International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:10, 2008

A Fast Neural Algorithm for Serial Code Detection in a Stream of Sequential Data

Hazem M. El-Bakry, and Qiangfu Zhao

***Abstract*—**In recent years, fast neural networks for object/face detection have been introduced based on cross correlation in the frequency domain between the input matrix and the hidden weights of neural networks. In our previous papers [3,4], fast neural networks for certain code detection was introduced. It was proved in [10] that for fast neural networks to give the same correct results as conventional neural networks, both the weights of neural networks and the input matrix must be symmetric. This condition made those fast neural networks slower than conventional neural networks. Another symmetric form for the input matrix was introduced in [1-9] to speed up the operation of these fast neural networks. Here, corrections for the cross correlation equations (given in [13,15,16]) to compensate for the symmetry condition are presented. After these corrections, it is proved mathematically that the number of computation steps required for fast neural networks is less than that needed by classical neural networks. Furthermore, there is no need for converting the input data into symmetric form. Moreover, such new idea is applied to increase the speed of neural networks in case of processing complex values. Simulation results after these corrections using MATLAB confirm the theoretical computations.

***Keywords*—**Fast Code/Data Detection, Neural Networks, Cross Correlation, real/complex values.

I. Introduction

RECENTLY, neural networks have shown very good results for detecting two dimensional sub-image in a given image [11,12,14]. Some authors tried to speed up the detection process of neural networks [13,15,16]. They proposed a multilayer perceptron (MLP) algorithm for fast object/face detection based on cross correlation in the frequency domain between the input image and the hidden weights of neural networks. Then, they established an equation for the speed up ratio. It was proved in [1-12] that their equations contain many errors, which lead to invalid speed up ratio.

Manuscript received October 21, 2004.

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

Q. Zhao is professor with the Information Systems Department, University of Aizu, Japan (e-mail: qf-zhao@u-aizu.ac.jp).

Here, another error in the definition of cross correlation equation presented in [13,15,16] is presented. In [1-10] a symmetry condition was introduced in both the input matrix (image) and the weights of neural networks to compensate for this error. This symmetry condition allowed those fast neural networks to give the same correct results as conventional neural network for detecting sub-matrix in a given large input matrix. In [3,4], the same principle was used for fast detecting a certain code/data in a given one dimensional matrix (sequential data). This was done by converting the input matrices into symmetric forms. In this paper, corrections for the errors in cross correlation equations introduced in [13,15,16] are presented. Theoretical and practical results after these corrections prove that our proposed fast neural algorithm is faster than the previous algorithms as well as classical neural networks. In section II, fast neural networks for code/data detection are described. The correct fast neural algorithm for detecting a certain code/data in a given one dimensional sequential data is presented in section III. This algorithm can be applied for communication applications. Here, such algorithm is used for increasing the speed of neural networks dealing with complex values. The new fast neural networks with real/complex successive input values will be presented in section IV.

II. Theory of Fast Neural Nets Based on Cross Correlation in the Frequency Domain for Sequential Data Detection

Finding a certain code/data in the input one dimensional matrix is a searching problem. Each position in the input matrix is tested for the presence or absence of the required code/data. At each position in the input matrix, each sub-matrix is multiplied by a window of weights, which has the same size as the sub-matrix. 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-matrix under test contains the required code/data and vice versa. Thus, we may conclude that this searching problem is a cross correlation between the matrix under test and the weights of the hidden neurons.

The convolution theorem in mathematical analysis says that a convolution of f with h is identical to the result of the following steps: let F and H be the results of the Fourier Transformation of f and h in the frequency domain. Multiply F and H in the frequency domain point by point and then transform this product into 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, speed up in an order of magnitude can be achieved during the detection process [1,2,3,4,5,7,8,9]. In the detection phase, a sub matrix I of size 1xn (sliding window) is extracted from the tested matrix, which has a size 1xN, and fed to the neural network. Let X_i be the vector of weights between the input sub-matrix and the hidden layer. This vector has a size of 1xn and can be represented as 1xn matrix. The output of hidden neurons h(i) can be calculated as follows:

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

where g is the activation function and b(i) is the bias of each hidden neuron (i). Equation 1 represents the output of each hidden neuron for a particular sub-matrix I. It can be obtained to the whole matrix Z as follows:

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

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

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

Therefore, Eq.2 can be written as follows [1]:

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

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

Now, the above cross correlation can be expressed in terms of one dimensional Fast Fourier Transform as follows [1]:

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

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:

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

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

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

1- For a tested matrix of 1xN elements, the 1D-FFT requires a number equal to $N\log_2 N$ of complex computation steps. Also, the same number of complex computation steps is required for computing the 1D-FFT of the weight matrix at each neuron in the hidden layer.

2- At each neuron in the hidden layer, the inverse 1D-FFT is computed. Therefore, q backward and (1+q) forward transforms have to be computed. Therefore, for a given matrix under test, the total number of operations required to compute the 1D-FFT is $(2q+1)N\log_2 N$.

3- 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 one dimensional Fast Fourier Transform requires $(N/2)\log_2 N$ complex multiplications and $N\log_2 N$ 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. Therefore, the total number of computation steps required to obtain the 1D-FFT of a 1xN matrix is:

$$\rho = 6((N/2)\log_2 N) + 2(N\log_2 N) \quad (7)$$

which may be simplified to:

$$\rho = 5(N\log_2 N) \quad (8)$$

4- Both the input and the weight matrices should be dot multiplied in the frequency domain. Thus, a number of complex computation steps equal to qN should be considered. This means 6qN real operations will be added to the number of computation steps required by fast neural networks.

5- In order to perform cross correlation in the frequency domain, the weight matrix must be extended to have the same size as the input matrix. So, a number of zeros = (N-n) must be added to the weight matrix. This requires a total real number of computation steps = q(N-n) for all neurons. Moreover, after computing the FFT for the weight matrix, the conjugate of this matrix must be obtained. As a result, a real number of computation steps =qN 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 matrix or by previous complex numbers during the computation of FFT. To create a complex number requires two real floating point operations. Thus, the total number of computation steps required for fast neural networks becomes:

$$\sigma=((2q+1)(5N\log_2 N)+6qN+q(N-n)+qN+N) \quad (9)$$

which can be reformulated as:

$$\sigma=((2q+1)(5N\log_2 N)+q(8N-n)+N) \quad (10)$$

6- Using sliding window of size 1xn for the same matrix of 1xN pixels, (q(2n-1)(N-n+1)) computation steps are required when using classical neural networks for certain code detection. The theoretical speed up factor η can be evaluated as follows [11]:

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

7- But as proved in [10], this cross correlation in the frequency domain (Fast Neural Networks) gives the same correct results as conventional cross correlation (Conventional Neural Networks) only when the input matrices are symmetric.

II. Fast Neural Networks for Detecting a Certain Code in a Stream of One Dimensional Serial Data

In [3,4], another symmetric form for the input one dimensional matrix so that fast neural networks can give the same correct results as conventional neural networks. The input matrix is converted into symmetric form by obtaining its mirror and testing both the matrix and its mirror version as a one (symmetric) matrix consists of two matrices. In this case, the symmetric matrix will have dimensions of (1x2N). Assume that the original input matrix (1xN dimensions) X_o is:

$$X_o = [x_1, x_2, \ldots\ldots, x_N] \quad (12)$$

Then the symmetric matrix X_s (1x2N dimensions) after conversion from non-symmetric to symmetric one will be:

$$X_s = [x_1, x_2, \ldots\ldots, x_N, x_N, \ldots\ldots, x_2, x_1] \quad (13)$$

By substituting in Eq.9 for the new dimensions, the number of computation steps required for cross correlating this new matrix with the weights in the frequency domain can be calculated as follows [3,4]:-

$$\sigma=((2q+1)(5(2N\log_2 2N))+6q(2N)+q(2N-n)+q(2N) +2N) \quad (14)$$

which can be simplified to:

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

So, the speed up ratio in this case can be calculated as [3,4]:

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

The theoretical speed up ratio in this case with different sizes of the input matrix and different in size weight matrices is shown in Fig.1. Also, practical speed up ratio for manipulating matrices of different sizes and different in size weight matrices is shown in Fig.2 using 700 MHz processor and MATLAB.

There are critical errors in cross correlation equations presented in [13,15,16]. Eq.3 and Eq.4 (which is Eq.4 in [16] and also Eq.13 in [15] but in two dimensions) are not correct. Eq.3 is not correct because the definition of cross correlation is:

$$f(x)\otimes d(x)=\left(\sum_{n=-\infty}^{\infty} f(x+n)d(n)\right) \quad (17)$$

and then Eq.4 must be written as follows:

$$h_i = g\left(Z \otimes X_i + b_i\right) \quad (18)$$

Therefore, the cross correlation in the frequency domain given by Eq.5 does not represent Eq.4. 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.4 does not give the same correct results as conventional neural networks. This error leads the researchers in [1-10] who consider the references

[13,15,16] to think about how to modify the operation of cross correlation so that Eq.4 can give the same correct results as conventional neural networks. Therefore, the errors in these equations must be cleared to all the researchers. In [1-10], the authors proved that a symmetry condition must be found in input matrices (the input matrix under test 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 the input matrices.

