Fast Normalized Neural Networks For Object/Face Detection

ABSTRACT- Finding an object or a face in the input image is a search problem in the spatial domain. Neural networks have shown good results for detection of object/face in a given image. However, they consume long time in the dection process. Our idea is to move the test process from the spatial domain to the frequency one. A fast algorithm for object/face detection is presented. Such algorithm is designed based on cross correlation in the frequency domain between the input image and the input weights of neural networks. Moreover, the problem of local sub-image normalization in the frequency domain is solved. Furthermore, the effect of image normalization on the speed up ratio of object/face detection is also presented. Simulation results show that local sub-image normalization by normalizing the input weights is faster than sub-image normalization in the spatial domain. Moreover, the overall speed up ratio of the detection process is increased as the normalization of weights is done off line.

Keywords: Fast Object/Face Detection, Neural Networks, Cross Correlation, Image Normalization

I. Introduction

Object detection is a fundamental step before object recognition. Its reliability and performance have a major influence in a whole object recognition system. Nowadays, neural networks have shown very good results for detecting an object or a face in a given image [3,6,10]. But the problem with neural networks is that the computational complexity is very high and expensive. This is because the test is done in the spatial domain and neural networks have to process many small local windows in the images under test [5,7]. Some authors tried to speed up the detection process of neural networks [9,11,12]. They proposed a multilayer perceptron (MLP) algorithm for fast object/face detection. They claimed that applying cross correlation in the frequency domain between the input image and the neural weights is much faster than using conventional neural networks. They stated this without any constrains and introduced formulas for the number of computation steps needed by conventional neural networks and their proposed fast neural networks. Then, they deduced an equation for the speed up ratio. It was proved in [2] that their equations contain many errors which lead to an invalid speed up ratio. Moreover, a symmetry condition is necessary and must be found either in the input image or in the neural weights so that those fast neural networks can give the same correct results as conventional neural networks for detecting an object or a face in a given image. Recently, we succeeded accelerating the behavior of neural networks during the search process [1]. The speed up of the detection process is achieved by converting the input image into a symmetric one and applying cross correlation in the frequency domain between the new symmetric image and the input weights of neural networks. Mathematical proof and simulation results for fast testing of the new proposed symmetric image using Matlab are given.

There is no problem to normalize the training examples used for learning neural networks. Also, there is no problem to normalize each sub-image if the test is done in the spatial domain. Here, the proposed new idea is to move the test from the spatial domain to the frequency one for fast detection process. Thus, the question arises, how to normalize each sub-image in the frequency domain?. The problem of sub-image (local) normalization in the Fourier space was presented in [8]. This problem was solved in [4]. We proved that the number of computation steps required for normalizing the input weights is less than that needed for image normalization. But, we did not discuss the effect of normalization on the speed up ratio. Here, 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 normalization of the input weights requires fewer of computation steps than sub-image normalization. Moreover, for neural networks, normalization of input weights can be easily done off line

before starting the search process. In section II, fast neural networks for object/face detection are described. The effect of normalizing the input weights on the speed up ratio is presented in section III.

II. Fast Neural Networks Based on Cross Correlation in the Frequency Domain For Sub-Image (Object/Face) Detection

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 learnt 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 object/face. 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 object/face and viceversa. 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.

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 the spatial domain via the inverse Fourier Transform. As a result, these cross correlations can be represented by a product in the frequency domain. Thus, by using cross correlation in the frequency domain a speed up in an order of magnitude can be achieved during the detection process [1,2,3,4,6].

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

$$h_i = g\left(\sum_{j=1}^{m} \sum_{k=1}^{n} X_i(j,k) I(j,k) + b_i \right) \quad (1)$$

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 obtained to the whole image Z as follows:

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

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

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

Therefore, eq. 2 may be written as follows:

$$h_i = g\left(X_i \otimes Z + b_i\right) \quad (4)$$

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

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

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

Hence, by evaluating this cross correlation, a speed up ratio can be obtained comparable to conventional neural networks. Also, the final output of the neural network can be evaluated as follows:

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

where q is the number of neurons in the hidden layer. O(u,v) is the output of the neural network when the sliding window is located at the position (u,v) in the input image Z.

