5738
1800 × 1800	0.2140	1.5740
1850 × 1850	0.2133	1.5742
1900 × 1900	0.2126	1.5743
1950 × 1950	0.2119	1.5745
2000 × 2000	0.2112	1.5747

Table 11
The speed up ratio in case of using FNN and FNN after image decomposition into sub-images (10×10 pixels) for images of different sizes (from N = 50 to 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
50 × 50	1.1202	3.1369
100 × 100	1.1503	3.9558
150 × 150	1.1303	4.2397
200 × 200	1.1063	4.3829
250 × 250	1.0842	4.4691
300 × 300	1.0647	4.5267
350 × 350	1.0474	4.5678
400 × 400	1.0321	4.5987
450 × 450	1.0185	4.6228
500 × 500	1.0063	4.6420
550 × 550	0.9952	4.6578
600 × 600	0.9851	4.6709
650 × 650	0.9758	4.6820
700 × 700	0.9672	4.6915
750 × 750	0.9593	4.6998
800 × 800	0.9519	4.7070
850 × 850	0.9451	4.7133
900 × 900	0.9386	4.7190
950 × 950	0.9325	4.7241
1000 × 1000	0.9268	4.7286

Table 12
The speed up ratio in case of using FNN and FNN after image decomposition into sub-images (10×10 pixels) for images of different sizes (from $N = 1050$ to 2000, $n = 10$, $q = 30$)

Image size	Speed up ratio in case of using FNN	Speed up ratio in case of using FNN after image decomposition
1050 × 1050	0.9214	4.7328
1100 × 1100	0.9163	4.7365
1150 × 1150	0.9114	4.7399
1200 × 1200	0.9068	4.7431
1250 × 1250	0.9023	4.7460
1300 × 1300	0.8981	4.7486
1350 × 1350	0.8941	4.7511
1400 × 1400	0.8902	4.7534
1450 × 1450	0.8865	4.7555
1500 × 1500	0.8829	4.7575
1550 × 1550	0.8795	4.7594
1600 × 1600	0.8762	4.7611
1650 × 1650	0.8730	4.7628
1700 × 1700	0.8699	4.7643
1750 × 1750	0.8669	4.7658
1800 × 1800	0.8640	4.7672
1850 × 1850	0.8613	4.7685
1900 × 1900	0.8586	4.7697
1950 × 1950	0.8559	4.7709
2000 × 2000	0.8534	4.7720

For example, in case of detecting 5×5 pattern, the image must be decomposed into sub-images of size no more 5×5.

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 fast neural network simulated on a single processor or a separated node in a clustered system. The number of operations (ω) performed by each processor/node (sub-images tested by one processor/node):

$$\omega = \frac{\text{The total number of sub-images}}{\text{Number of Processors/nodes}} \tag{24}$$

$$\omega = \frac{\alpha}{\mathrm{Pr}} \tag{25}$$

where Pr is the number of processors or nodes.

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

$$\gamma = \omega\beta \tag{26}$$

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

$$\eta_{\mathrm{dp}} = \frac{q(2n^2-1)(N-n+1)^2}{\mathrm{ceil}(((q(\alpha+1)+\alpha)(5N_s^2\log_2 N_s^2)+\alpha q(8N_s^2-n^2)+N_s)/\mathrm{Pr})} \tag{27}$$

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

As shown in Tables 13 and 14, 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 (ε) is required to obtain the over all summation for each sub-image. If the sub-image has n^2 elements, then the required number of processors will be n^2. As a result, the number of computation steps will be $\alpha q(1+\varepsilon)$, where ε is a small number depending on the value of n. For example, when $n = 20$, then $\varepsilon = 6$ and if $n = 25$, then $\varepsilon = 7$. The speed up ratio can be calculated as:

$$\eta = (2n^2-1)(N-n+1)^2/\alpha(1+\varepsilon) \tag{28}$$

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

$$\eta = (2n^2-1)(N-n+1)^2/(1+\varepsilon) \tag{29}$$

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

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

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.

Table 13
The speed up ratio in case of using FNN after image decomposition into sub-images (25×25 pixels) for images of different sizes (from $N = 50$ to 1000, $n = 25$, $q = 30$) using 16 parallel processors or 16 nodes

Image size	Speed up ratio
50 × 50	81.1403
100 × 100	199.3946
150 × 150	250.5611
200 × 200	277.7780
250 × 250	294.5171
300 × 300	305.8174
350 × 350	313.9482
400 × 400	320.0748
450 × 450	324.8552
500 × 500	328.6882
550 × 550	331.8296
600 × 600	334.4509
650 × 650	336.6712
700 × 700	338.5758
750 × 750	340.2276
800 × 800	341.6738
850 × 850	342.9504
900 × 900	344.0856
950 × 950	345.1017
1000 × 1000	346.0164