The correct theoretical speed up ratio, given by Eq.11, with different sizes of the input matrix and different in size weight matrices is listed in Fig.3. Practical speed up ratio for manipulating matrices of different sizes and different in size weight matrices is listed in Fig. 4 using 700 MHz processor and MATLAB ver 5.3. In this case, both theoretical and practical results show that speed up ratios are faster than the previous speed up ratios listed in Figures 1 and 2. For different lengths of the detected code (n), a comparison between the numbers of computation steps required by fast and conventional neural networks is shown in Figures 5, 6 and 7. It is clear that the number of computation steps required by conventional neural networks is much more than that needed by fast neural networks. Furthermore, as shown in these three figures, the number of computation steps required by fast neural networks is the same. This is because the length of the code (n), which is required to be detected, does not affect the number of computation steps required by fast neural networks (Eq.10). Moreover, the number of computation steps required by conventional neural networks is increased with the length of the code (n). Thus, as shown in Figures 3 and 4, the speed up ratio is increased with the length of the input matrix.

IV. A New Fast Complex-Valued Code Detection Algorithm using Neural Networks

Detection of complex values has many applications especially in fields dealing with complex numbers such as telecommunications, speech recognition and image processing with the Fourier Transform [17,18]. For neural networks, processing complex values means that the inputs, weights, thresholds and activation function have complex values. For sequential data, neural networks accept serial input data with fixed size (n). Therefore, the number of input neurons equals to (n). Instead of treating (n) inputs, our new idea is to collect the input data together in a long vector (for example 100xn). Then the successive input data is tested by fast neural networks as a single pattern with length L (for example L=100xn). Such test is performed in the frequency domain as described in section II. In this section, formulas for the speed up ratio with different types of inputs will be presented. The special case when the imaginary part of the inputs=0 (i.e. real input values) is considered. Also, the speed up ratio in case of one and two dimensional input matrix will be concluded. The operation of fast neural networks depends on computing the Fast Fourier Transform for both the input and weight matrices and obtaining the resulted two matrices. After performing dot multiplication for the resulted two matrices in the frequency domain, the Inverse Fast Fourier Transform is calculated for the final matrix. As the Fast Fourier Transform is already dealing with complex numbers, so there is no change in the number of computation steps required by fast neural networks. Therefore, the speed up ratio in case of fast neural networks dealing with different types of inputs can be evaluated as follows:

1) In case of real inputs and complex weights

A) For one dimensional input matrix

Multiplication of (n) complex-valued weights by (n) real inputs requires (2n) real operations. This produces (n) real numbers and (n) imaginary numbers. The addition of these numbers requires (2n-2) real operations. Therefore, the number of computation steps required by conventional neural networks can be calculated as:

$$\theta = 2q(2n-1)(N-n+1) \tag{19}$$

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

$$\eta = \frac{2q(2n-1)(N-n+1)}{(2q+1)(5N\log_2 N)+q(8N-n)+N} \tag{20}$$

A comparison between the numbers of computation steps required by conventional and fast neural networks with different code size is shown in Figures 8,9,10. The theoretical speed up ratio for searching of a short successive code of length (n) in a long input vector (L) using fast neural networks is shown in Fig.11. Also, practical speed up ratio for manipulating matrices of different sizes (L) and different in size complex-valued weight matrices (n) is shown in Fig.12 using 700 MHz processor and MATLAB. It is clear that the speed up ratio is proportionally increased with the size of the code to be detected. This is very important for fast detecting large size codes. Such result proves that the proposed algorithm is a good achievement.

B) For two dimensional input matrix

Multiplication of (n^2) complex-valued weights by (n^2) real inputs requires ($2n^2$) real operations. This produces (n^2) real numbers and (n^2) imaginary numbers. The addition of these numbers requires ($2n^2$-2) real operations. Therefore, the number of computation steps required by conventional neural networks can be calculated as:

$$\theta=2q(2n^2-1)(N-n+1)^2 \quad (21)$$

For two dimensional input matrix, the number of computation steps (σ) required by fast neural networks can be calculated as [19-23]:

$$\sigma=(2q+1)(5N^2\log_2N^2)+q(8N^2-n^2)+N \quad (22)$$

Therefore, the speed up ratio in this case can be computed as follows:

$$\eta=\frac{2q(2n^2-1)(N-n+1)^2}{(2q+1)(5N^2\log_2N^2)+q(8N^2-n^2)+N} \quad (23)$$

A comparison between the numbers of computation steps required by conventional and fast neural networks for detecting (nxn) real-valued sub-matrix in a large real-valued matrix (NxN) is shown in Figures 13,14,15. The theoretical speed up ratio for detecting (nxn) real-valued sub-matrix in a large real-valued matrix (NxN) using fast neural networks is shown in Fig.16. Also, practical speed up ratio for manipulating real-valued matrices of different sizes (NxN) and different in size complex-valued weight matrices (n) is shown in Fig.17 using 700 MHz processor and MATLAB.

2) In case of complex inputs and weights

A) For one dimensional input matrix

Multiplication of (n) complex-valued weights by (n) complex inputs requires (6n) real operations. This produces (n) real numbers and (n) imaginary numbers. The addition of these numbers requires (2n-2) real operations. Therefore, the number of computation steps required by conventional neural networks can be calculated as:

$$\theta=2q(4n-1)(N-n+1) \quad (24)$$

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

$$\eta=\frac{2q(4n-1)(N-n+1)}{(2q+1)(5N\log_2N)+q(8N-n)+N} \quad (25)$$

A comparison between the numbers of computation steps required by conventional and fast neural networks with different code size is shown in Figures 18,19,20. The theoretical speed up ratio for searching of a short complex-valued successive code of length (n) in a long complex-valued input vector (L) using fast neural networks is shown in Fig.21. Also, practical speed up ratio for manipulating complex-valued matrices of different sizes (L) and different in size complex-valued weight matrices (n) is shown in Fig.22 using 700 MHz processor and MATLAB.

B) For two dimensional input matrix

Multiplication of (n^2) complex-valued weights by (n^2) real inputs requires ($6n^2$) real operations. This produces (n^2) real numbers and (n^2) imaginary numbers. The addition of these numbers requires ($2n^2$-2) real operations. Therefore, the number of computation steps required by conventional neural networks can be calculated as:

$$\theta=2q(4n^2-1)(N-n+1)^2 \quad (26)$$

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

$$\eta=\frac{2q(4n^2-1)(N-n+1)^2}{(2q+1)(5N^2\log_2N^2)+q(8N^2-n^2)+N} \quad (27)$$

A comparison between the numbers of computation steps required by conventional and fast neural networks for detecting (nxn) complex-valued sub-matrix in a large complex-valued matrix (NxN) is shown in Figures 23, 24, and 25. The theoretical speed up ratio for detecting (nxn) complex-valued sub-matrix in a large complex-valued matrix (NxN) using fast neural networks is shown in Fig.26. Also, practical speed up ratio for manipulating complex-valued matrices of different sizes (NxN) and different in size complex-valued weight matrices (n) is shown in Fig.27 using 700 MHz processor and MATLAB.

For one dimensional matrix, from Figures 8,9,10,11,12,18,19,20,21,22, we can conclude that the response time for vectors with small lengths are faster than those which have larger lengths. For example, the speed up ratio for the vector of length 10000 is faster that of length 1000000. The number of computation steps required for a vector of length 10000 is much less than that required for a vector of length 40000. So, if the vector of length 40000 is divided into 4 shorter vectors of length 10000, the number of computation steps will be less than that required for the vector of length 40000. Therefore, for each application, it is useful at the first to calculate the optimum length of the input vector. The same conclusion can be drawn in case of processing the two dimensional input matrix. From Figures 13,14,15,16,17,23,24,25,26,27, it is clear that the maximum speed up ratio is achieved at matrix size (N=200x200) when n=20, then at matrix size (N=300x300) when n=25, and at matrix size (N=400x400) when n=30. This confirms our previous results presented in [7] on fast sub-image detection based on neural networks and image decomposition. Using such technique, it was proved that the speed up ratio of neural networks becomes faster when dividing the input image into many sub-images and processing each sub-image in the

frequency domain separately using a single fast neural processor.

V. Conclusion

A fast neural algorithm for detecting a certain code/data in a given sequential data has been presented. A new correct formula for the speed up ratio has been established. Furthermore, correct equations for one dimensional cross correlation in the spatial and frequency domains have been presented. Moreover, commutative cross correlation in one dimension has been achieved by converting the non-symmetric input matrices into symmetric forms. Theoretical computations after these corrections have shown that fast neural networks requires fewer computation steps than conventional one. Simulation results have confirmed this approval by using MATLAB. The presented fast neural networks have been applied successfully to detect a small matrix with real/complex values in a given large input one/two dimensional matrix. Therefore, such algorithm can be used for communication applications such as detecting certain serial code in a given large stream of sequential input data.

References

[1] H. M. El-Bakry, and Q. Zhao, "*A New Symmetric Form for Fast Sub-Matrix (Object/Face) Detection Using Neural Networks and FFT*," International Journal of Signal Processing, to be published.

[2] H. M. El-Bakry, and Q. Zhao, "*Fast Pattern Detection Using Normalized Neural Networks and Cross Correlation in the Frequency Domain*," EURASIP Journal on Applied Signal Processing, to be published.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[17] A. Hirose, Complex-Valued Neural Networks Theories and Applications, Series on innovative Intellegence, vol.5. Nov. 2003.

[18] S. Jankowski, A. Lozowski, and M. Zurada, "*Complex-valued Multistate Neural Associative Memory*," IEEE Trans. on Neural Networks, vol.7, 1996, pp.1491-1496.

[19] H. M. El-Bakry, and Q. Zhao, "*Fast Object/Face Detection Using Neural Networks and Fast Fourier Transform*," International Journal on Signal Processing, vol.1, no.3, pp. 182-187, 2004.

