ENHANCED ENERGY BACKPROPAGATION ALGORITHM

Ahmad Hashim Hussein Aal-Yhia
Post-Graduate Institute for Accounting and Financial Studies, University of Baghdad. Baghdad - Iraq

Abstract

The energy backpropagation algorithm (EBP) used the net, which contains two nodes of input layer, hidden layer and output layer. In this paper, we will use a net, which contains three nodes and four nodes of input layer, hidden layer and output layer. This study will compares among times of the learning, times of the identification and times of the converging by using the three nets (2, 3 and 4 nodes) in the Energy back propagation Algorithm. The results of experiments show the nets which contain three nodes and four nodes have better performance for time of the learning than the net which contains two nodes, while the net which contains two nodes has better performance for the time of identification and the time of converging than the nets which contain three nodes and four nodes.

المحسنة خوارزمية توليد الطاقةَ العكسي

احمد هاشم حسين آل يحيى
المعهد العالي للدراسات المحاسبية والمالية ، جامعة بغداد. بغداد – العراق.

الخلاصة

خوارزمية توليدِ الطاقةَ العكسي إستعملتْ الشبكةَ التي تَحتوي عُقَدتان في طبقةِ الادخال وعقدتان في الطبقة الوسطية وعقدتان في طبقة الاخراج. في هذا البحث سَنَستعملُ شبكة التي تَحتوي ثلاث عُقَدِ وأربع عقد في طبقةِ الادخال والطبقة الوسطية وطبقة الاخراج. هذه الدراسةِ سَتُقارنُ بين أوقاتِ التَعَلّم وأوقات التحديد وأوقاتِ الاسترجاع بواسطة إستعمال الشبكاتِ الثلاث التي تحتوي عقدتان وثلاث عقد واربع عقد في خوارزميةِ توليدِ الطاقةَ العكسي. نَتائِج التجاربِ تبين بان الشبكاتَ التي تَحتوي ثلاث عُقَدِ وأربع عُقَدِ لَها أداءُ أفضلُ لوقتِ التَعَلّم مِنْ الشبكةِ التي تَحتوي عقدتان، بينما الشبكة التي تَحتوي عقدتان لها أداءُ أفضلُ لوقتِ التحديد ووقتِ الاسترجاع مِنْ الشبكاتِ التي تَحتوي ثلاث عُقَدِ وأربع عُقَدِ.

Introduction

Neural net can be applied to a wide variety of problems such as: storing and recalling data or patterns, classifying patterns, performing general mappings from input patterns to output patterns, grouping similar patterns, or finding solutions to constrained optimization problems [1]. The NNs provide a greater degree of robustness and fault tolerance [2].

The feedforward backpropagation (FFBP) network is a very popular model in neural networks. It does not have feedback connections, but errors are backpropagated during training. Least mean squared error (LMST) is used [3]. Multilayer feed forward networks trained using the backpropagation learning algorithm [4].

The network edges connect the processing units called neurons. With each neuron input there is associated a weight, representing its relative importance in the set of the neuron's inputs. The inputs' values to each neuron are accumulated through the net function to yield the net value: the net value is a weighted linear combination of the neuron's inputs' values [5].

A backpropagation net can be used to solve problems in many areas [1]. The backpropagation algorithm has the limitation of slow convergence [6] and lengthy training cycles [7], the energy backpropagation (EBP) algorithm overcame those drawbacks [8]. The EBP used net which contains two nodes of input layer, hidden layer and output layer [8], in this study will use nets which contain three nodes and four nodes in order to determine the better net for energy backpropagation algorithm.

A. The Energy Function

One system of updating is to update the units in sequence. The update mechanism posed by Hopfield (1982) chooses the unit randomly. Usually, all processing units must be updated many times before the network reaches a stable state. Each state of the network has an associated "energy" value, which is defined as:

$$E = -1/2 \sum_{j} \sum_{i, i \neq j} t_{ji} v_j v_i \qquad (1)$$

When the network is in a stable state, the energy function is at a minimum, which may be local or global [7]. The existence of such function enables us to prove that the net will converge to stable set of activation, rather than oscillating. The function decreases as the system states change. Such a function needs to be found and watched as the network operation continues from one cycle to another. The least mean squared error is an example of such function. Energy function usage assures a stability of the system that cannot occur without convergence. It is convenient to have one value, that of the energy function specifying the system behavior [9].

