A Comparative Study on the Development of Binary Object Extraction System using Different Self Organizing Neural Network

Harshshikha Nandan
Department of CSE
Sikkim Manipal Institute of Technology
Majitar, East Sikkim

Manisha Jindal
Department of CSE
Sikkim Manipal Institute of Technology
Majitar, East Sikkim

Arsh
Department of CSE
Sikkim Manipal Institute of Technology
Majitar, East Sikkim

Debanjan Konar
Department of CSE
Sikkim Manipal Institute of Technology
Majitar, East Sikkim

ABSTRACT
Accurately Extraction of a binary object from a noisy perspective has been a daunting task in the field of pattern recognition. Several techniques have been tried to optimally solve the problem of denoising of object over the decades. In this paper, different binary object extraction methods are reviewed which are basically guided by different Self-Organizing Neural Networks (SONN) architectures as Bi-Directional Self Organizing Neural Network (BDSONN), multi-Layer Self Organizing neural Network (MLSONN) and quantum version of MLSONN (QMLSONN). The result shows that QMLSONN outperforms over other network architectures in terms of time and also it restores shape of the object with great accuracy.

Keywords
Binary object extraction, Fuzzy context sensitive thresholding, Quantum computing, Multilayer self-organizing neural network, Quantum back-propagation algorithm, Bidirectional self-organizing neural network, System transfer index, Quantum multilayer self-organizing neural network.

1. INTRODUCTION
The main objective of extraction [1,2,3,4] of binary object from a noisy perspective is to split the given image application into foreground and background regions. This region contains the noisy region and the object region. A wide variety of neural networks in their architectures have been used over the decades to deal with preprocessing of image tasks [2,3,4,5,6,7,8,9,10]. For the purpose of identification and recognition of binary objects from an image scene [11,12,13], neural networks have routinely been employed by researchers. Neural networks have also been used for classification [6,11,14,15,16] of relevant objects from redundant image information bases and segmenting[8] the image data. Various neural network [17,18] architectures, consisting of both self-supervised and unsupervised, have also been proposed to produce outputs in real time. The multilayer self-organizing neural network (MLSONN)[19] is a feed-forward network architecture and it resorts to the fuzzy measures of the image information for the purpose of derivation of system errors therein. The Bi-Directional Self-Organizing Neural Network (BDSONN)[19,20] architecture is guided by counter-propagation of the intermediate and the output network states for self-organizing [21] the input information. The results show the restoring capabilities of the network in the extracted images. The main problems with these architectures lie in the adjustment and reassignment of the interconnection weight [22] matrices required for the processing of the information. Most of these approaches use back-propagation algorithm for updating the interconnection weights.

This article is a survey on these approaches- BDSONN [20], BDSONN using β-activation [19], MLSONN [23][14] and QMLSONN[23], which have been used to achieve the objective of binary object extraction.

2. Bidirectional Self-Organizing Neural Network Architecture (BDSONN):
An input layer, an intermediate layer and an output layer are the fundamental components of this BDSONN. This architecture exploits a feed-forward mode of propagation of input information and counter-propagation of network states which self-organize the input information into extracted object regions. The transmission of necessary information is guided by the standard sigmoid activation function along with image context sensitive thresholding values that is derived from the fuzzy[24] image information content through the fuzzy cardinality estimates of image pixel neighborhoods. BDSONN obviates time consuming back propagation algorithm in this architecture which improves the performance of the extraction procedure.

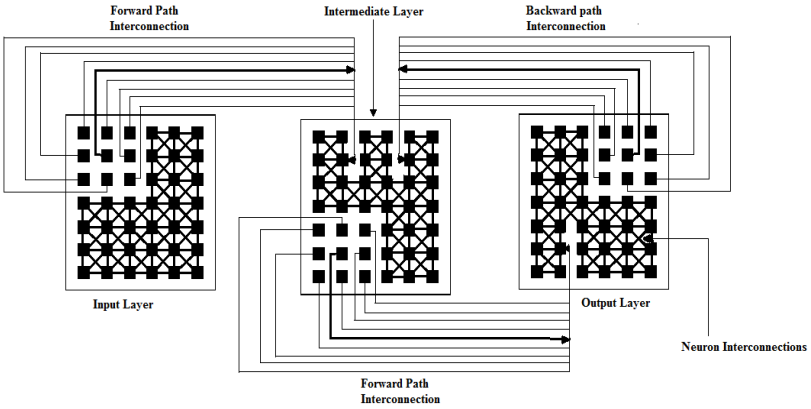


Fig 1: Bidirectional self-organizing neural network (BDSONN) architecture (Not all intra-layer interconnections are shown for clarity)[20]