[20] H. M. El-Bakry, and Q. Zhao, "*A Modified Cross Correlation in the Frequency Domain for Fast Pattern Detection Using Neural Networks*," International Journal on Signal Processing, vol.1, no.3, pp. 188-194, 2004.

[21] Hazem M. El-Bakry, and Q. Zhao, "*Face Detection Using Fast Neural Processors and Image Decomposition*," International Journal on Computational Intelligence, vol.1, no.4, pp. 313-316, 2004.

[22] H. M. El-Bakry, and Q. Zhao, "*Fast Pattern Detection Using Neural Networks Realized in Frequency Domain*," Proc. of the International Conference on Pattern Recognition and Computer Vision, The Second World Enformatika Conference WEC'05, Istanbul, Turkey, 25-27 Feb., 2005, pp.89-92.

[23] H. M. El-Bakry, and Q. Zhao, "*Sub-Image Detection Using Fast Neural Processors and Image Decomposition*," Proc. of the International Conference on Pattern Recognition and Computer Vision, The Second World Enformatika Conference WEC'05, Istanbul, Turkey, 25-27 Feb., 2005, pp.85-88

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

His research interests include neural networks, pattern recognition, image processing, biometrics, cooperative intelligent systems and electronic circuits. In these areas, he has published more than 39 papers as a single author in major international journals and conferences. He is the first author in 10 refereed international journal papers and more than 70 refereed international conference papers.

Eng. El-Bakry has the patent No. 2003E 19442 DE HOL / NUR, Magnetic Resonance, SIEMENS Company, Erlangen, Germany, 2003. He is a referee for the International Journal of Machine Graphics & Vision and many different international conferences. He was selected as a chairman for the Facial Image Processing Session in the 6th International Computer Science Conference, Active Media Technology (AMT) 2001, Hong Kong, China, December 18-20, 2001 and for the Genetic Programming Session, in ACS/IEEE International Conference on Computer Systems and Applications

Lebanese American University Beirut, Lebanon, June 25-29, 2001. He was invited for a talk in the Biometric Consortium, Orlando, Florida, USA, 12-14 Sep. 2001, which co-sponsored by the United States National Security Agency (NSA) and the National Institute of Standards and Technology (NIST).

Dr. Zhao received the Ph. D degree from Tohoku University of Japan in 1988. He joined the Department of Electronic Engineering of Beijing Institute of Technology of China in 1988, first as a post doctoral fellow and then associate professor. He was associate professor from Oct. 1993 at the Department of Electronic Engineering of Tohoku University of Japan. He joined the University of Aizu of Japan from April 1995 as associate professor, and became tenure full professor in April 1999. Prof. Zhao research interests include image processing, pattern recognition and understanding, computational intelligence, neurocomputing and evolutionary computation.

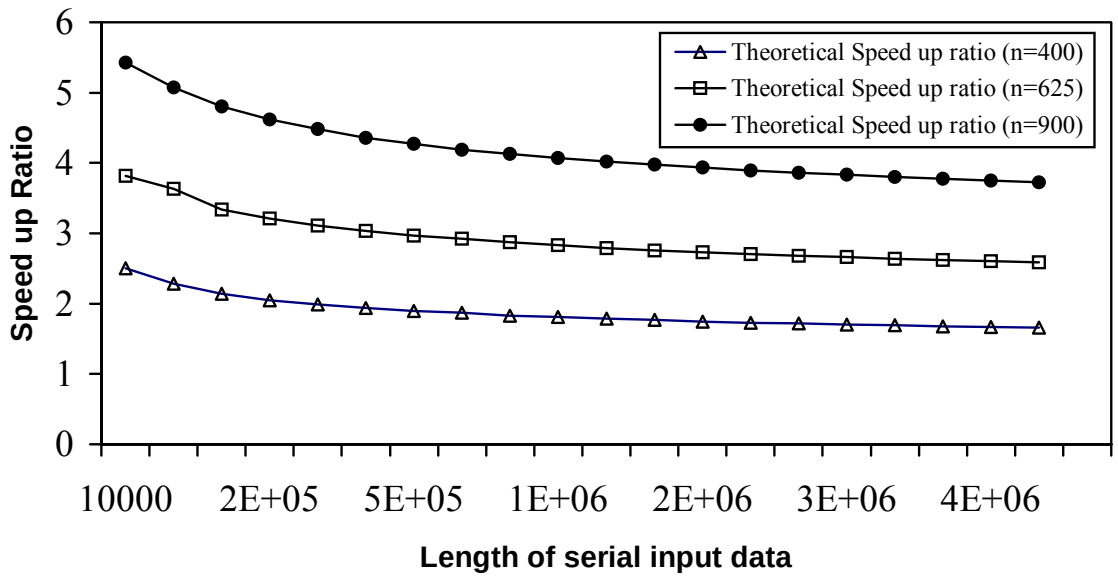

Fig. 1 The theoretical speed up ratio in case of converting sequential data into symmetric form by mirroring the input matrix

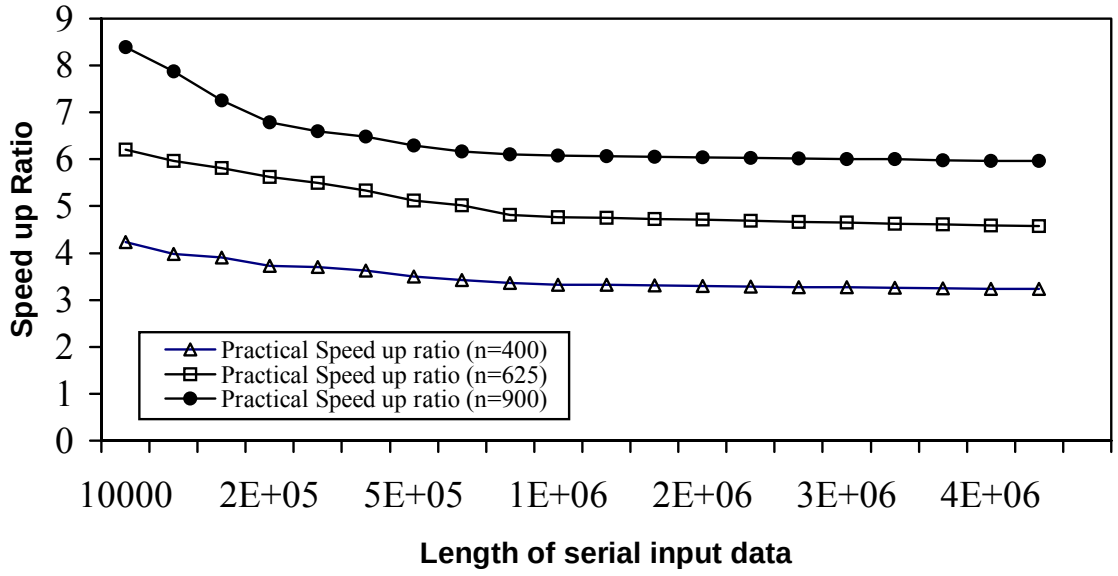

Fig. 2 Practical simulation results for speed up ratio in case of converting sequential data into symmetric form by mirroring the input matrix

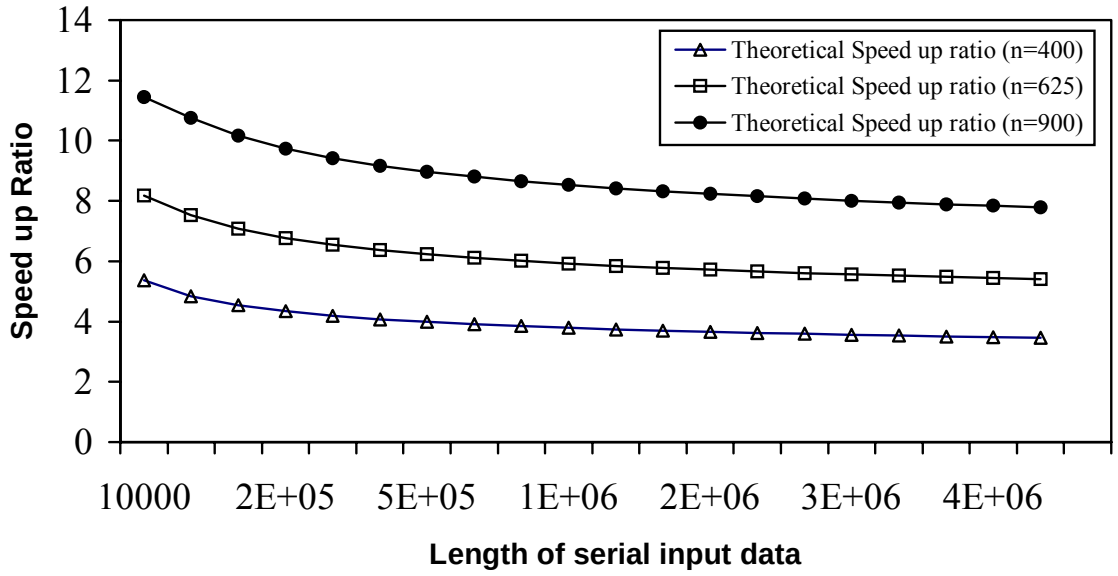

Fig. 3 The theoretical speed up ratio for detecting a certain code (n) in a stream of serial data