Table 14
The speed up ratio in case of using FNN after image decomposition into sub-images (25×25 pixels) for images of different sizes (from $N = 1050$ to 2000, $n = 25$, $q = 30$) using 16 parallel processors or 16 nodes

Image size	Speed up ratio
1050 × 1050	346.8442
1100 × 1100	347.5970
1150 × 1150	348.2844
1200 × 1200	348.9147
1250 × 1250	349.4946
1300 × 1300	350.0300
1350 × 1350	350.5258
1400 × 1400	350.9862
1450 × 1450	351.4150
1500 × 1500	351.8152
1550 × 1550	352.1896
1600 × 1600	352.5406
1650 × 1650	352.8704
1700 × 1700	353.1808
1750 × 1750	353.4735
1800 × 1800	353.7500
1850 × 1850	354.0115
1900 × 1900	354.2593
1950 × 1950	354.4943
2000 × 2000	354.7177

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

In [4], the authors stated that image normalization to avoid weak or strong illumination could not be done in the frequency space. This is because the image normalization is local and not easily computed in the Fourier space of the whole image. Here, a simple method for image normalization is presented. In [17–19], the authors stated that centering and normalizing the image can be obtained by centering and normalizing the weights as follows [17–19]:

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

$$\bar{X}_{rc} = X_{rc} - \bar{x}_{rc} \tag{31}$$

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:

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - \bar{x}_{rc} \otimes W_i \tag{32}$$

where

$$\bar{x}_{rc} = \frac{X_{rc}}{n^2} \tag{33}$$

Combining Eqs. (32) and (33), the following expression can be obtained:

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - \frac{X_{rc}}{n^2} \otimes W_i \tag{34}$$

which is the same as:

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - X_{rc} \otimes \frac{W_i}{n^2} \tag{35}$$

The centered weights are given by:

$$\bar{W}_i = W_i - \frac{W_i}{n^2} \tag{36}$$

Also, Eq. (35) can be written as:

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes \left(W_i - \frac{W_i}{n^2}\right) \tag{37}$$

So, it can be concluded that:

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes \bar{W}_i \tag{38}$$

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

During the test phase, each sub-image in the input image is multiplied (dot multiplication) by the weight matrix and this operation is repeated for all possible sub-images in the input image. Repeating this process for all sub-images in the input image is equivalent to the cross correlation operation. Therefore, there is no cross correlation between each sub-image and the weight matrix. The cross correlation is done between the weight matrix and the whole input image. Thus, this proves that there is no need to the proof of Eq. (38) (presented in [17–19] which is mathematically wrong. The result of Eq. (38) 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:

$$\bar{X}_{rc} \otimes W_i \neq X_{rc} \otimes \bar{W}_i \tag{39}$$

This fact is true for all types and values of matrices except symmetric matrices and our new technique of image decomposition presented in Section 3. A practical example is given in Appendix D.

Furthermore, the definition of the mean value, Eq. (34) which presented in [17–19] is not correct and must be :

$$\bar{x}_{rc} = \frac{\sum_{i,j=1}^{n} X_{rc}(i,j)}{n^2} \tag{40}$$

which leads to the conclusion that the proof of Eq. (38) (presented in [17–19]) not true.

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:

$$X \cdot W = \sum_{i,j=1}^{n^2} X_{ij} W_{ij} \tag{41}$$

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

$$1_{n\times n} \cdot X = X \cdot 1_{n\times n} = \sum_{i,j=1}^{n^2} X_{ij} \tag{42}$$

Where $1_{n\times n}$ is a $n \times n$ matrix where every element is 1.

$$1_{n\times n} \cdot W = W \cdot 1_{n\times n} = \sum_{i,j=1}^{n^2} W_{ij} \tag{43}$$

Lemma 1. $\bar{w}1_{n\times n} \cdot X = \bar{x}1_{n\times n} \cdot W$.

Proof. From Eqs. (40)–(43), we can conclude that:

$$\bar{w}1_{n\times n} \cdot X = \bar{w}\sum_{i,j=1}^{n^2} X_{ij} = \frac{1}{n^2}\sum_{i,j=1}^{n^2} W_{ij} \cdot \sum_{i,j=1}^{n^2} X_{ij} \tag{44}$$

Which can be reformulated as:

$$\bar{w}1_{n\times n} \cdot X = \frac{1}{n^2}\sum_{i,j=1}^{n^2} W_{ij} \cdot \sum_{i,j=1}^{n^2} X_{ij} \tag{45}$$

Also,

$$\bar{x}1_{nxn} \cdot W = \bar{x}\sum_{i,j=1}^{n^2} W_{ij} = \frac{1}{n^2}\sum_{i,j=1}^{n^2} X_{ij} \cdot \sum_{i,j=1}^{n^2} W_{ij} \tag{46}$$

Which is the same as:

$$\bar{x}1_{nxn} \cdot W = \frac{1}{n^2}\sum_{i,j=1}^{n^2} X_{ij} \cdot \sum_{i,j=1}^{n^2} W_{ij} \tag{47}$$