The BDSONN architecture comprises an input layer for accepting the external inputs and two self-organizing layers, viz. the intermediate layer and the output layer. Every neuron in each network layer corresponds to the number of pixels in the input image scene. The input layer is given the task of accepting the fuzzy membership values of the pixels in the input image. The neurons of every layer are connected to each other within the same layer forming a cellular network structure. All the intra-layer interconnections strengths are set to a fixed value of unity. Each neuron in a layer is also connected to the corresponding neuron in the neighboring layer through forward path inter layer interconnections. Similarly, the output layer neurons are connected to the corresponding neuron in the previous layer via backward path inter-layer interconnections. The measures of the membership values at the individual neurons of the different layers decide the strengths of the inter-layer interconnections between the input layer and the intermediate layer, the intermediate layer and the output layer and between the output layer and the intermediate layer neurons.

The central candidate neurons accumulate the fuzzy cardinality values corresponding to the different neighborhood fuzzy subsets in the input layer through the intra layer interconnections. These fuzzy cardinality values are used to define the context sensitive thresholding information for the characteristic transfer function of the processing neurons.

The BDSONN architecture self-organizes input information into extracted output information. This self-organization procedure is carried in the following four phases: First is the initialization phase, where the intra-layer interconnections are initialized to one between the constituent neurons within the different network layers. Second is the input phase, where input layer of the network is fed in with the fuzzy membership values of the external world input noisy image scene pixels. Next is the forward propagation phase, where the fuzzy membership values are propagated from the input layer to the intermediate layer and the intermediate layer to the output layer. This forward path inter-layer interconnection is determined from the relative fuzzy membership values at the constituent neurons of the preceding network layer and the processed outputs of the preceding network layer. Last comes the backward propagation phase. Here the output layer outputs of the network are propagated to the previous network intermediate layer after the backward path inter-layer interconnections. Each propagation phase includes the process of determining the fuzzy cardinality estimates of the varying network layer neighborhood fuzzy subsets. This is used for computing the fuzzy context sensitive thresholding information required to process operation of the succeeding network layer. The following illustrates the working of the BDSONN architecture: [20]

Initialization phase:

1. Initialize I[l], l=1,2,3 ;

/* I[l] intra-layer connections*/

2. Read P[l][a][b];

/* P[l][a][b] is the membership value of neuron at position ath row and bth column of lth layer*/

Forward Propagation phase:

3. T[l+1][i][j]= C[l][i][j];

/* T is the context sensitive thresholding accumulated at ith row and jth column of (l+1)th layer and C is the cardinality of the neuron at ith row and jth column of lth layer*/

4. IC[t][l][l+1][a][b]=1-(P[l][a][b]-P[l][i][j]);

/* IC is the interconnection that needs to be propagated to corresponding neuron of the next layer */

5.P[l+1][i][j]= SUM[fsig(P[1][a][b]*IC[t][l][l+1][a][b])];

/* Calculation of the necessary membership value to be sent to the corresponding neuron in the next layer */

6. do

7. Repeat steps 4,5,6 with intermediate layers outputs

Backward Propagation phase:

8. T[l][i][j]= C[l-1][i][j];

9. IC[t][l-1][l][a][b]=1-(P[l-1][a][b]-P[l-1][i][j]);

10. P[l][i][j] = SUM[fsig(P[l-1][a][b]*Inter[t][l-1][l][a][b])];

11. Loop until ((IC[t][l][l+1][a][b]-IC[t-1][l][l+1][a][b]) <Ɛ);

/* Ɛ is the tolerable error*/

12. End

2.2 Bidirectional Self-Organizing Neural Network Architecture using β Activation Function:

The complete network and the functions of this architecture work similar to the BDSONN architecture discussed above. The difference lies only in the part where the activation function used is β-activation [19,25] rather than the sigmoid activation function.