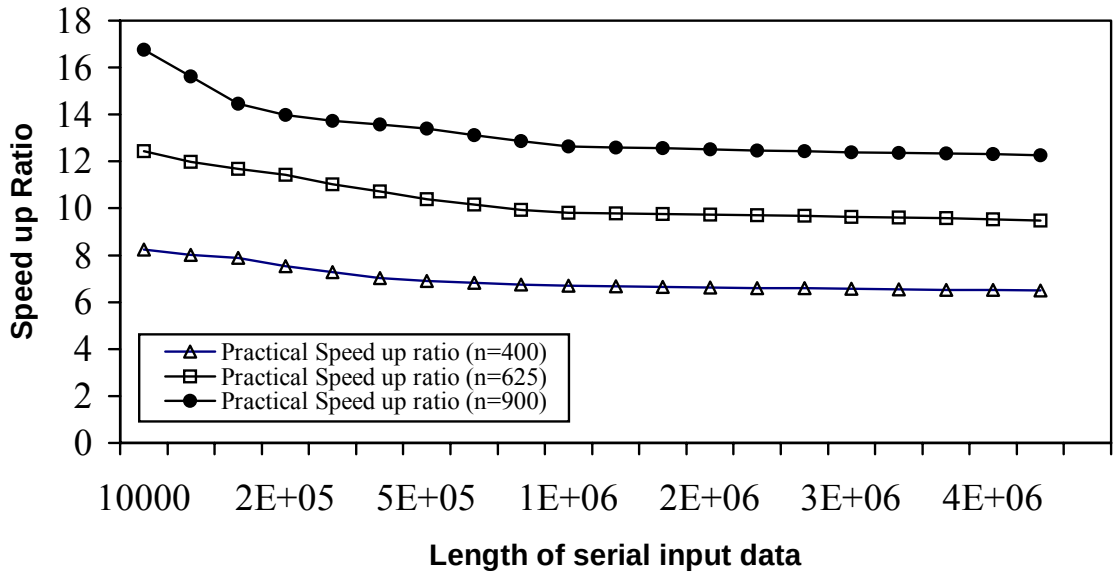

Fig. 4 Practical speed up ratio for detecting a certain code (n) in a stream of serial data

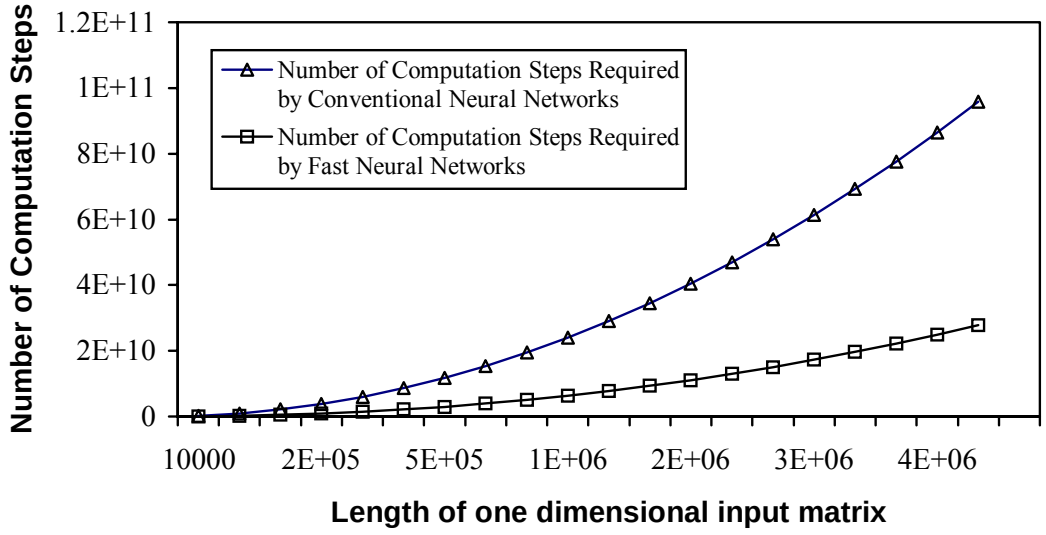

Fig. 5 A comparison between the numbers of computation steps required by fast and conventional neural networks (n=400)

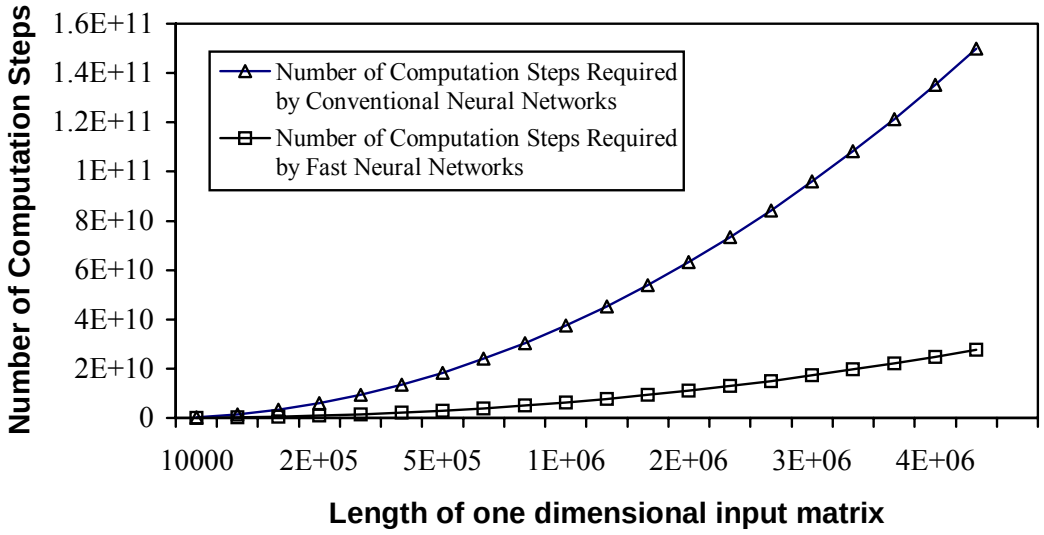

Fig. 6 A comparison between the numbers of computation steps required by fast and conventional neural networks (n=625)

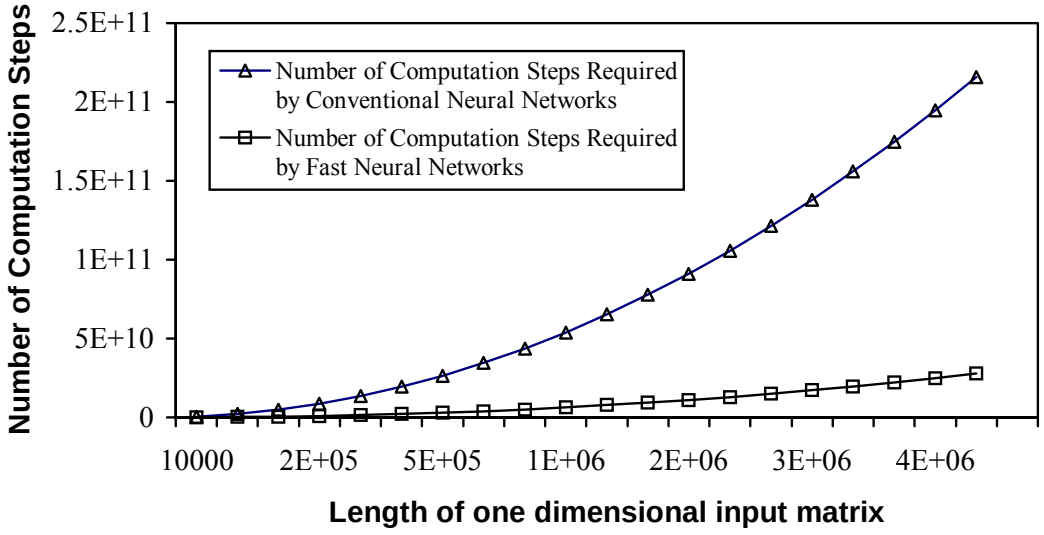

Fig. 7 A comparison between the numbers of computation steps required by fast and conventional neural networks (n=900)

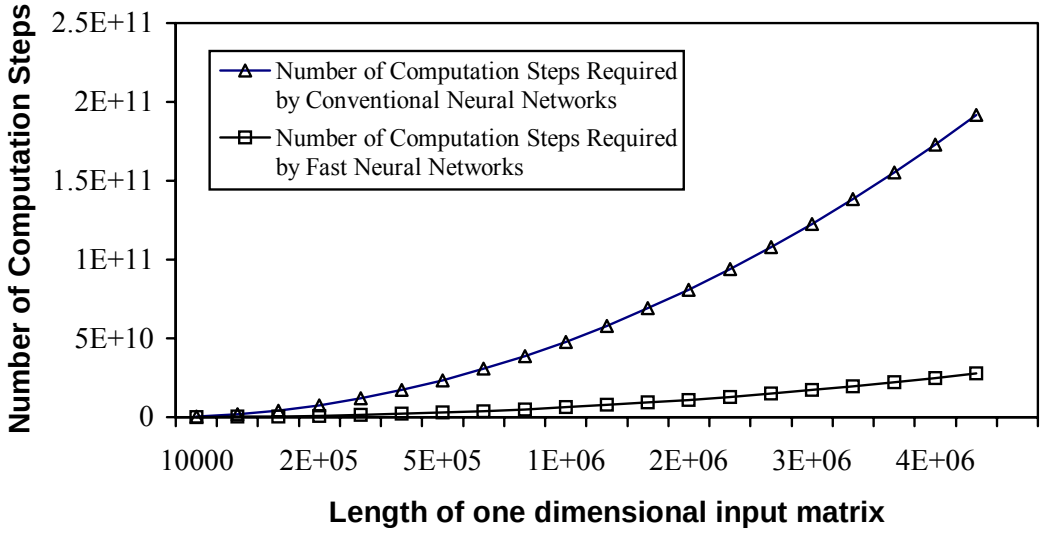

Fig. 8 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of real-valued one dimensional input matrix and complex-valued weight matrix (n=400)