We can analayze the complexity of cross correlation in the frequency domain as follows [2]:

1- For a tested image of NxN 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 2DFFT to compute is $(2q+1)N^2\log_2N^2$.

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

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

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

which may be simplified to:

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

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

5- In order to perform cross correlation in the frequency domain, the weight matrix must have the same size as the input image. So, a number of zeros = (N^2-n^2) must be added to the weight matrix. This requires a total real number of computation steps = $q(N^2-n^2)$ for all neurons. Moreover, after computing the FFT2 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 FFT2. To create a complex number requires two real floating point operations. So, the total number of computation steps required for fast neural networks becomes [2]:

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

which can be reformulated as:

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

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 object/face detection process. The theoretical speed up factor η can be evaluated as follows [2]:

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

7- But as proved in [1,2], this cross correlation in the frequency domain (Fast Neural Networks) gives the same results as conventional cross correlation (Conventional Neural Networks) only in two cases. Either the weights are symmetric or the input image is symmetric. It is very complex to allow the weights to be symmetric in the required form which needs to be as follows [2]:

$$W = \begin{bmatrix} w & 0 \\ 0 & w_d \end{bmatrix} \qquad (12)$$

Adding this constraint to the learning rules will cause many well known problems during the training process of the neural network. The probability that the network will not be trained is very high. Another solution is to convert the input non symmetric (original) image as shown in Fig. 1 into one of the required symmetric forms as shown in Fig. 2. As the input image has a dimension of (N), the new symmetric image will have a length of (2N). In this case, the number of computation steps required for fast neural networks can be calculated as follows [2]:

$$\sigma_{2N}=((2q+1)(5(2N)^2\log_2(2N)^2)+q(8(2N)^2-n^2) +2N) \qquad (13)$$

But, converting the non-symmetric input image into a symmetric one will slow down the proposed fast neural networks more and more compared to conventional neural networks. In this case, for any size of the input image, dividing the number of operations required for conventional neural networks by those needed by fast neural networks (eq. 11) gives a speed up ratio lower than one listed in Table (1) [2].

8- In order to reduce the number of computation steps required by neural networks for object/face detection, another new symmetric form for the input image is

presented. As shown in Fig. 3, the input image is converted into symmetric form by rotating it down. Then, both the up and down images are tested as a single (symmetric) image consists of two images. In this case, the new symmetric image has (2NxN) dimensions. By substituting in eq. 9 for the new dimensions, the number of computation steps required for cross correlating this new image with the weights in the frequency domain can be calculated as follows [1]:-

$$\sigma=((2q+1)(5(2N^2\log_2 N+2N^2\log_2 2N))+q6(2N^2)+ q(2N^2-n^2)+q(2N^2)+2N) \quad (14)$$

which can be simplified to:

$$\sigma=((2q+1)(10N^2(\log_2 2N+ \log_2 N)) +q(16N^2-n^2) +2N) \quad (15)$$

So, the speed up ratio in this case can be calculated as:

$$\eta=\frac{q(2n^2-1)(N-n+1)^2}{(2q+1)(10N^2(\log_2 2N+\log_2 N))+q(16N^2-n^2)+2N} \quad (16)$$

The theoretical speed up ratio in this case with different sizes of the input image and different in size weight matrices is shown in Fig. 4. The algorithm of the conventional neural networks is compared with that of fast neural networks. Practical speed up ratio for manipulating images of different sizes and different in size weight matrices is shown in Fig. 5 using 2.80 GHz processor and Matlab ver 5.3. In our pervious work [1], the algorithm of fast neural networks was compared with the cross correlation (xcorr2) function in Matlab. This function uses the sum- Table scheme to fast the operation of cross correlation [13]. In our previous paper, we proved that our proposed fast neural networks algorithm is faster than this function. In this paper the comparison is accomplished between our proposed fast neural and conventional neural networks (conventional cross correlation in the spatial domain).