2.3 Multilayer Self-Organizing Neural Network (MLSONN) Architecture:

MLSONN is a self-supervised architecture efficient in extracting binary objects from a noisy background. It involves several layers, viz. the input layer, any number of hidden layers and an output layer for obtaining the final processed

output. A second order neighborhood topology is followed by the interconnection strategy employed between the layers. The network is characterized by subsequent processing at the hidden layers and the output layer and forward propagation of incident information. Using the linear indices of fuzziness of the intermediate output obtained the network system errors are calculated at the output layer which are used to adjust the network interconnection weights through the standard back propagation algorithm. These intermediate outputs are fed back to the input layer for further processing and this process is continued until the network errors stabilize.

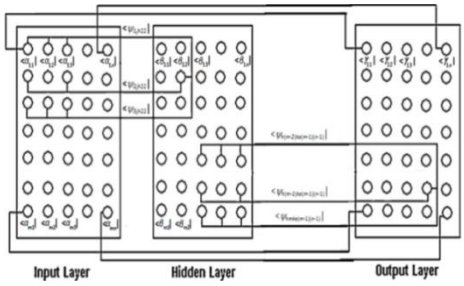


Fig 2: Schematic diagram of QMLSONN (Only few connections are shown for clarity)[23]

2.4 Quantum Multilayer Self-Organizing neural Network (QMLSONN) Architecture:

2.4.1 Concept of Qubits:

The basic unit of information processing that can be used in quantum computation is called qubits [23], which is also known as quantum bits.

A qubit can exist in the state |0> or |1>, but it can also exist in a superposition state which is a liner combination of the states |0> and |1>. It can be represented as |ψ> which is written as

$$\psi >= \alpha|0> + \beta|1> \qquad (1)$$

2.4.2 Concept of Rotation Gates:

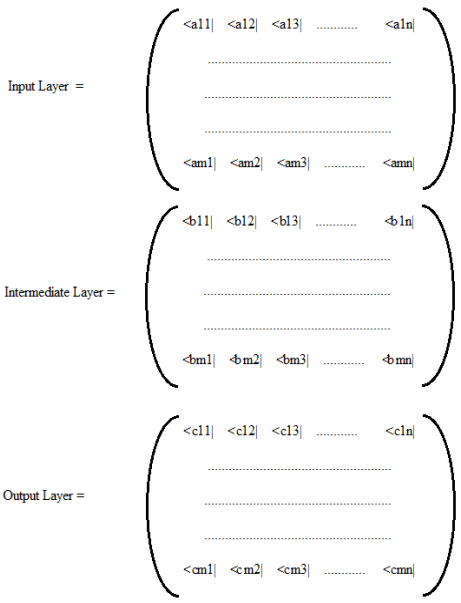
The single qubit rotation gate [2] can be defined: $\text{Rot}(\theta) = \begin{bmatrix} cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ (2)

This is executed on a sine qubit to change it to:

$$\text{Rot}(\theta) = \begin{bmatrix} cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} * \begin{bmatrix} cos\phi \\ sin\phi \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta + \phi) \\ \sin(\theta + \phi) \end{bmatrix} \qquad (3)$$

The quantum version of the MLSONN [23][19] architecture is QMLSONN[2] in which the processing nodes of the different layers are simply qubits like



The interconnection weights follow a second order neighborhood topology and are in the form of rotation gates.