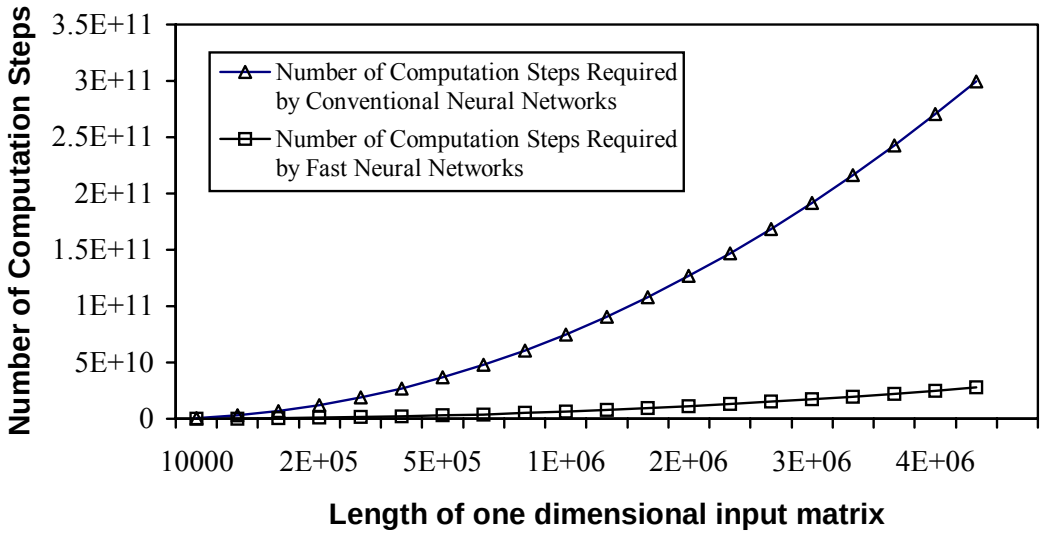

Fig. 9 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of real-valued one dimensional input matrix and complex-valued weight matrix (n=625)

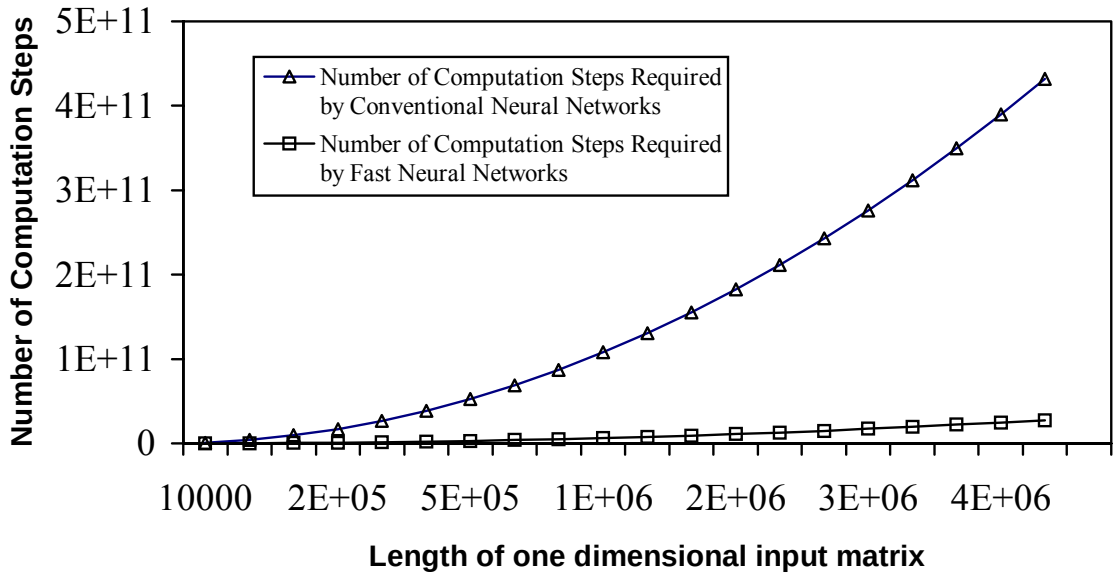

Fig. 10 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of real-valued one dimensional input matrix and complex-valued weight matrix (n=900)

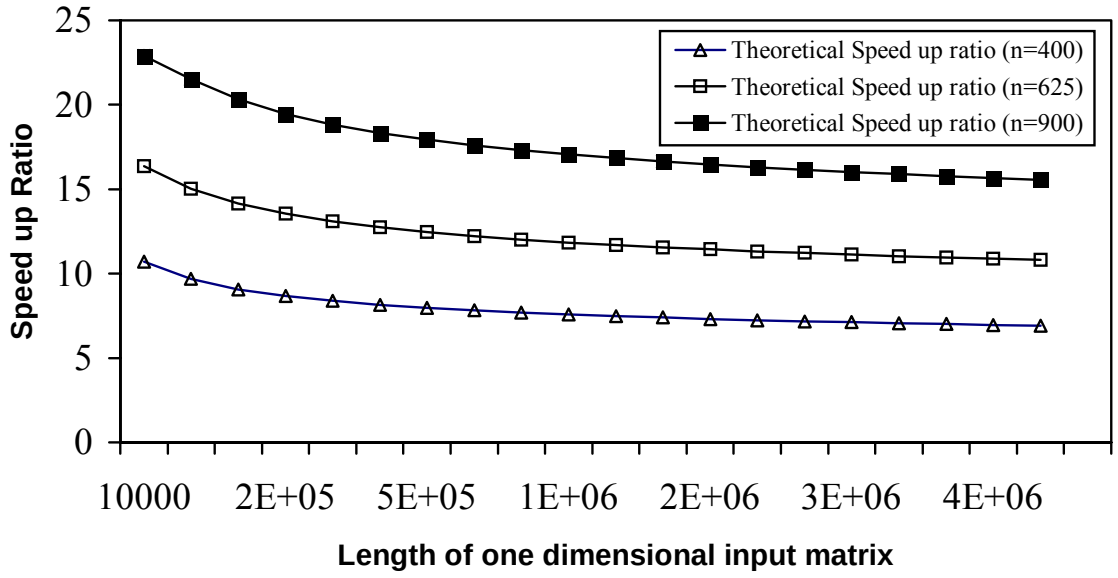

Fig. 11 The relation between the speed up ratio and the length of one dimensional real-valued input matrix in case of complex-valued weights

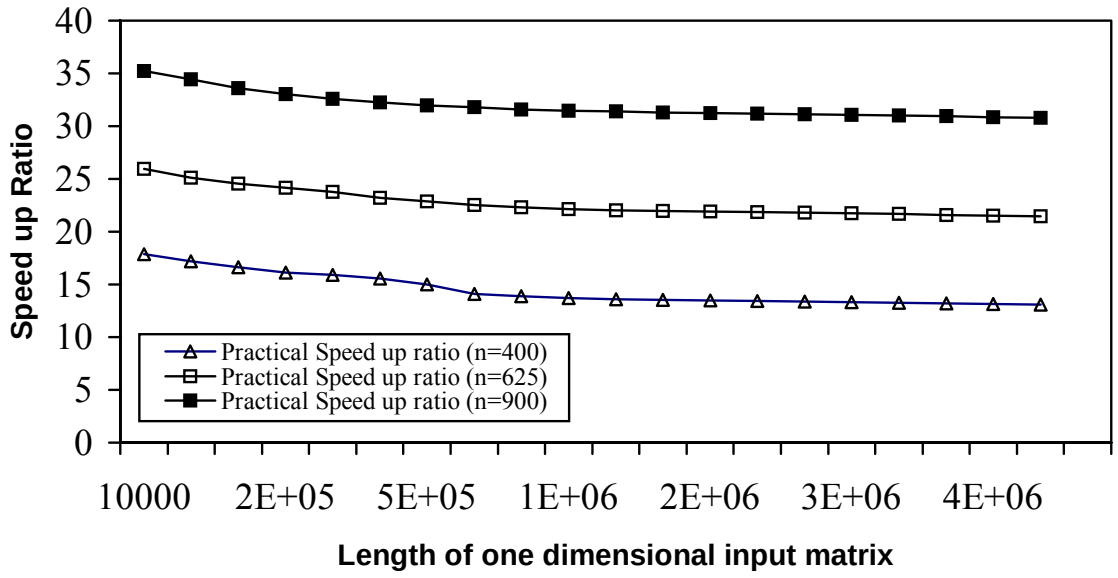

Fig. 12 Practical speed up ratio in case of one dimensional real-valued input matrix and complex-valued weights