Moreover, this new symmetric configuration is useful for reducing the number of patterns that the neural network will learn. This is because the image is rotated down as shown in Fig. 3. Then, the up image and its rotated down version are tested together as one (symmetric) image. If a face is detected in the rotated down image, then, this means that this face 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 face, fast neural networks will be trained only to up examples when using the presented configuration for the input image. 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.

III. Effect of Normalizing the Input Weights on the Speed up Ratio

In the learning phase, all of the training examples are normalized before starting the learning process. There is no problem in normalizing training examples before learning. The problem is how to normalize the input image in the test phase. In [6], the authors stated that image normalization (in the test phase) to avoid weak or strong illumination could not be done in frequency space. This is because the image normalization is local and not easily computed in the Fourier space of the whole image. In a previous paper [4], a simple method for image normalization has been presented. Centering and normalizing the image can be obtained by centering and normalizing the input weights according to the following equation [4]:

$$\overline{X}_{rc} \otimes W_i = X_{rc} \otimes \overline{W}_i \quad (17)$$

which means that cross-correlating a normalized sub-image ($\overline{X}_{rc}$) located at (r,c) in the input image ψ with the weight matrix is equal to the cross-correlation of the non – normalized image with the normalized input weight matrix ($\overline{W}_i$).

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 normalizing the input weights, the speed up ratio for image normalization Γ can be calculated as:

$$\Gamma=\frac{(N-n+1)^2}{q} \quad (18)$$

The speed up ratio of the normalization process for images of different sizes is listed in Table 2. As a result, we may conclude that:

1- Using this technique, normalization in the frequency domain can be done by normalizing the input weights in the spatial domain.
2- Normalization of an image by normalizing the input weights is faster than normalization of each sub-image.
3- Normalization of input 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 to 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:

$$\eta_c = \frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{q(2n^2-1)(N-n+1)^2} \quad (19)$$

which can be simplified to:

$$\eta_c = 1 + \frac{2n^2}{q(2n^2-1)} \quad (20)$$

b) For Fast Neural Networks:

The over all speed up ratio equals to 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 overall speed up ratio η_o can be given by:

$$\eta_o = \frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{((2q+1)(10N^2(\log_2 2N + \log_2 N)) + q(16N^2 - n^2) + 2N)} \quad (21)$$

which can be simplified to:

$$\eta_o = \frac{(N-n+1)^2(q(2n^2-1) + 2n^2)}{((2q+1)(10N^2(\log_2 2N + \log_2 N)) + q(16N^2 - n^2) + 2N)} \quad (22)$$

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

$$\eta_o = \eta + \frac{2n^2(N-n+1)^2}{((2q+1)(10N^2(\log_2 2N + \log_2 N)) + q(16N^2 - n^2) + 2N)} \quad (23)$$

The practical over all speed up ratio with images of different sizes and different sizes of windows is shown in Fig. 6 using 2.80 GHz processor and Matlab ver 5.3. It is clear that the speed up ration is inreased with the size of the sliding window (n). Moreover, the speed up ratio in case of image normalization through normalization of the input weights is larger than the speed up ratio without normalization (which is shown in Fig. 5). This means that the search process with nomalized fast neural networks is done faster than conventional neural networks with or wihout normalization of the input image.

V. Conclusion

Normalized neural networks for fast sub-image (object/face) 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 converting the input image into the presented symmetric form and normalizing the weights of the neural networks. Simulation results have confirmed this by using Matlab. Moreover, by using this new symmetric configuration for the input image, the number of neurons in the hidden layer has been reduced. As a result, fast neural networks, based on cross correlation in the frequency domain, have become faster than conventional neural networks. Furthermore, we have generally proved that, the speed up ratio in the case of image normalization through normalization of weights is faster than without normalization.

References

[1] Hazem M. El-Bakry, and Qiangfu Zhao ”A New Symmetric Form for Fast Sub-Matrix (Object/Face) Detection Using Neural Networks and FFT,” accepted for publication in the International Journal of Signal Processing.