2.4.3 Dynamics of operation

When input data (input) is given into the network, the fuzzified input value [0,1] is

First converted by the input layer into the phase [0, π/2] in quantum states.

$$y_i^I = \pi/2(\text{input}) \qquad (4)$$

Here, y_i^I are quantum inputs and (input) are general input data's. Also we have input neurons

i = 1 to l and hidden neurons j= 1 to m.

$$U_j^h = \sum_{i=1}^{l} f(\psi ipjhll) f(yiI) - f(\lambda jh) \qquad (5)$$

Where, ψ_{ipjhll} are the connection weights between input and hidden layer and λjh is the threshold of j^{th} hidden neuron.

$$U_j^h = \sum f(\psi_{ipjhll}) f(y_i^I) - f(\lambda_j^h) \qquad (6)$$

$$= \sum e^{i\psi_{ipjhll}} + y_i^I) - f(\lambda_j^h) \qquad (7)$$

$$= \cos(\psi_{ipjhll} + y_i^I) + i\sin(\psi_{ipjhll} + y_i^I) - \cos(\lambda_j^h)$$

$$-i\sin(\lambda_j^h) \qquad (8)$$

Now,

$$Y_j^h = \pi/2g(\delta_j^h) - \arg(u_j^h) \qquad (9)$$

$$= \pi/2g(\delta_j^h) - \tan^{-1}\frac{\sum \sin(\psi ipjhll + yiI) - \sin(\lambda jh)}{\sum \cos(\psi ipjhll + yiI) - \cos(\lambda jh)} \qquad (10)$$

Here δ_j^h is the reversal parameter of j^{th} hidden neuron and $\arg(u_j^h)$ means the phase extracted from a complex number u.

Similarly,

$$U_k = \sum f(\psi_{hqkoll}) f(y_j^h) - f(\lambda_k) \qquad (11)$$

Where, λ_k is the threshold of k^{th} output neuron and ψ_{hqkoll} are the connection weights between hidden and output layer.

$$U_k = \sum e^i \psi_{hqkoll} e^{iy_j^h} - e^{i\lambda_k} \qquad (12)$$

$$= \sum e^i (\psi_{hqkoll} + y_j^h) - e^{i\lambda_k}$$

In this neuron model two kinds of parameters exist: the phase parameter of weight connection and θ threshold λ and the reversal parameter δ.

The network error is represented as

$$E_{total} = \tfrac{1}{2} \sum \sum (t_{n,p} - output_{n,p})^2 \quad (13)$$

$$= \cos(\psi_{hqkoll} + y_j^h) + i\sin(\psi_{hqkoll} + y_j^h) - \cos(\lambda_k) - i\sin(\lambda_k) \quad (14)$$

Now,

$$Y_k = \pi/2\, g(\delta_k) - \arg(u_k) \quad (15)$$

Where $g(\delta_k)$ is the sigmodial activation function.

$$= \pi/2 * g(\delta_k) - \tan^{-1} \frac{\sum \sin(\psi hqkoll + yjh) - \sin(\lambda k)}{\sum cos(\psi hqkoll + yjh) - \cos(\lambda k)} \quad (16)$$

The quantum states of the outputs are destroyed by a quantum measurement at the output layer and convert the same to either 0 or 1 depending on a probability. These outputs are retained in a safe custody since these outputs are to be further processed. Using linear indices of fuzziness, the probability amplitudes of the quantum states are compared with 1's and 0's to compute the system errors which are used to regulate the interconnection weights of the network layers. For further processing, the retained outputs are fed back to the input layer. This process is repeated until the network system errors fall below a certain reasonable limit when the object gets extracted from the noisy background.

2.4.4 Quantum Back propagation Algorithm

The quantum back propagation error adjustment algorithm [23] is discussed and generalized in this section for any number of layers. The probability amplitude |1> is attached to the imaginary part and\0> to the real part for this purpose.

Here, P is the number of learning patterns. $t_{n,p}$ is a target signal for the n^{th} neuron and $output_{n,p}$ means an output at the P^{th} pattern. The error gradient is given by

$$\frac{\partial E}{\partial \theta jk} = t_{n,p} - output_{n,p} \frac{\partial output}{\partial \theta jk} \quad (17)$$

After successive calculations the equation retrieved is: $\frac{\partial y\, hj}{\partial \delta\, hj}$

$$= \pi/2\, e^{-\delta j^h}/(1+\delta_j^h) \quad (18)$$

2.5 Results and Discussions:

In this paper, all these four algorithms are implemented on both real world image and synthetic image application with various uniform and Gaussian noise levels (σ=8,10,12,14) and also compared them. The result is shown in Fig 4 and Fig.5 below. The percentage of correct classification of pixels (pcc) [22] is calculated on the basis of the achieved image and the original image to determine the quality of the extracted image. It uses the formula: $pcc = \frac{tcc}{tnp} * 100$, where t_{cc} stands for total number of correctly classified pixels and the total number of pixels in the image scene is designated to t_{np}