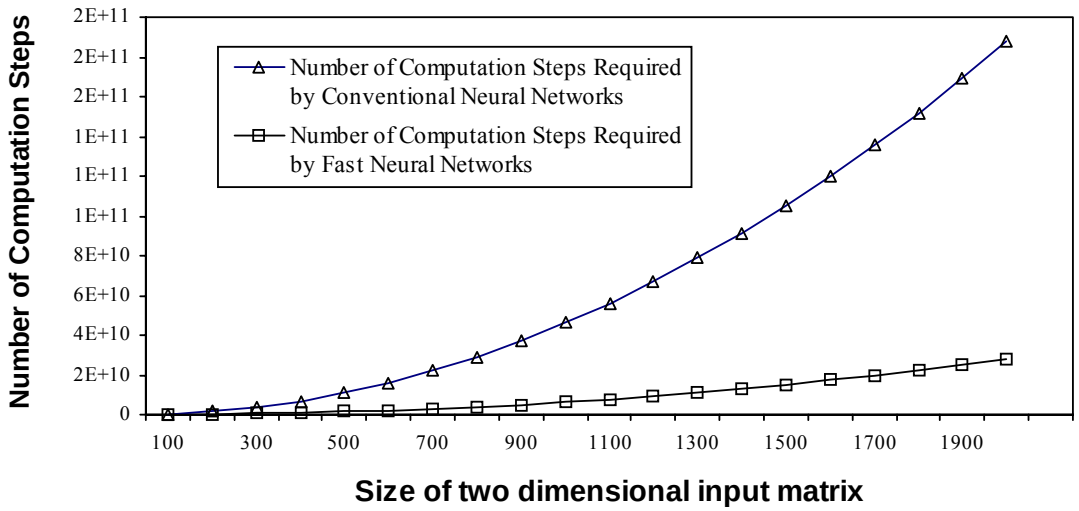

Fig. 13 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of real-valued two dimensional input matrix and complex-valued weight matrix (n=20)

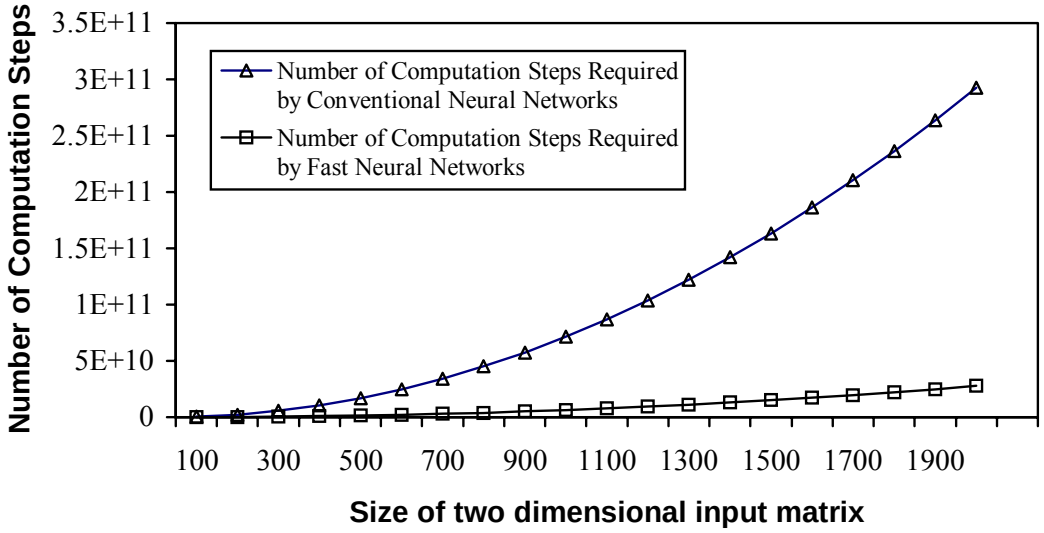

Fig. 14 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of real-valued two dimensional input matrix and complex-valued weight matrix (n=25)

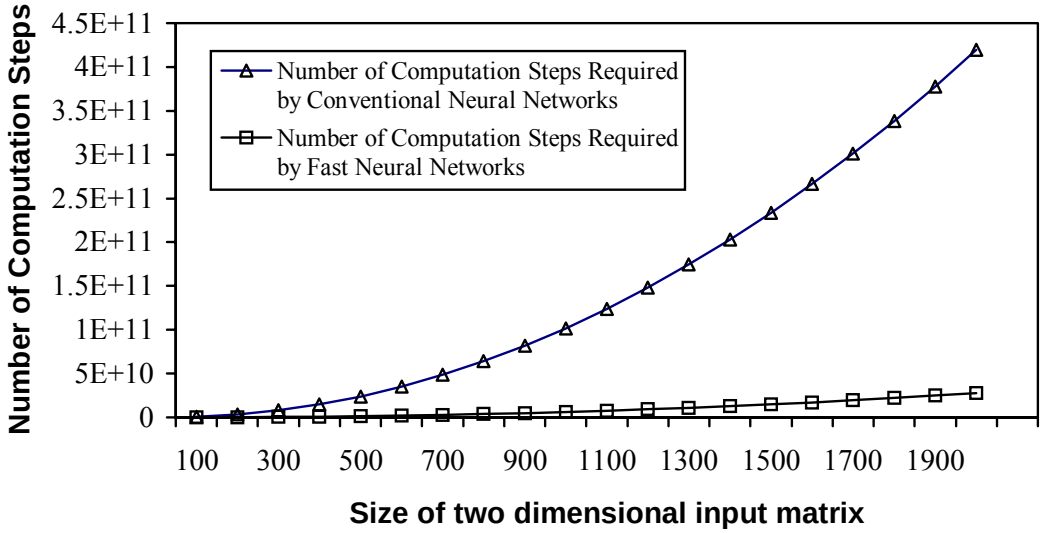

Fig. 15 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of real-valued two dimensional input matrix and complex-valued weight matrix (n=30)

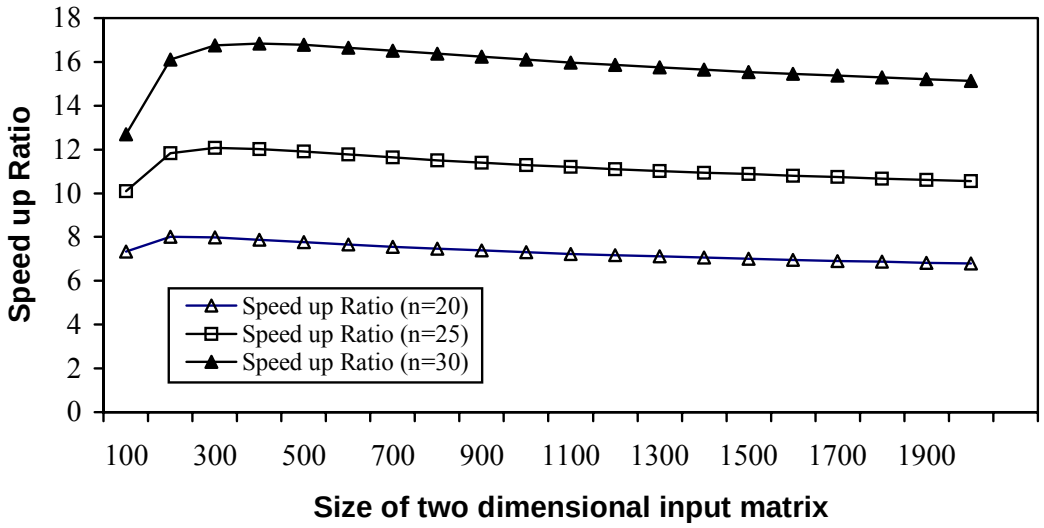

Fig. 16 The relation between the speed up ratio and the size of two dimensional real-valued input matrix in case of complex-valued weights

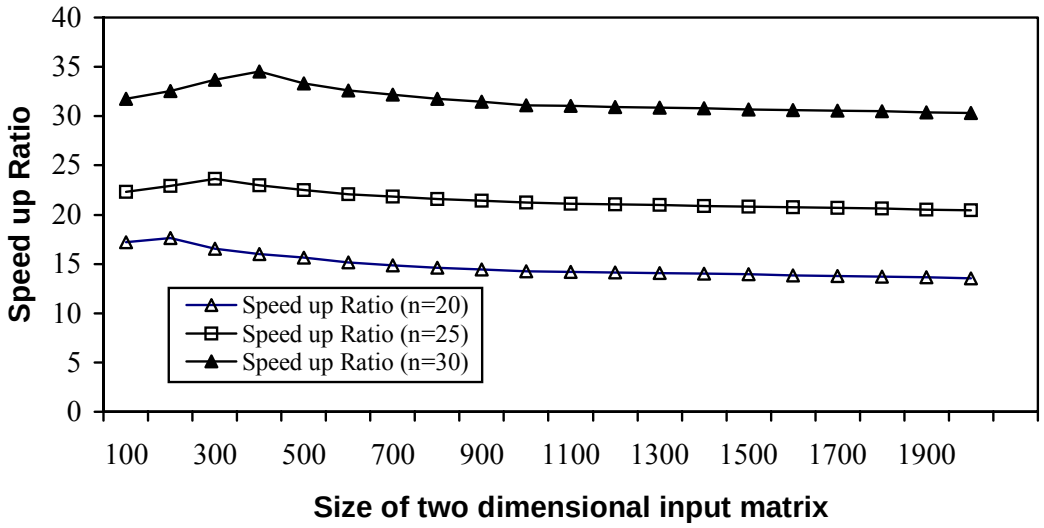

Fig. 17 Practical speed up ratio in case of two dimensional real-valued input matrix and complex-valued weights