It is clear that Eq. (45) is the same as Eq. (47), which means:

$$\bar{w}1_{nxn} \cdot X = \bar{x}1_{nxn} \cdot W \quad \square \tag{48}$$

Theorem

$\bar{X} \cdot W = \bar{W} \cdot X$

Proof

$$\begin{aligned}\bar{X} \cdot W &= (X - \bar{x}1_{n\times n}) \cdot W \\ &= X \cdot W - \bar{x}1_{n\times n} \cdot W \\ &= X \cdot W - X \cdot 1_{n\times n}\overline{w} \\ &= X(W - \overline{w} \cdot 1_{n\times n}) \\ &= X \cdot \bar{W}\end{aligned}$$

So, we may conclude that:

$$\bar{X}_{rc} \cdot W_i = X_{rc} \cdot \bar{W}_i \tag{49}$$

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. (49) and a practical example is given in Appendix E. □

As proved in our previous paper [22], the relation defined by Eq. (38) is true only for the resulting middle value. This is under two conditions. The first is to apply the technique of fast 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 were 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. (38) 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 Γ can be calculated as:

$$\Gamma = \frac{(N-n+1)^2}{q} \tag{50}$$

The speed up ratio of the normalization process for images of different sizes is listed in Table 15. As a result, we may conclude that:

Table 15
The speed up ratio of the normalization process for images of different sizes ($n = 20$, q = 100)

Image size	Speed up ratio
100 × 100	62
200 × 200	328
300 × 300	790
400 × 400	1452
500 × 500	2314
600 × 600	3376
700 × 700	4638
800 × 800	6100
900 × 900	7762
1000 × 1000	9624
1100 × 1100	11686
1200 × 1200	13948
1300 × 1300	16410
1400 × 1400	19072
1500 × 1500	21934
1600 × 1600	24996
1700 × 1700	28258
1800 × 1800	31720
1900 × 1900	35382
2000 × 2000	39244

1. Using this technique, normalization in the frequency domain can be done through normalizing the weights in spatial domain.
2. Normalization of an image through normalization of weights is faster than normalization of each 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 η_c in this case can be given by:

$$\eta_c = \frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{q(2n^2-1)(N-n+1)^2} \tag{51}$$

which can be simplified to:

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

(b) *For fast 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 needed by fast neural networks with weight normalization, which is done off line. The over all speed up ratio η_o can be given by:

$$\eta_o = \frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{(2q+1)(5N^2\log_2 N^2) + q(8N^2-n^2) + N} \tag{53}$$

which can be simplified to:

$$\eta_o = \frac{(N-n+1)^2(q(2n^2-1)+2n^2)}{(2q+1)(5N^2\log_2 N^2) + q(8N^2-n^2) + N} \tag{54}$$

Table 16
Theoretical results for the speed up ratio in case of image normalization by normalizing the input weights

Image size	Speed up ratio ($n = 20$)	Speed up ratio ($n = 25$)	Speed up ratio ($n = 30$)
100 × 100	3.7869	5.2121	6.5532
200 × 200	4.1382	6.1165	8.3167
300 × 300	4.1320	6.2313	8.6531
400 × 400	4.0766	6.2063	8.7031
500 × 500	4.0152	6.1467	8.6684
600 × 600	3.9570	6.0796	8.6054
700 × 700	3.9039	6.0132	8.5334
800 × 800	3.8557	5.9502	8.4603
900 × 900	3.8120	5.8915	8.3891
1000 × 1000	3.7723	5.8369	8.3212
1100 × 1100	3.7360	5.7862	8.2568
1200 × 1200	3.7027	5.7391	8.1961
1300 × 1300	3.6719	5.6952	8.1389
1400 × 1400	3.6434	5.6542	8.0849
1500 × 1500	3.6169	5.6158	8.0340
1600 × 1600	3.5922	5.5798	7.9858
1700 × 1700	3.5690	5.5458	7.9403
1800 × 1800	3.5472	5.5138	7.8971
1900 × 1900	3.5266	5.4835	7.8560
2000 × 2000	3.5072	5.4547	7.8169

Table 17
Practical speed up ratio for images with different sizes in case of image normalization by normalizing the input weights

Image size	Speed up ratio ($n = 20$)	Speed up ratio ($n = 25$)	Speed up ratio ($n = 30$)
100 × 100	8.91	12.03	16.74
200 × 200	7.43	10.42	15.39
300 × 300	6.72	9.72	14.45
400 × 400	5.99	8.61	13.59
500 × 500	5.75	8.32	12.94
600 × 600	5.61	8.09	11.52
700 × 700	5.49	7.97	11.04
800 × 800	5.41	7.83	10.74
900 × 900	5.32	7.71	10.56
1000 × 1000	5.29	7.58	10.45
1100 × 1100	5.41	7.83	10.81
1200 × 1200	5.36	7.77	10.76
1300 × 1300	5.32	7.71	10.71
1400 × 1400	5.28	7.65	10.66
1500 × 1500	5.24	7.60	10.62
1600 × 1600	5.21	7.56	10.58
1700 × 1700	5.18	7.52	10.52
1800 × 1800	5.14	7.48	10.47
1900 × 1900	5.11	7.44	10.43
2000 × 2000	5.08	7.41	10.38

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

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

The overall speed up ratio Eq. (55) with images of different sizes and different sizes of windows is listed in Table 16. 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 fast normalized neural networks is done faster than conventional neural networks with or without normalization of the input image. The overall practical speed up ratio Eq. (55) after off line normalization of weights is listed in Table 17.