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

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

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

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

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

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

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

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

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

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

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

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

Table 1: A comparison between the number of computation steps (in millions) required for conventional and fast neural networks to manipulate images shown in Fig. 2 with different sizes (n=20).

Image size	Conventional Neural Networks	Fast Neural Networks (2N)	Speed up ratio
100x100	1.5727e+008	1.9609e+008	0.8020
200x200	7.8528e+008	8.8202e+008	0.8903
300x300	1.8927e+009	2.1130e+009	0.8957
400x400	3.4795e+009	3.9185e+009	0.8880
500x500	5.5457e+009	6.3191e+009	0.8776
600x600	8.0913e+009	9.3306e+009	0.8672
700x700	1.1116e+010	1.2966e+010	0.8574
800x800	1.4621e+010	1.7236e+010	0.8483
900x900	1.8605e+010	2.2150e+010	0.8399
1000x1000	2.3068e+010	2.7716e+010	0.8323
1100x1100	2.8010e+010	3.3943e+010	0.8252
1200x1200	3.3432e+010	4.0836e+010	0.8187
1300x1300	3.9334e+010	4.8402e+010	0.8127
1400x1400	4.5715e+010	5.6646e+010	0.8070
1500x1500	5.2575e+010	6.5574e+010	0.8018
1600x1600	5.9914e+010	7.5190e+010	0.7968
1700x1700	6.7733e+010	8.5499e+010	0.7922
1800x1800	7.6032e+010	9.6505e+010	0.7879
1900x1900	8.4810e+010	1.0821e+011	0.7837
2000x2000	9.4067e+010	1.2063e+011	0.7798

Table 2: The speed up ratio of the normalization process for images of different sizes (n=20,q=30).

Image Size	Speed up ratio
100x100	219
200x200	1092
300x300	2632
400x400	4839
500x500	7712
600x600	11252
700x700	15459
800x800	20332
900x900	25872
1000x1000	32079
1100x1100	38952
1200x1200	46492
1300x1300	54699
1400x1400	63572
1500x1500	73112
1600x1600	83319
1700x1700	94192
1800x1800	105730
1900x1900	117940
2000x2000	130810

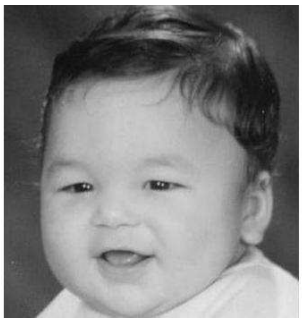

Fig. 1: The original non-symmetric image.

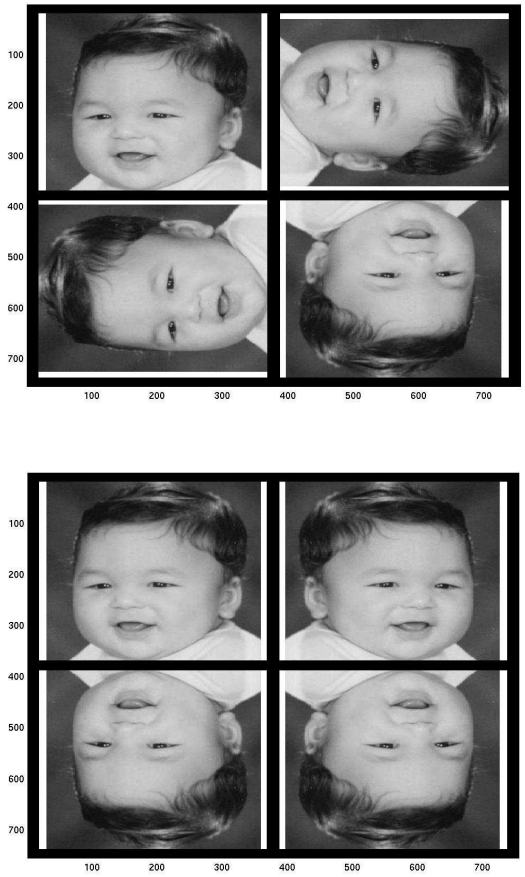

Fig. 2. Image conversion from non-symmetric to symmetric one.