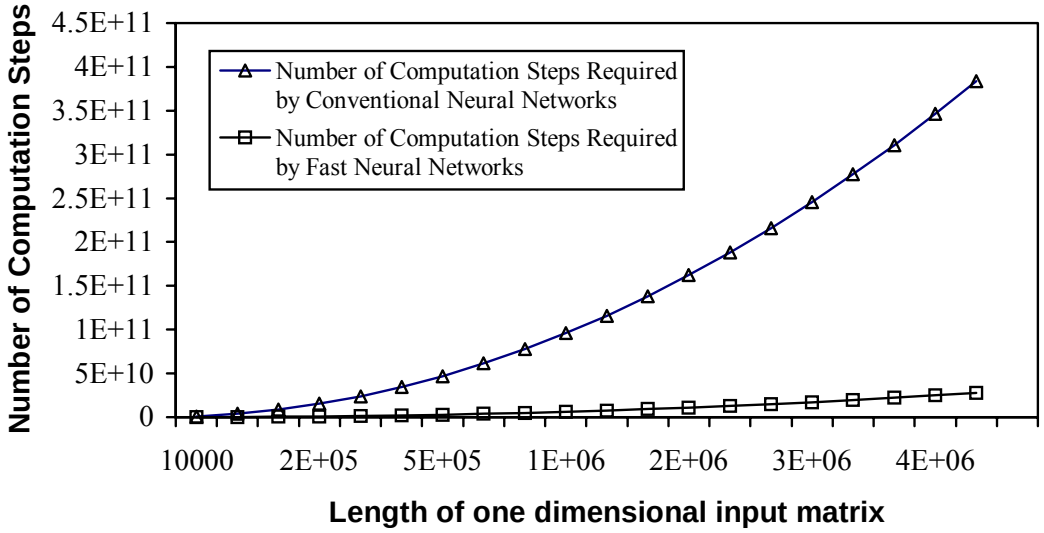

Fig. 18 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of complex-valued one dimensional input matrix and complex-valued weight matrix (n=400)

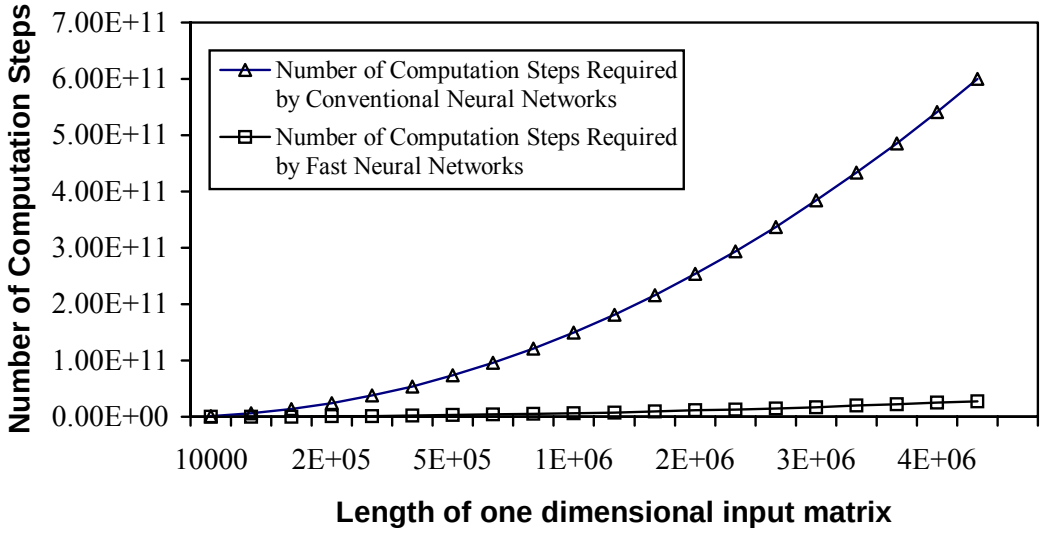

Fig. 19 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of complex-valued one dimensional input matrix and complex-valued weight matrix (n=625)

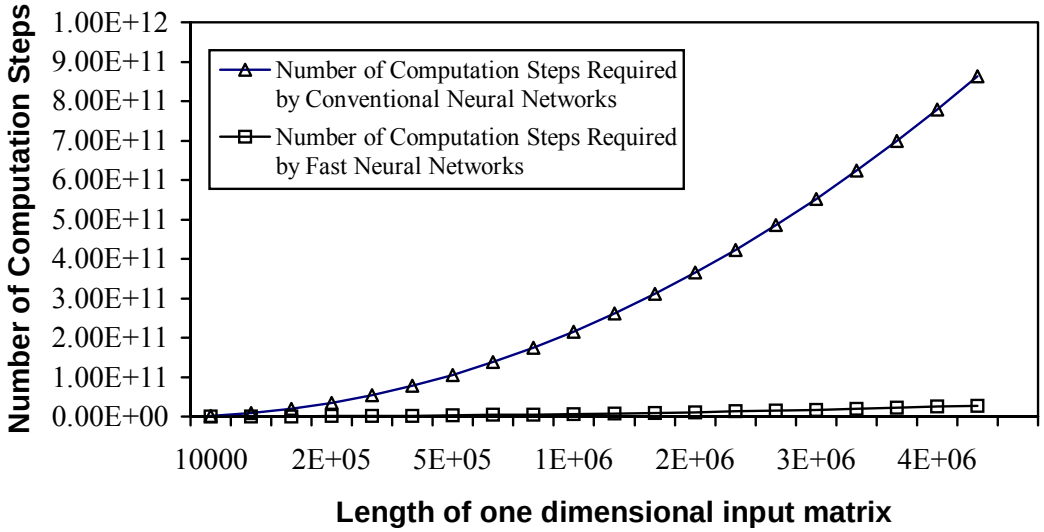

Fig. 20 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of complex-valued one dimensional input matrix and complex-valued weight matrix (n=900)

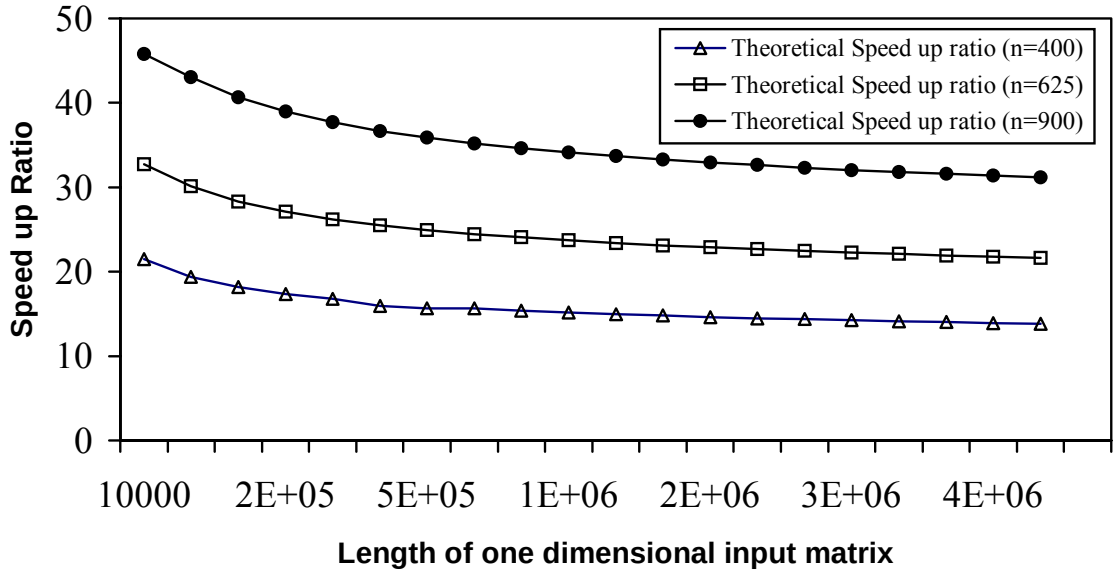

Fig. 21 The relation between the speed up ratio and the length of one dimensional input real-valued matrix in case of complex-valued weights

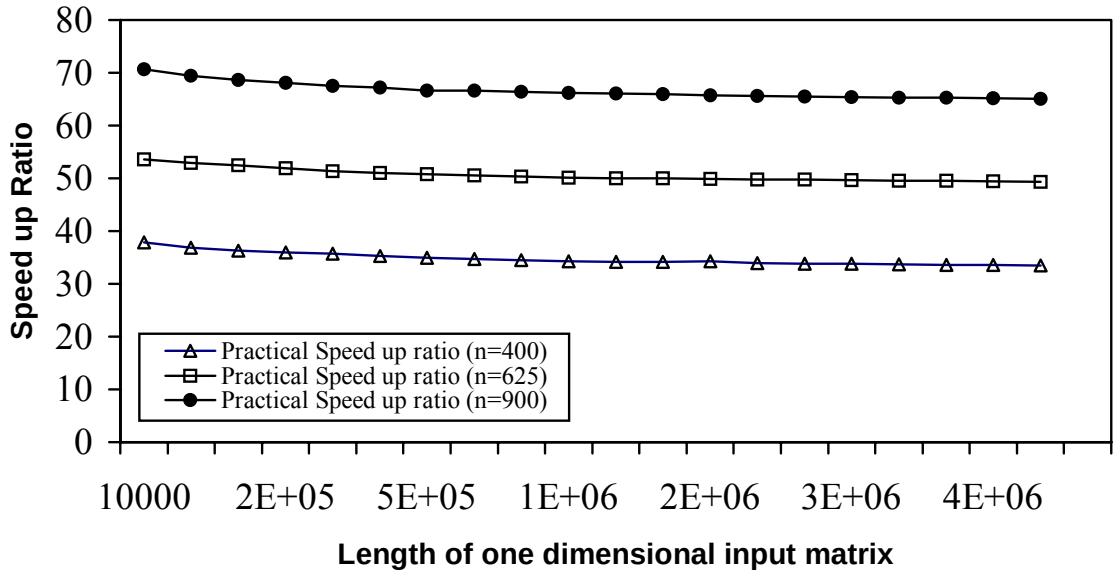

Fig. 22 Practical speed up ratio in case of one dimensional complex-valued input matrix and complex-valued weights