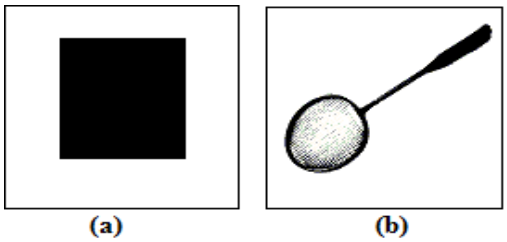


Fig3: Original images (a) synthetic image (b) spanner image

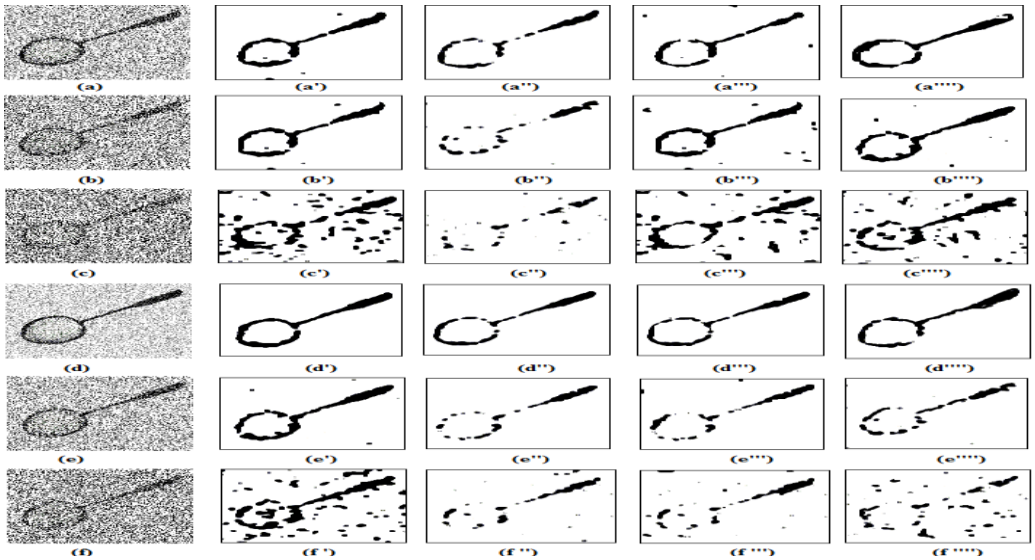


Fig 4:Noisy and extracted real world images a, b, c noisy images with Gaussian noise at σ = 8, 10,12 and d, e, f noisy images with Uniform noise = 8,10,12; a', b', c', d', e', f' BDSONN extracted images; a'', b'', c'', d'', e'', f'' BDSONN with β-activation function extracted images; a''', b''', c''' ,d''' , e''', f''' QMLSONN extracted images; and a'''', b'''', c'''', d'''', e'''', f'''' MLSONN extracted images.

