World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:5, 2008

Fast Complex Valued Time Delay Neural Networks

Hazem M. El-Bakry and Qiangfu Zhao

Abstract**—**Here, a new idea to speed up the operation of complex valued time delay neural networks is presented. The whole data are collected together in a long vector and then tested as a one input pattern. The proposed fast complex valued time delay neural networks uses cross correlation in the frequency domain between the tested data and the input weights of neural networks. It is proved mathematically that the number of computation steps required for the presented fast complex valued time delay neural networks is less than that needed by classical time delay neural networks. Simulation results using MATLAB confirm the theoretical computations.

Keywords**—**Fast Complex Valued Time Delay Neural Networks, Cross Correlation, Frequency Domain

I. Introduction

In recent years, complex-valued neural networks have widened the scope of application in optoelectronics, imaging, remote sensing, quantum neural devices and systems, spatiotemporal analysis of physiological neural systems, and artificial neural information processing [6,10]. The main objective of this paper is to reduce the response time of complex valued time delay neural networks. The idea is to perform the testing process in the frequency domain instead of time domain. This is our idea which was applied successfully for sub-image detection using fast neural networks proposed in [1,2,3]. Furthermore, such idea was presented for fast face detection [5,7,9]. Moreover, the same principles were applied for fast iris detection [8]. Another idea to further increase the speed of fast neural networks through image decomposition was suggested in [7].

Manuscript received January 1, 2005.
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, Japan 965-8580 (phone +81-242-37-2519, fax. +81-242-37-2743, e-mail: helbakry20@yahoo.com).
Q. Zhao is professor with the Information Systems Department, University of Aizu, Japan (e-mail: qf-zhao@u-aizu.ac.jp).

Fast neural networks for detecting a certain code in one dimensional serial stream of sequential data was described in [4]. Compared with conventional neural networks, fast neural networks based on cross correlation between the tested data and the input weights of neural networks in the frequency domain showed a significant reduction in the number of computation steps required for certain data detection [1,2,3,4,5,7,8,9]. Here, we make use of our theory on fast neural networks implemented in the frequency domain to increase the speed of complex valued time delay neural networks. The idea of moving the testing process from the time domain to the frequency domain is applied to complex valued time delay neural networks. Theoretical and practical results show that the proposed fast complex valued time delay neural networks are faster than classical complex valued time delay neural networks. In section II, our theory on fast neural networks for detecting certain data in one dimensional matrix is described. Experimental results for fast complex valued time delay neural networks are presented in section III.

II. Theory of Fast Neural Networks Based on Cross Correlation in the Frequency Domain

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 may 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]:

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

III. Experimental Results for Complex Valued Time Delay Neural Networks

Complex valued neural networks have many applications in fields dealing with complex numbers such as telecommunications, speech recognition and image processing with the Fourier transform [6]. Complex valued neural networks means that the inputs, weights, thresholds and the activation function have complex values. Complex valued time delay 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 input data is tested by complex valued time delay neural networks as a single pattern with length L (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 for fast neural networks. Therefore, the speed up ratio in case of complex valued time delay neural networks can be evaluated as follows:

1) In case of real inputs

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:

$$\theta=(2q(2n-1)(N-n+1)) \quad (12)$$

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

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

The theoretical speed up ratio for searching of a short successive (n) data in a long input vector (L) using complex valued time delay neural networks is listed in Tables 1, 2, and 3. Also, practical speed up ratio for manipulating matrices of different sizes (L) and different in size weight matrices (n) is listed in Table 4 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 ($2n^2$) 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:

$$\theta=(2q(2n^2-1)(N^2-n^2+1)) \quad (14)$$

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

$$\eta=\frac{2q(2n^2-1)(N^2-n^2+1)}{(2q+1)(5N^2\log_2 N^2)+q(8N^2-n^2)+N} \quad (15)$$

The theoretical speed up ratio for detecting (nxn) real valued submatrix in a large real valued matrix (NxN) using complex

valued time delay neural networks is listed in Tables 5, 6, 7. Also, practical speed up ratio for manipulating matrices of different sizes (NxN) and different in size weight matrices (n) is listed in Table 8 using 700 MHz processor and MATLAB.