6. Conclusions

Normalized neural networks for fast pattern detection in a given image have been presented. It has been proved mathematically and practically that the speed of the detection process becomes faster than conventional neural networks. This has been accomplished by applying cross correlation in the frequency domain between the input image and the normalized input weights of the neural networks. New general formulas for fast cross correlation as well as the speed up ratio have been given. A faster 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 neural networks has been presented in order to speed up the execution time. Simulation results have shown that, using a parallel processing technique, large values of speed up ratio could be achieved. Moreover, by using fast neural networks and image decomposition, the speed up ratio has been increased with the size of the input image. Also, 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 not commutative

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

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

$$W \otimes X = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} = \begin{bmatrix} 8\times 5 & 8\times 1+9\times 5 & 9\times 1 \\ 5\times 5+8\times 3 & 6\times 5+5\times 1+9\times 3+8\times 7 & 6\times 1+9\times 7 \\ 5\times 3 & 6\times 3+5\times 7 & 6\times 7 \end{bmatrix} = \begin{bmatrix} 40 & 53 & 9 \\ 49 & 118 & 63 \\ 15 & 53 & 42 \end{bmatrix}$$

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

$$X \otimes W = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} \otimes \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} = \begin{bmatrix} 7\times 6 & 3\times 6+7\times 5 & 3\times 5 \\ 1\times 6+7\times 9 & 5\times 6+1\times 5+3\times 9+7\times 8 & 5\times 5+3\times 8 \\ 1\times 9 & 5\times 9+1\times 8 & 5\times 8 \end{bmatrix} = \begin{bmatrix} 42 & 53 & 15 \\ 69 & 118 & 49 \\ 9 & 53 & 40 \end{bmatrix}$$

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

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

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

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

$$W \otimes X = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 8\times 5 & 8\times 3+9\times 5 & 9\times 3 \\ 5\times 5+8\times 3 & 6\times 5+5\times 3+9\times 3+8\times 5 & 6\times 3+9\times 5 \\ 5\times 3 & 6\times 3+5\times 5 & 6\times 5 \end{bmatrix} = \begin{bmatrix} 40 & 69 & 27 \\ 49 & 112 & 63 \\ 15 & 43 & 30 \end{bmatrix}$$

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

$$X \otimes W = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} \otimes \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} = \begin{bmatrix} 5\times 6 & 3\times 6+5\times 5 & 3\times 5 \\ 3\times 6+5\times 9 & 5\times 6+3\times 5+3\times 9+5\times 8 & 5\times 5+3\times 8 \\ 3\times 9 & 5\times 9+3\times 8 & 5\times 8 \end{bmatrix} = \begin{bmatrix} 30 & 43 & 15 \\ 63 & 112 & 49 \\ 27 & 69 & 40 \end{bmatrix}$$

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

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

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

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

$$W \otimes X = \begin{bmatrix} 8 & 9 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 8\times 5 & 9\times 5+8\times 3 & 9\times 3 \\ 9\times 5+8\times 3 & 8\times 5+9\times 3+9\times 3+8\times 5 & 9\times 5+8\times 3 \\ 9\times 3 & 5\times 9+3\times 8 & 8\times 5 \end{bmatrix} = \begin{bmatrix} 40 & 69 & 27 \\ 69 & 122 & 69 \\ 27 & 69 & 40 \end{bmatrix}$$

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

$$X \otimes W = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} \otimes \begin{bmatrix} 8 & 9 \\ 9 & 8 \end{bmatrix} = \begin{bmatrix} 5\times 8 & 5\times 9+3\times 8 & 3\times 9 \\ 5\times 9+3\times 8 & 5\times 8+3\times 9+3\times 9+5\times 8 & 5\times 9+3\times 8 \\ 3\times 9 & 5\times 9+3\times 8 & 5\times 8 \end{bmatrix} = \begin{bmatrix} 40 & 69 & 27 \\ 69 & 122 & 69 \\ 27 & 69 & 40 \end{bmatrix}$$

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

Appendix B. An example proves that the cross correlation between any two matrices is different from their convolution