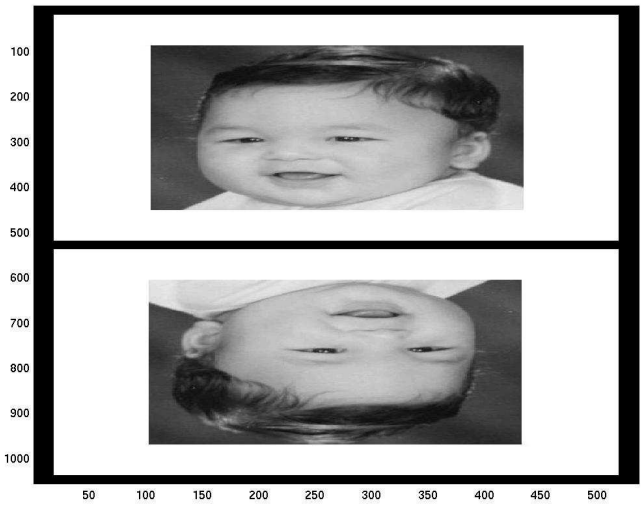

Fig. 3. Image conversion from non-symmetric to symmetric one through rotation into down direction.

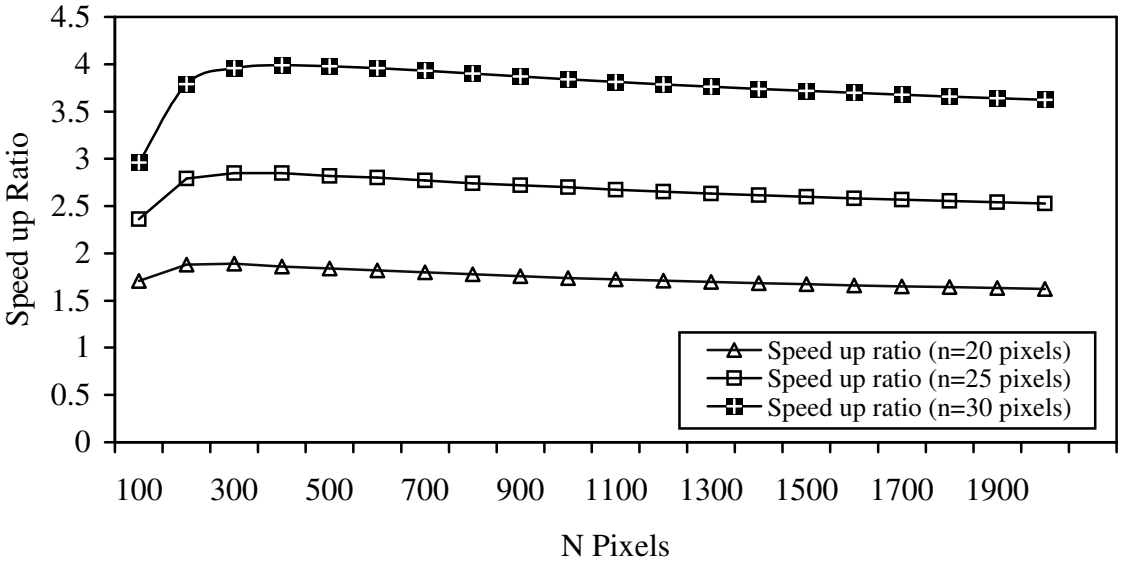

Fig. 4: The theoretical speed up ratio in case of converting an image into symmetric one through rotation into down direction.