2) In case of complex inputs

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:

$$\theta=(2q(4n-1)(N-n+1)) \tag{16}$$

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

$$\eta=\frac{2q(4n-1)(N-n+1)}{(2q+1)(5N\log_2 N)+q(8N-n)+N} \tag{17}$$

The theoretical speed up ratio for searching of a short complex successive (n) data in a long complex valued input vector (L) using complex valued time delay neural networks is listed in Tables 9, 10, and 11. Also, practical speed up ratio for manipulating matrices of different sizes (L) and different in size weight matrices (n) is listed in Table 12 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:

$$\theta=(2q(4n^2-1)(N^2-n^2+1)) \tag{18}$$

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

$$\eta=\frac{2q(4n^2-1)(N^2-n^2+1)}{(2q+1)(5N^2\log_2 N^2)+q(8N^2-n^2)+N} \tag{19}$$

The theoretical speed up ratio for detecting (nxn) complex valued submatrix in a large complex valued matrix (NxN) using complex valued neural networks is listed in Tables 13, 14, and 15. Also, practical speed up ratio for manipulating matrices of different sizes (NxN) and different in size weight matrices (n) is listed in Table 16 using 700 MHz processor and MATLAB.

For one dimensional matrix, from Tables 1, 2,3,4,9,10,11, and 12 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 as listed in Tables 5,6,7,8,13,14,15, and 16. From these tables, it is clear that the maximum speed up ratio is achieved at image size (N=200) when n=20, then at image size (N=300) when n=25, and at image size (N=400) when n=30. This confirms our previous results presented in [7] on fast subimage 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 subimages and processing each subimage in the frequency domain separately using a single fast neural processor.

IV. Conclusion

A new fast complex valued time delay neural networks has been presented. Theoretical computations have shown that fast complex valued time delay neural networks requires fewer computation steps than conventional one. This has been achieved by applying cross correlation in the frequency domain between the input data and the input weights of complex valued time delay neural networks. Simulation results have confirmed this proof by using MATLAB. This algorithm can be successfully applied for any application uses complex valued time delay neural networks.

References

[1] 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 Congress WEC'05, Istanbul, Turkey, 25-27 Feb., 2005.

[2] 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 Congress WEC'05, Istanbul, Turkey, 25-27 Feb., 2005.

[3] H. M. El-Bakry, and Q. Zhao, "Fast Pattern Detection Using Normalized Neural Networks and Cross Correlation in the Frequency Domain,"

accepted and under publication in the EURASIP Journal on Applied Signal Processing.

[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, "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.

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

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

[8] 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.

[9] 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.

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

[11] H. M. El-Bakry, and Q. Zhao, " New Fast Time Delay Neural Networks," Accepted for publication in the International Conference on Information and Knowledge Engineering (IKE'05), June 20-23, 2005, Las Vegas, USA.

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

His research interests include neural networks, pattern recognition, image processing, biometrics, cooperative intelligent systems and electronic circuits. In these areas, he has published more than 39 papers as a single author in major international journals and conferences. He is the first author in 9 refereed international journal papers and more than 60 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.

TABLE 1
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (1D-REAL VALUES INPUT MATRIX, n=400)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
10000	4.6027e+008	4.2926e+007	10.7226
40000	1.8985e+009	1.9614e+008	9.6793
90000	4.2955e+009	4.7344e+008	9.0729
160000	7.6513e+009	8.8219e+008	8.6731
250000	1.1966e+010	1.4275e+009	8.3823
360000	1.7239e+010	2.1134e+009	8.1571
490000	2.3471e+010	2.9430e+009	7.9752
640000	3.0662e+010	3.9192e+009	7.8237
810000	3.8812e+010	5.0442e+009	7.6945
1000000	4.7921e+010	6.3201e+009	7.5823

TABLE 2
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (1D-REAL VALUES INPUT MATRIX, n=625)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
10000	7.0263e+008	4.2919e+007	16.3713
40000	2.9508e+009	1.9613e+008	15.0452
90000	6.6978e+009	4.7343e+008	14.1474
160000	1.1944e+010	8.8218e+008	13.5388
250000	1.8688e+010	1.4275e+009	13.0915
360000	2.6932e+010	2.1134e+009	12.7433
490000	3.6674e+010	2.9430e+009	12.4612
640000	4.7915e+010	3.9192e+009	12.2257
810000	6.0655e+010	5.0442e+009	12.0247
1000000	7.4893e+010	6.3201e+009	11.8500

TABLE 3
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (1D-REAL VALUES INPUT MATRIX, n=900)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
10000	9.823 e+008	4.2911e+007	22.8933
40000	4.2206e+009	1.9612e+008	21.5200
90000	9.6176e+009	4.7343e+008	20.3149
160000	1.7173e+010	8.8217e+008	19.4671
250000	2.6888e+010	1.4275e+009	18.8356
360000	3.8761e+010	2.1134e+009	18.3409
490000	5.2794e+010	2.9430e+009	17.9385
640000	6.8985e+010	3.9192e+009	17.6018
810000	8.7334e+010	5.0442e+009	17.3139
1000000	1.0784e+011	6.3201e+009	17.0635