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

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

$$W \Diamond X = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix} \Diamond \begin{bmatrix} 8 & 9 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 6\times 5 & 5\times 5+6\times 1 & 5\times 1 \\ 9\times 5+6\times 3 & 8\times 5+9\times 1+5\times 3+6\times 7 & 8\times 1+5\times 7 \\ 9\times 3 & 8\times 3+9\times 7 & 8\times 7 \end{bmatrix} = \begin{bmatrix} 30 & 31 & 5 \\ 63 & 106 & 43 \\ 27 & 87 & 56 \end{bmatrix}$$

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

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

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

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

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

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

Appendix C. An example for cross correlation with matrix decomposition

$$\text{Let } X = \begin{bmatrix} 5 & 1 & 8 & 6 \\ 3 & 7 & 3 & 4 \\ 1 & 2 & 9 & 5 \\ 6 & 5 & 4 & 2 \end{bmatrix} \text{ and } W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix}$$

Then the cross correlation (CC) between W and X can be computed as follows:

$$CC = W \otimes X = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 1 & 8 & 6 \\ 3 & 7 & 3 & 4 \\ 1 & 2 & 9 & 5 \\ 6 & 5 & 4 & 2 \end{bmatrix}$$
$$= \begin{bmatrix} 8\times 5 & 9\times 5+8\times 1 & 9\times 1+8\times 8 & 9\times 8+8\times 6 & 9\times 6 \\ 5\times 5+8\times 3 & 6\times 5+5\times 1+9\times 3+8\times 7 & 6\times 1+5\times 8+9\times 7+8\times 3 & 6\times 8+5\times 6+9\times 3+8\times 4 & 6\times 6+9\times 4 \\ 5\times 3+8\times 1 & 6\times 3+5\times 7+9\times 1+8\times 2 & 6\times 7+5\times 3+9\times 2+8\times 9 & 6\times 3+5\times 4+9\times 9+8\times 5 & 6\times 4+9\times 5 \\ 5\times 1+8\times 6 & 6\times 1+5\times 2+9\times 6+8\times 5 & 6\times 2+5\times 9+9\times 5+8\times 4 & 6\times 9+5\times 5+9\times 4+8\times 2 & 6\times 5+9\times 2 \\ 5\times 6 & 6\times 6+5\times 5 & 6\times 5+5\times 4 & 6\times 4+5\times 2 & 6\times 2 \end{bmatrix}$$
$$CC = \begin{bmatrix} 40 & 53 & 73 & 120 & 54 \\ 49 & 118 & 133 & 137 & 72 \\ 23 & 78 & 147 & 159 & 69 \\ 53 & 110 & 134 & 131 & 48 \\ 30 & 61 & 50 & 34 & 12 \end{bmatrix}$$

Suppose that X is decomposed into four smaller matrices $x1$, $x2$, $x3$, and $x4$ as follows:

$$x1 = \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}, x2 = \begin{bmatrix} 8 & 6 \\ 3 & 4 \end{bmatrix}, x3 = \begin{bmatrix} 1 & 2 \\ 6 & 5 \end{bmatrix}, \text{ and } x4 = \begin{bmatrix} 9 & 5 \\ 4 & 2 \end{bmatrix}$$

Then, the cross correlation between each resulting matrix and the matrix W can be computed as follows:

$$\text{CC1} = W \otimes x1 = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 5 & 1 \\ 3 & 7 \end{bmatrix}$$
$$= \begin{bmatrix} 8\times 5 & 9\times 5+8\times 1 & 9\times 1 \\ 5\times 5+8\times 3 & 6\times 5+5\times 1+9\times 3+8\times 7 & 6\times 1+9\times 7 \\ 5\times 3 & 6\times 3+5\times 7 & 6\times 7 \end{bmatrix}$$
$$= \begin{bmatrix} 40 & 53 & 9 \\ 49 & 118 & 69 \\ 15 & 53 & 42 \end{bmatrix}$$

$$\text{CC2} = W \otimes x2 = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 8 & 6 \\ 3 & 4 \end{bmatrix}$$
$$= \begin{bmatrix} 8\times 8 & 9\times 8+8\times 6 & 9\times 6 \\ 5\times 8+8\times 3 & 6\times 8+5\times 6+9\times 3+8\times 4 & 6\times 6+9\times 4 \\ 5\times 3 & 6\times 3+5\times 4 & 6\times 4 \end{bmatrix}$$
$$= \begin{bmatrix} 64 & 120 & 54 \\ 64 & 137 & 72 \\ 15 & 38 & 24 \end{bmatrix}$$

$$\text{CC3} = x3 \otimes W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \\ 6 & 5 \end{bmatrix}$$
$$= \begin{bmatrix} 8\times 1 & 9\times 1+8\times 2 & 9\times 2 \\ 5\times 1+8\times 6 & 6\times 1+5\times 2+9\times 6+8\times 5 & 6\times 2+9\times 5 \\ 5\times 6 & 6\times 6+5\times 5 & 6\times 5 \end{bmatrix}$$
$$= \begin{bmatrix} 8 & 25 & 18 \\ 53 & 110 & 57 \\ 30 & 61 & 30 \end{bmatrix}$$