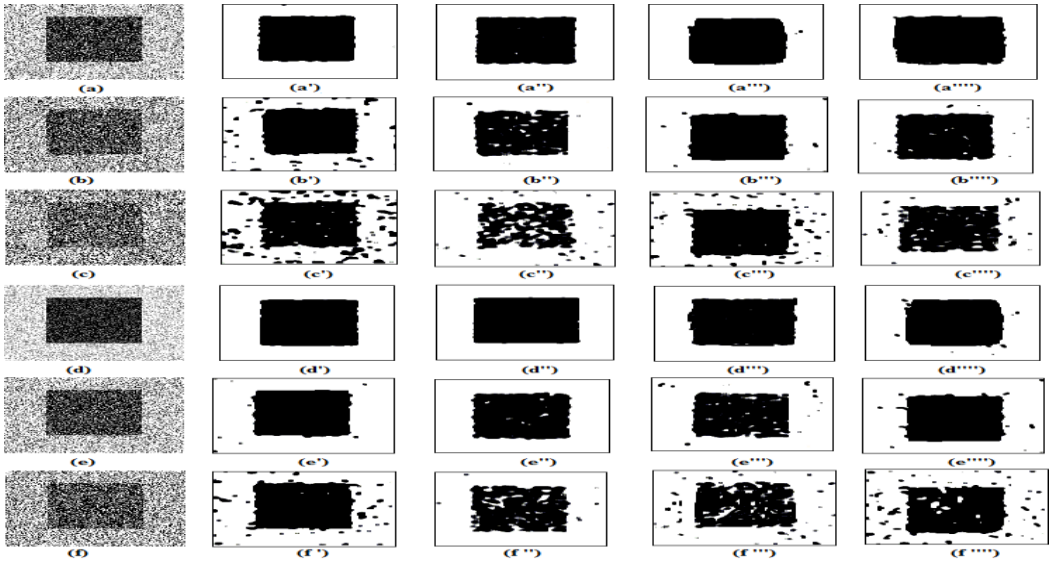


Fig 5:Noisy and extracted synthetic images a, b, c noisy images with Gaussian noise at σ = 8, 10,12 and d, e, f noisy images with Uniform noise = 8,10,12; a', b', c', d', e', f' BDSONN extracted images; a'', b'', c'', d'', e'', f'' BDSONN with β-activation function extracted images; a''', b''', c''', d''', e''', f''' QMLSONN extracted images; and a'''', b'''', c'''', d'''', e'''', f'''' MLSONN extracted images.

Table 1: Comprative performance results on Gaussian noisy images:

Synthetic image	**MLSONN**		**BDSONN**		**β - BDSONN**		**QMLSONN**	
σ	*t*(secs)	*pcc*	*t*(secs)	*pcc*	*t*(secs)	*pcc*	*t*(secs)	*pcc*
8	79	99.59	32	99.94	72	99.94	12	99.96
10	73	98.78	43	99.65	142	99.07	15	99.75
12	115	96.28	62	98.96	211	98.41	21	98.97
14	156	95.79	104	98.27	235	97.05	39	98.17
Real life image								
σ	*t*(secs)	*pcc*	*t*(secs)	*pcc*	*t*(secs)	*pcc*	*t*(secs)	*pcc*
8	74	99.82	31	99.86	72	99.94	3	99.91
10	72	99.52	32	99.75	179	97.05	2	99.59
12	73	98.23	42	99.77	206	97.61	3	99.14
14	71	97.29	41	99.25	212	96.78	4	98.38

Table 2: Comprative performance results on uniform noisy images:

Synthetic image	**MLSONN**		**BDSONN**		**β - BDSONN**		**QMLSONN**	
σ	*t*(secs)	*pcc*	*t*(secs)	*pcc*	*t*(secs)	*pcc*	*t*(secs)	*pcc*
8	74	99.89	32	100	71	99.90	3	99.96
10	73	99.62	31	99.99	140	99.52	4	99.69
12	72	99.39	42	99.97	200	99.41	4	99.11
14	72	97.29	41	99.25	215	97.68	5	98.45
Real life image								
σ	*t*(secs)	*pcc*	*t*(secs)	*pcc*	*t*(secs)	*pcc*	*t*(secs)	*pcc*
8	71	98.83	31	99.75	70	99.51	1	92.67
10	74	98.36	30	99.72	160	98.39	2	92.34
12	73	96.98	40	99.39	209	97.05	3	91.98
14	115	95.15	50	98.11	222	96.75	4	90.65

Table 1 and 2 clearly show that QMLSONN is superior to MLSONN and BDSONN in terms of time and restores the quality of image with great precision.

3. CONCLUSION

Binary object extraction from noisy perspective is a daunting task that has applicability in wide variety of fields. The widely used SONN for denoising of binary objects have been discussed in this article. This paper provides a broad study of the various approaches to fulfill this objective and their working has been elaborately presented relying on the work done by Bhattacharyya et al.[19,20,23,25]. This paper also shows the performance comparison between the various algorithms discussed. It is noted that incorporating quantum computing MLSONN outperforms than the rest. The future work mainly consists of developing algorithms that completely extracts an image from a noisy perspective within lesser span of time. The authors are working in this direction.