TABLE 4
PRACTICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (1D-REAL VALUES INPUT MATRIX)

Length of input matrix	Speed up ratio (n=400)	Speed up ratio (n=625)	Speed up ratio (n=900)
10000	17.88	25.94	35.21
40000	17.19	25.11	34.43
90000	16.65	24.56	33.59
160000	16.14	24.14	33.05
250000	15.89	23.76	32.60
360000	15.58	23.23	32.27
490000	15.28	22.87	31.99
640000	14.08	22.54	31.78
810000	13.87	22.32	31.60
1000000	13.69	22.11	31.47

TABLE 5
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (2D-REAL VALUES INPUT MATRIX, n=20)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	3.1453e+008	4.2916e+007	7.3291
200x200	1.5706e+009	1.9610e+008	8.0091
300x300	3.7854e+009	4.7335e+008	7.9970
400x400	6.9590e+009	8.8203e+008	7.8898
500x500	1.1091e+010	1.4273e+009	7.7711
600x600	1.6183e+010	2.1130e+009	7.6585
700x700	2.2233e+010	2.9426e+009	7.5556
800x800	2.9242e+010	3.9186e+009	7.4623
900x900	3.7209e+010	5.0434e+009	7.3778
1000x1000	4.6136e+010	6.3191e+009	7.3010

TABLE 6
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (2D-REAL VALUES INPUT MATRIX, n=25)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	4.3285e+008	4.2909e+007	10.0877
200x200	2.3213e+009	1.9609e+008	11.8380
300x300	5.7086e+009	4.7334e+008	12.0602
400x400	1.0595e+010	8.8202e+008	12.0119
500x500	1.6980e+010	1.4273e+009	11.8966
600x600	2.4863e+010	2.1130e+009	11.7667
700x700	3.4246e+010	2.9425e+009	11.6381
800x800	4.5127e+010	3.9185e+009	11.5163
900x900	5.7507e+010	5.0434e+009	11.4025
1000x1000	7.1386e+010	6.3191e+009	11.2969

TABLE 7
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (2D-REAL VALUES INPUT MATRIX, n=30)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	5.4413e+008	4.2901e+007	12.6834
200x200	3.1563e+009	1.9608e+008	16.0966
300x300	7.9272e+009	4.7334e+008	16.7476
400x400	1.4857e+010	8.8201e+008	16.8444
500x500	2.3946e+010	1.4273e+009	16.7773
600x600	3.5193e+010	2.1130e+009	16.6552
700x700	4.8599e+010	2.9425e+009	16.5160
800x800	6.4164e+010	3.9185e+009	16.3745
900x900	8.1888e+010	5.0434e+009	16.2367
1000x1000	1.0177e+011	6.3191e+009	16.1052

TABLE 8
PRACTICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (2D-REAL VALUES INPUT MATRIX)

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	17.19	22.32	31.74
200x200	17.61	22.89	32.55
300x300	16.54	23.66	33.71
400x400	15.98	22.95	34.53
500x500	15.62	22.49	33.32
600x600	15.16	22.07	32.58
700x700	14.87	21.83	32.16
800x800	14.64	21.61	31.77
900x900	14.43	21.42	31.45
1000x1000	14.26	21.22	31.12

TABLE 9
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (1D-COMPLEX VALUES INPUT MATRIX, N=400)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	9.2111e+008	4.2926e+007	21.4586
200x200	3.7993e+009	1.9614e+008	19.3706
300x300	8.5963e+009	4.7344e+008	18.1571
400x400	1.5312e+010	8.8219e+008	17.3570
500x500	2.3947e+010	1.4275e+009	16.7750
600x600	3.4500e+010	2.1134e+009	16.3245
700x700	4.6972e+010	2.9430e+009	15.9604
800x800	3.9192e+009	6.1363e+010	15.6571
900x900	7.7673e+010	5.0442e+009	15.3985
1000x1000	9.5902e+010	6.3201e+009	15.1740

TABLE 10
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (1D-COMPLEX VALUES INPUT MATRIX, N=625)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	1.4058e+009	4.2919e+007	32.7558
200x200	5.9040e+009	1.9613e+008	30.1025
300x300	1.3401e+010	4.7343e+008	28.3061
400x400	2.3897e+010	8.8218e+008	27.0883
500x500	3.7391e+010	1.4275e+009	26.1934
600x600	5.3885e+010	2.1134e+009	25.4969
700x700	7.3377e+010	2.9430e+009	24.9324
800x800	9.5868e+010	3.9192e+009	24.4612
900x900	1.2136e+011	5.0442e+009	24.0590
1000x1000	1.4985e+011	6.3201e+009	23.7095