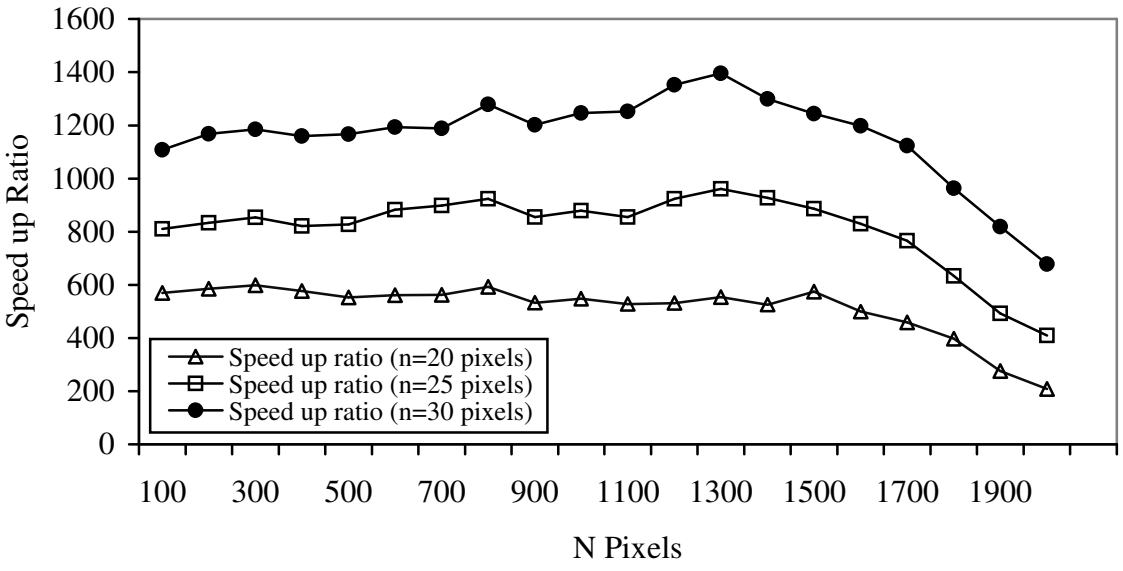

Fig. 5: Simulation results for speed up ratio in case of converting an image into symmetric one through rotation into down direction.

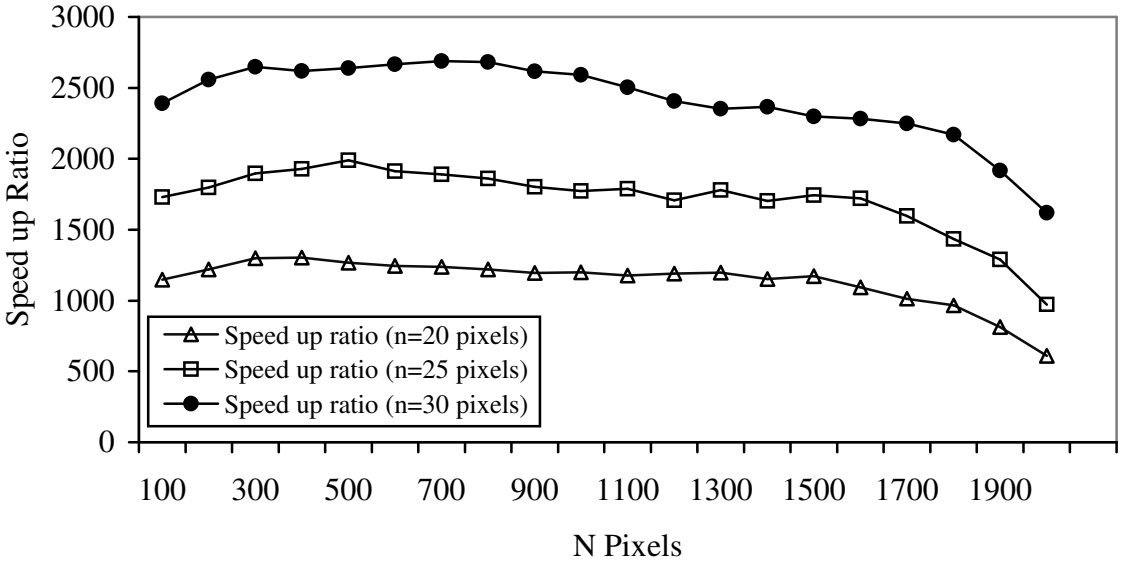

Fig. 6: Simulation results for speed up ratio in case of image normalization by normalizing the input weights.