4. REFERENCES

[1] Forrest, B.M. et al 1988 Neural network models. Parallel Computation 8:71–83

[2] Lippmann, R.P. 1987 An introduction to computing with neural nets. IEEE ASSP Magazine, 3–22.

[3] Hay, K.S. 1994 neural networks: a comprehensive foundation. Macmillan College, New York.

[4] Hertz, J., Krogh, A., Palmer, R.G. 1991 Introduction to the theory of neural computation.

[5] Antonucci, M., Tirozzi, B., Yarunin N.D. et al. 1994 "Numerical simulation of neural networks with translation and rotation invariant pattern recognition," International Journal of Modern Physics B, vol. 8, no. 11-12, pp. 1529-1541

[6] Pao, Y.H., 1989 Adaptive pattern recognition and neural networks. Addison-Wesley, New York]

[7] Wesley, A., Ekstrom, M.P. 1984 Digital image processing techniques. Academic, New York G.

[8] Bilbro, G.L., White, M., Synder, W. 1988 Image segmentation with neuro computers. In: Eckmiller R, Malsburg CVD (eds) Neural computers. Springer, New York.

[9] Kim, T., Devarajan, B. and Manry, M. "Road extraction from aerial images using neural networks," Proceedings of ASPRS Annual Convention, vol. 3, pp. 146-154

[10] Carpenter, A. and Ross, W.D. 1995 "ART-EMAP: A neural network architecture for object recognition by evidence accumulation," IEEE Transactions on Neural Networks, vol. 6, no. 4, pp. 805-818

[11] Abdallah, M.A., Samu, T.I., Grisson, W.A. 1995 Automatic target identification using neural networks. SPIE Proc Intell Robots Comput Vis XIV 2588:556–565V]

[12] Tang, H.W., Srinivasan, V., Ong, S.H. 1996 Invariant object recognition using a neural template classifier. Image VisComput14(7):473–483

[13] Tou, J.T. and Gonzalez, R.C. Pattern Recognition Principles, Reading, MA: Addison Wesley, 1974

[14] Chua, L.O., Yang, L. 1988 Cellular neural network: theory. IEEE Trans Circuits Syst 35:1257–1272]

[15] Kosko, B. 1988 Bidirectional associative memories. IEEE Trans Syst

[16] Duda, R.O., Hart, P.E. 1973 Pattern classification and scene analysis. Wiley, New York

[17] Kosko, B. 1992 Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence. Prentice-Hall, Englewood Cliffs

[18] Zamparel, M. 1997 Genetically trained cellular neural networks. Neural Network 10(6):1143–1151

[19] Bhattacharyya, S., Dutta, P., Maulik, U. and Nandi, P.K. - Intelligent Technology Volume 1 Number 4A Self Supervised Bi-directional Neural Network (BDSONN) Architecture for Object Extraction Guided by Beta Activation Function and Adaptive Fuzzy Context Sensitive Thresholding.

[20] Bhattacharya, S., Dutta, P., Maulik, U. 2007. Binary object extraction using bi-directional self-organizing neural network (BDSONN) architecture with fuzzy context sensitive thresholding Springer-Verlag London Limited

[21] Ghosh, A. and Sen, A. "Self organizing neural network for multi-level image segmentation," Soft Computing Applications to Pattern Recognition

[22] Ghosh, A., Pal, N.R., Pal, S.K. 1993 Self-organization for object extraction using multilayer neural network and fuzziness measures. IEEE Trans Fuzzy Syst 1(1):54–68

[23] Bhattacharya, S., Pal, P., Bhowmick, S. - A Quantum Multilayer Self Organizing Neural Network (QMLSONN) For Binary Object Extraction From A Noisy Perspective.

[24] Ross, T.J., Ross, T. 1995 Fuzzy logic with engineering applications. McGraw Hill CollegeDiv.

[25] Bhattacharyya S., Dutta P., Maulik U. 2006 A self-supervised bi-directional neural network (BDSONN) architecture for object extraction guided by beta activation function and adaptive fuzzy context sensitive thresholding. Int J Intell Technol 1(4):345–365.