TABLE 11
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (1D-COMPLEX VALUES INPUT MATRIX, N=900)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	1.9653e+009	4.2911e+007	45.7993
200x200	8.4435e+009	1.9612e+008	43.0519
300x300	1.9240e+010	4.7343e+008	40.6410
400x400	3.4356e+010	8.8217e+008	38.9450
500x500	5.3791e+010	1.4275e+009	37.6817
600x600	7.7544e+010	2.1134e+009	36.6920
700x700	1.0562e+011	2.9430e+009	35.8870
800x800	1.3801e+011	3.9192e+009	35.2134
900x900	1.7472e+011	5.0442e+009	34.6375
1000x1000	2.1575e+011	6.3201e+009	34.1365

TABLE 12
PRACTICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (1D-COMPLEX VALUES INPUT MATRIX)

Length of input matrix	Speed up ratio (n=400)	Speed up ratio (n=625)	Speed up ratio (n=900)
10000	37.90	53.58	70.71
40000	36.82	52.89	69.43
90000	36.34	52.47	68.69
160000	35.94	51.88	68.05
250000	35.69	51.36	67.56
360000	35.28	51.02	67.15
490000	34.97	50.78	66.86
640000	34.67	50.56	66.58
810000	34.45	50.38	66.39
1000000	34.28	50.11	66.19

TABLE 13
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (2D-COMPLEX VALUES INPUT MATRIX, N=20)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	6.2946e+008	4.2916e+007	14.6674
200x200	3.1431e+009	1.9610e+008	16.0281
300x300	7.5755e+009	4.7335e+008	16.0040
400x400	1.3927e+010	8.8203e+008	15.7894
500x500	2.2197e+010	1.4273e+009	15.5519
600x600	3.2386e+010	2.1130e+009	15.3266
700x700	4.4493e+010	2.9426e+009	15.1206
800x800	5.8520e+010	3.9186e+009	14.9340
900x900	7.4465e+010	5.0434e+009	14.7649
1000x1000	9.2329e+010	6.3191e+009	14.6110

TABLE 14
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (2D-COMPLEX VALUES INPUT MATRIX, N=25)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	8.6605e+008	4.2909e+007	20.1836
200x200	4.6445e+009	1.9609e+008	23.6856
300x300	1.1422e+010	4.7334e+008	24.1301
400x400	2.1198e+010	8.8202e+008	24.0333
500x500	3.3973e+010	1.4273e+009	23.8028
600x600	4.9746e+010	2.1130e+009	23.5427
700x700	6.8519e+010	2.9425e+009	23.2856
800x800	9.0290e+010	3.9185e+009	23.0418
900x900	1.1506e+011	5.0434e+009	22.8142
1000x1000	1.4283e+011	6.3191e+009	22.6027

TABLE 15
THE THEORETICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (2D-COMPLEX VALUES INPUT MATRIX, N=30)

Length of input matrix	Number of computation steps required for classic complex valued neural networks	Number of computation steps required for fast complex valued neural networks	Speed up ratio
100x100	1.0886e+009	4.2901e+007	25.3738
200x200	6.3143e+009	1.9608e+008	32.2021
300x300	1.5859e+010	4.7334e+008	33.5045
400x400	2.9722e+010	8.8201e+008	33.6981
500x500	4.7904e+010	1.4273e+009	33.5640
600x600	7.0405e+010	2.1130e+009	33.3197
700x700	9.7225e+010	2.9425e+009	33.0412
800x800	1.2836e+011	3.9185e+009	32.7581
900x900	1.6382e+011	5.0434e+009	32.4825
1000x1000	2.0360e+011	6.3191e+009	32.2193

TABLE 16
PRACTICAL SPEED UP RATIO FOR COMPLEX VALUED TIME DELAY NEURAL NETWORKS (2D-COMPLEX VALUES INPUT MATRIX)

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	38.33	46.99	62.88
200x200	39.17	47.79	63.77
300x300	38.44	48.86	64.83
400x400	37.92	47.23	65.99
500x500	37.32	46.89	64.89
600x600	36.96	46.48	64.01
700x700	36.67	46.08	63.31
800x800	36.38	45.78	62.64
900x900	36.11	45.54	61.95
1000x1000	35.86	45.36	61.40