$$\text{CC4} = x4 \otimes W = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix} \otimes \begin{bmatrix} 9 & 5 \\ 4 & 2 \end{bmatrix}$$
$$= \begin{bmatrix} 8\times 9 & 9\times 9+8\times 5 & 9\times 5 \\ 5\times 9+8\times 4 & 6\times 9+5\times 5+9\times 4+8\times 2 & 6\times 5+9\times 2 \\ 5\times 4 & 6\times 4+5\times 2 & 6\times 2 \end{bmatrix}$$
$$= \begin{bmatrix} 72 & 121 & 45 \\ 77 & 131 & 48 \\ 20 & 34 & 12 \end{bmatrix}$$

The total result of cross correlating the resulting smaller matrices with the matrix W can be computed as:

$$\mathrm{CCT} = \mathrm{CC1} + \mathrm{CC2} + \mathrm{CC3} + \mathrm{CC4}$$

$$\mathrm{CCT} = \begin{bmatrix} \mathrm{CC1}(1,1) & \mathrm{CC1}(1,2) & \mathrm{CC1}(1,3)+\mathrm{CC2}(1,1) & \mathrm{CC2}(1,2) & \mathrm{CC2}(1,3) \\ \mathrm{CC1}(2,1) & \mathrm{CC1}(2,2) & \mathrm{CC2}(2,3)+\mathrm{CC2}(2,1) & \mathrm{CC2}(2,2) & \mathrm{CC2}(2,3) \\ \mathrm{CC1}(3,1)+\mathrm{CC3}(1,1) & \mathrm{CC1}(3,2)+\mathrm{CC3}(1,2) & \mathrm{CC1}(3,3)+\mathrm{CC2}(3,1)+\mathrm{CC3}(1,3)+\mathrm{CC4}(1,1) & \mathrm{CC2}(3,2)+\mathrm{CC4}(1,2) & \mathrm{CC2}(3,3)+\mathrm{CC4}(1,3) \\ \mathrm{CC3}(2,1) & \mathrm{CC3}(2,2) & \mathrm{CC3}(2,3)+\mathrm{CC4}(1,2) & \mathrm{CC4}(2,2) & \mathrm{CC4}(2,3) \\ \mathrm{CC3}(3,1) & \mathrm{CC3}(3,2) & \mathrm{CC3}(3,3)+\mathrm{CC4}(1,3) & \mathrm{CC4}(3,2) & \mathrm{CC4}(3,3) \end{bmatrix}$$

$$\mathrm{CCT} = \begin{bmatrix} 40 & 53 & 9+64 & 120 & 54 \\ 49 & 118 & 69+64 & 137 & 72 \\ 15+8 & 53+25 & 42+15+18+72 & 38+121 & 24+45 \\ 53 & 110 & 57+77 & 131 & 48 \\ 30 & 61 & 30+20 & 34 & 12 \end{bmatrix}$$

$$\mathrm{CCT} = \begin{bmatrix} 40 & 53 & 73 & 120 & 54 \\ 49 & 118 & 133 & 137 & 72 \\ 23 & 78 & 147 & 159 & 69 \\ 53 & 110 & 134 & 131 & 48 \\ 30 & 61 & 50 & 34 & 12 \end{bmatrix}$$

Which means that CC = CCT. This proves that the result of cross correlating a large matrix (X) with another matrix (W) equals to the results of cross correlating the resulting smaller sub matrices ($X1$, $X2$, $X3$, and $X4$) after decomposition.

Appendix D. A cross correlation example between a normalized matrix and other non-normalized one and vise versa

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

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

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

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

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

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

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

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 E.

Appendix E. A dot product example between a normalized matrix and other non-normalized one and vise versa

This is to validate the correctness of Eq. (49). The left hand side of Eq. (49) can be expressed as follows:

$$\bar{X} \cdot W = \begin{bmatrix} X_{1,1} - \bar{X} \ldots X_{1,n} - \bar{X} \\ \vdots \\ X_{n,1} - \bar{X} \ldots X_{n,n} - \bar{X} \end{bmatrix} \cdot \begin{bmatrix} W_{1,1} \ldots W_{1,n} \\ \vdots \\ W_{n,1} \ldots W_{n,n} \end{bmatrix} \tag{56}$$

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

$$X \cdot \bar{W} = \begin{bmatrix} X_{1,1} \ldots X_{1,n} \\ \vdots \\ X_{n,1} \ldots X_{n,n} \end{bmatrix} \cdot \begin{bmatrix} W_{1,1} - \bar{W} \ldots W_{1,n} - \bar{W} \\ \vdots \\ W_{n,1} - \bar{W} \ldots W_{n,n} - \bar{W} \end{bmatrix} \tag{57}$$

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

$$\begin{aligned} \bar{X} &= \frac{X_{1,1} + X_{1,2} + \ldots + X_{n,n}}{n^2} \\ \bar{W} &= \frac{W_{1,1} + W_{1,2} + \ldots + W_{n,n}}{n^2} \end{aligned} \tag{58}$$

By substituting from Eq. (58) in Eqs. (56) and (57), then simplifying the results we can easily conclude that $\bar{X}_{rc} \cdot W_i = X_{rc} \cdot \bar{W}_i$. Here is also a practical example:

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

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

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

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

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

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

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.