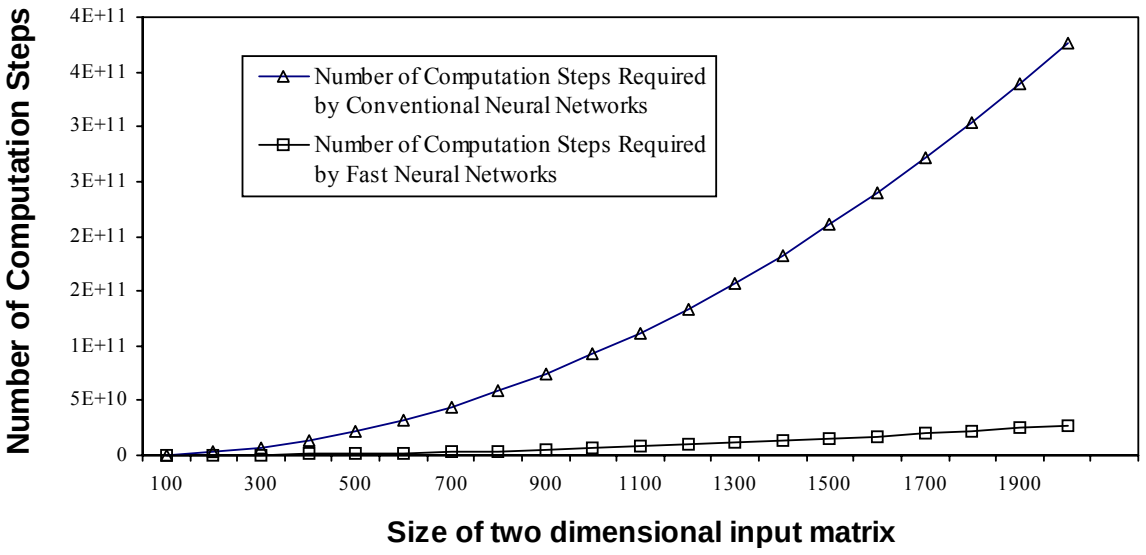

Fig. 23 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of complex-valued two dimensional input matrix and complex-valued weight matrix (n=20)

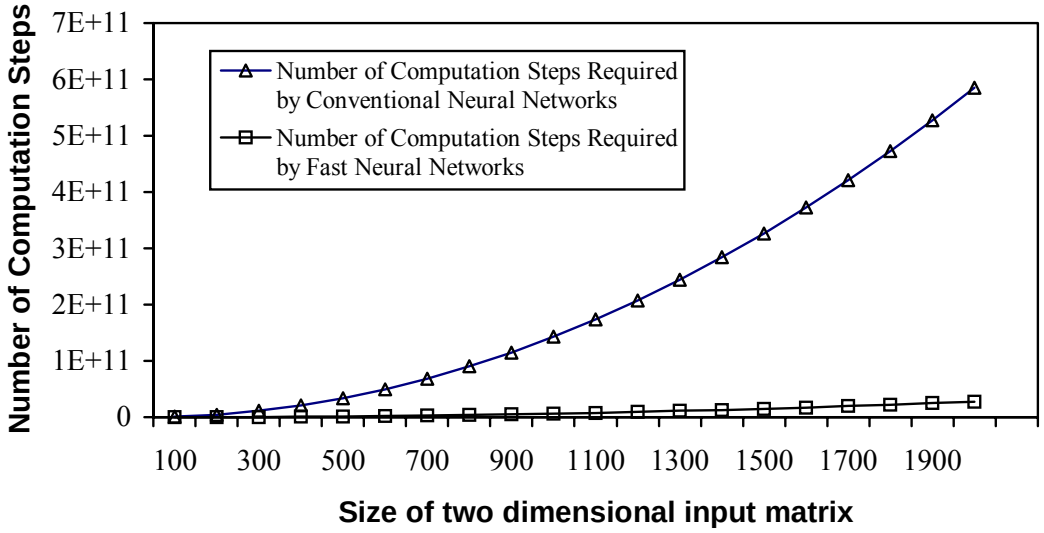

Fig. 24 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of complex-valued two dimensional input matrix and complex-valued weight matrix (n=25)

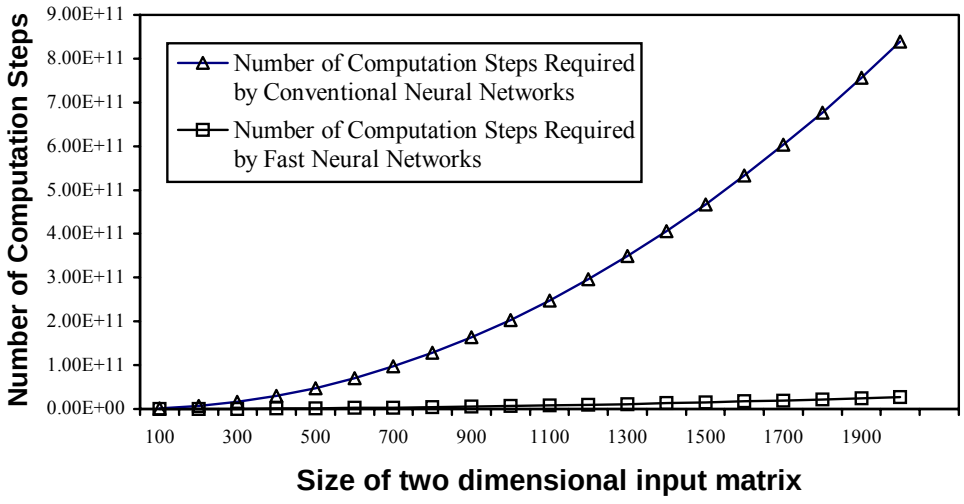

Fig. 25 A comparison between the numbers of computation steps required by fast and conventional neural networks in case of complex-valued two dimensional input matrix and complex-valued weight matrix (n=30)

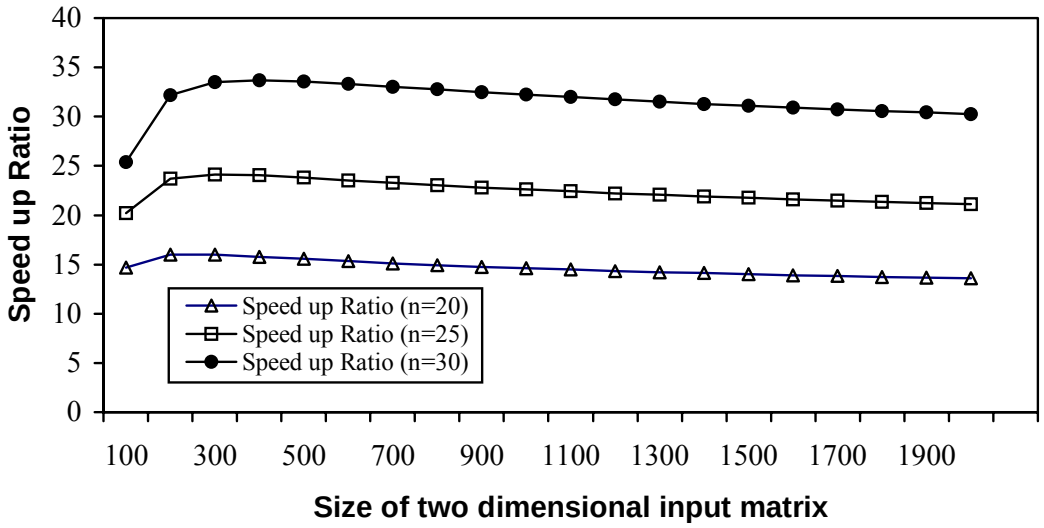

Fig. 26 The relation between the speed up ratio and the size of two dimensional real-valued input matrix in case of complex-valued weights

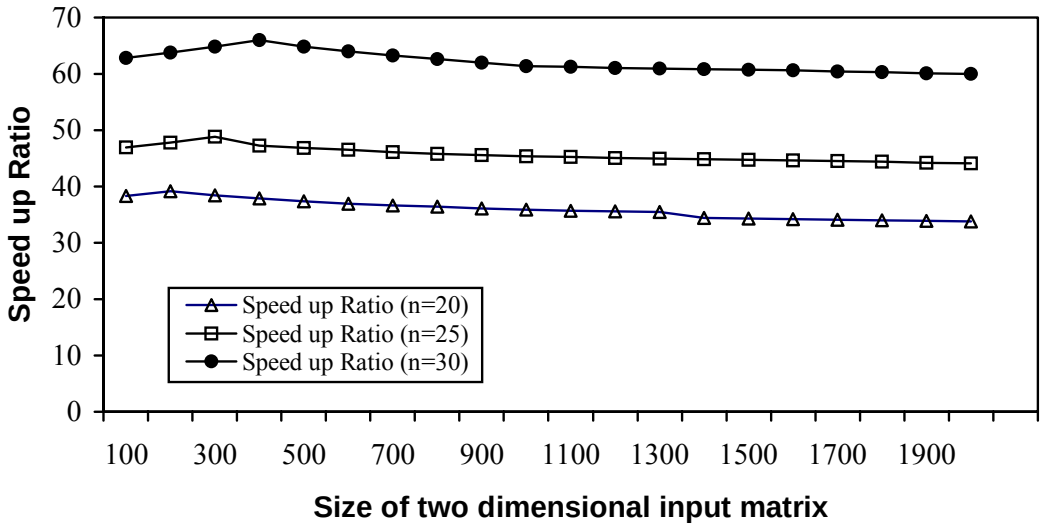

Fig. 27 Practical speed up ratio in case of two dimensional complex-valued input matrix in and complex-valued weights