References

[1] R. Klette, P. Zamperon, Handbook of image processing operators, John Wiley and Sons, New York, 1996.

[2] H.A. Rowley, S. Baluja, T. Kanade, Neural network-based face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1) (1998) 23–38.

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

[4] R. Feraud, O. Bernier, J.E. Viallet, M. Collobert, 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, 28–30 March, 2000.

[5] Y. Zhu, S. Schwartz, M. Orchard, Fast face detection using subspace discriminate wavelet features, in: Proceedings of IEEE Computer Society International Conference on Computer Vision and Pattern Recognition (CVPR'00), South Carolina, June 13–15, 2000, vol.1, pp. 1636–1643.

[6] H.M. El-Bakry, Automatic human face recognition using modular neural networks, Machine Graphics & Vision Journal (MG& V) 10 (1) (2001) 47–73.

[7] S. Srisuk, W. Kurutach, A new robust face detection in color images, in: Proceedings of IEEE Computer Society International Conference on Automatic Face and Gesture Recognition, Washington D.C., USA, May 20–21, 2002, pp. 306–311.

[8] H.M. El-Bakry, Face detection using fast neural networks and image decomposition, Neurocomputing Journal 48 (2002) 1039–1046.

[9] H.M. El-Bakry, Human iris detection using fast cooperative modular neural networks and image decomposition, Machine Graphics & Vision Journal (MG& V) 11 (4) (2002) 498–512.

[10] H.M. El-Bakry, Q. Zhao, Fast object/face detection using neural networks and fast Fourier transform, International Journal of Signal Processing 1 (3) (2004) 182–187.

[11] H.M. El-Bakry, Q. Zhao, A modified cross correlation in the frequency domain for fast pattern detection using neural networks, International Journal of Signal Processing 1 (3) (2004) 188–194.

[12] H.M. El-Bakry, Q. Zhao, Face detection using fast neural processors and image decomposition, International Journal of Computational Intelligence 1 (4) (2004) 313–316.

[13] H.M. El-Bakry, Q. Zhao, Fast complex valued time delay neural networks, International Journal of Computational Intelligence 2 (1) (2005) 16–26.

[14] H.M. El-Bakry, Q. Zhao, A fast neural algorithm for serial code detection in a stream of sequential data, International Journal of Information Technology 2 (1) (2005) 71–90.

[15] H.M. El-Bakry, Q. Zhao, A new high speed neural model for character recognition using cross correlation and matrix decomposition, International Journal of Signal Processing 2 (3) (2005) 183–202.

[16] H.M. El-Bakry, Q. Zhao, Fast pattern detection using normalized neural networks and cross correlation in the frequency domain, Special issue on Advances in Intelligent Vision Systems: Methods and Applications-Part I, EURASIP Journal on Applied Signal Processing 2005 (13) (2005) 2054–2060.

[17] S. Ben-Yacoub, B. Fasel, J. Luettin, Fast face detection using MLP and FFT, in: Proceedings of the Second International Conference on Audio and Video-based Biometric Person Authentication (AVBPA'99), 1999.

[18] B. Fasel, Fast multi-scale face detection, IDIAP-Com 98-04, 1998.

[19] S. Ben-Yacoub, Fast object detection using MLP and FFT, IDIAP-RR 11, IDIAP, 1997.

[20] James W. Cooley, John W. Tukey, An algorithm for the machine calculation of complex Fourier series, Mathematical Computing 19 (1965) 297–301.

[21] J.P. Lewis, Fast normalized cross correlation. Available from: http://www.idiom.com/~zilla/Papers/nvisionInterface/nip.html.

[22] H.M. El-Bakry, Face detection using fast neural networks and image decomposition, Neurocomputing Journal 48 (2002) 1039–1046.

[23] H.M. El-Bakry, Comments on Using MLP and FFT for fast object/face detection, in: Proceedings of IEEE IJCNN'03, Portland, Oregon, July, 20–24, 2003, pp. 1284–1288.

[24] H.M. El-Bakry, H. Stoyan, Fast neural networks for object/face detection, in: Proceedings of the 30th Anniversary SOFSEM Conference on Current Trends in Theory and Practice of Computer Science, 24–30 January, 2004, Hotel VZ MERIN, Czech Republic.

[25] H.M. El-Bakry, H. Stoyan, Fast neural networks for sub-matrix (object/face) detection, in: Proceedings of IEEE International Symposium on Circuits and Systems, Vancouver, Canada, 23–26 May, 2004.

[26] H.M. El-Bakry, Fast sub-image detection using neural networks and cross correlation in frequency domain, in: Proceedings of IS 2004: 14th Annual Canadian Conference on Intelligent Systems, Ottawa, Ontario, 6–8 June, 2004.

[27] H.M. El-Bakry, H. Stoyan, Fast neural networks for code detection in a stream of sequential data, in: Proceedings of CIC 2004 International Conference on Communications in Computing, Las Vegas, Nevada, USA, 21–24 June, 2004.

[28] H.M. El-Bakry, Fast neural networks for object/face detection, in: Proceedings of 5th International Symposium on Soft Computing for Industry with Applications of Financial Engineering, June 28–July 4, 2004, Sevilla, Andalucia, Spain.

[29] H.M. El-Bakry, H. Stoyan, A fast searching algorithm for sub-image (object/face) detection using neural networks, in: Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics, 18–21 July, 2004, Orlando, USA.

[30] H.M. El-Bakry, H. Stoyan, Fast neural networks for code detection in sequential data using neural networks for communication applications, in: Proceedings of the First International Conference on Cybernetics and Information Technologies, Systems and Applications: CITSA 2004, 21–25 July, 2004. Orlando, Florida, USA, vol. IV, pp. 150–153.

[31] H.M. El-bakry, M.A. Abo-elsoud, M.S. Kamel, Fast modular neural networks for human face detection, in: Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks, Como, Italy, vol. III, 24–27 July, 2000, pp. 320–324.

[32] H.M. El-bakry, Fast iris detection using cooperative modular neural nets, in: Proceedings of the 6th International Conference on Soft Computing, 1–4 October, 2000, Japan.

[33] H.M. El-Bakry, Automatic human face recognition using modular neural networks, Machine Graphics & Vision Journal (MG& V) 10 (1) (2001) 47–73.

[34] H.M. El-bakry, Fast iris detection using cooperative modular neural networks, in: Proceedings of the 5th International Conference on Artificial Neural Nets and Genetic Algorithms, 22–25 April, 2001, Sydney, Czech Republic, pp. 201–204.

[35] H.M. El-bakry, Fast iris detection using neural nets, in: Proceedings of the 14th Canadian Conference on Electrical and Computer Engineering, 13–16 May, 2001, Canada, pp.1409–1415.

[36] H.M. El-bakry, Human iris detection using fast cooperative modular neural nets, in: Proceedings of INNS-IEEE International Joint

Conference on Neural Networks, 14–19 July, 2001, Washington, DC, USA, pp. 577–582.

[37] H.M. El-bakry, Human iris detection for information security using fast neural nets, Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics, 22–25 July, 2001, Orlando, Florida, USA.

[38] H.M. El-bakry, Human iris detection for personal identification using fast modular neural nets, in: Proceedings of the 2001 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences, 25–28 July, 2001, Monte Carlo Resort, Las Vegas, Nevada, USA, pp. 112–118.

[39] H.M. El-bakry, Human face detection using fast neural networks and image decomposition, in: Proceedings of the fifth International Conference on Knowledge-Based Intelligent Information & Engineering Systems, 6–8 September, 2001, Osaka-kyoiku University, Kashiwara City, Japan, pp. 1330–1334.

[40] H.M. El-Bakry, Fast iris detection for personal verification using modular neural networks, in: Proceedings of the International Conference on Computational Intelligence, 1–3 October, 2001, Dortmund, Germany, pp. 269–283.

[41] H.M. El-bakry, Fast cooperative modular neural nets for human face detection, in: Proceedings of IEEE International Conference on Image Processing, 7–10 October, 2001, Thessaloniki, Greece.

[42] H.M. El-Bakry, Fast face detection using neural networks and image decomposition, in: Proceedings of the 6th International Computer Science Conference, Active Media Technology, December 18–20, 2001, Hong Kong, China, pp. 205–215, 2001.

[43] H.M. El-Bakry, Face detection using fast neural networks and image decomposition, in: Proceedings of INNS-IEEE International Joint Conference on Neural Networks, 14–19 May, 2002, Honolulu, Hawaii, USA.

[44] H.M. El-Bakry, Q. Zhao, Fast normalized neural processors for pattern detection based on cross correlation implemented in the frequency domain, Journal of Research and Practice in Information Technology 38 (2) (2006) 151–170.

[45] H.M. El-Bakry, Q. Zhao, A new high speed neural model for character recognition using cross correlation and matrix decomposition, International Journal of Signal Processing 2 (3) (2005) 183–202.

[46] H.M. El-Bakry, Q. Zhao, Speeding-up normalized neural networks for face/object detection, Machine Graphics & Vision Journal (MG& V) 14 (1) (2005) 29–59.

[47] K.A. Ishak, S.A. Samad, A. Hussian, B.Y. Majlis, 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, 2004, pp. 5–8.