The energy function is constant times the sum of products of outputs of different neurons and the connection weight between them, since pairs of neuron outputs are multiplied in each term [3]. The network with two neurons can be represented by four states: (00, 01, 10, 11). The States of three-neuron network can be represented by a cube. In general, a network with n neurons has 2n states and can be represented by an n2-dimensional cube. When a new input is applied, the network moves from vertex to vertex until it stabilizes. If the input vector is partial or incomplete, the network stabilizes to the closest vertex. A number of binary vectors representing different patterns can be stored in the network. Here, in order to store input vectors, the energy equation is used to assign the weight values such that each memory vector corresponds to a stable state or the minimum energy equation of the network [2].

B. Standard backpropagation algorithm

Many applications can be formulated for using (FFBP) network, and the methodology has been a model for most multilayer neural networks. Errors in the output determine measures of hidden layer output errors, which are used as a basis for adjustment of connection weights between the input and hidden layers. Adjusting the two sets of weights between the pairs of layers and recalculating the outputs is an iterative process that is carried on until the errors fall below a tolerance level. Learning rate parameters scale the adjustments to weights. A momentum parameter can also be used in scaling the adjustments from a previous iteration and adding to the adjustments in the current iteration [3].

The backpropagation training algorithm is an iterative gradient algorithm designed to minimize the mean square error (MSE) between the actual output of a multilayer feedforward perceptron and the desired output. It requires continuous differentiable non-linearities. The following assumes a sigmoid logistic nonlinearity [10].

Step 1. Initialize weights and offsets

Set all weights and node offsets to small random values.

Step 2. Present input and desired outputs

Present a continuous valued input vector $x_0, x_1, \ldots x_{N-1}$ and specify the desired outputs $d_0, d_1, \ldots d_{M-1}$. If the net is used as a classifier then all desired outputs are typically set to zero except for that corresponding to the class the input is from. That desired output is 1.The input could be new on each trial or samples from a training set could be presented cyclically until weights stabilize.

Step 3. Calculate actual outputs

Use the sigmoid nonlinearity and formulas as in Figure 1 to calculate outputs $y_0, y_1 \ldots y_{m-1}$.

Step 4. Adapt weights

Use a recursive algorithm starting at the output nodes and working back to the first hidden layer. Adjust weight by

$$W_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i' \qquad (2)$$

In this equation, $w_{ij}(t)$ is the weight from hidden node j or from an input to node j at time t, x_i' is either the output of node j or is an input, η is a gain term, and δ_j is an error term for node j.

If node j is an output node, then

$$\delta_j = y_j (1 - y_j) (d_j - y_j) , \qquad (3)$$

where dj is the desired output of node j and y_j is the actual output.

If node j is an internal hidden node, then

$$\delta_j = x_j' (1 - x_j') \sum_k \delta_k w_{jk} , \qquad (4)$$

Where k is over all nodes in the layers above node j. Interval node thresholds are adapted in a similar manner by assuming they are connection weights on links from auxiliary constant-valued inputs. Convergence is sometimes faster if a momentum term is added and weight changes are smoothed by

$$W_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i' + \alpha (w_{ij}(t) - w_{ij}(t-1)) , \qquad (5)$$

where $0 < \alpha < 1$.

Step 5. Repeat by going to step 2.

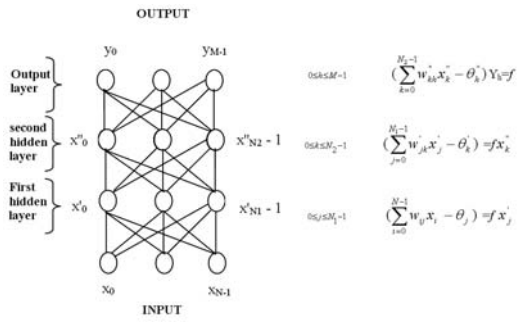

Figure 1: Calculation of output for backpropagation training algorithm [2].

C. The EBP Algorithm

The EBP algorithm consists of two stages: the learning stage and the convergence stage. In EBP algorithm, we used the net, which contains two nodes of input layer, hidden layer and output layer as shown in Figure 2. The size of this net will be the same. This representation of the net enables the learning and convergence processes of parts of the image under consideration [8].

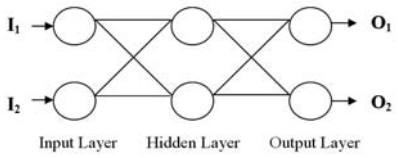

Figure 2: Net of small size.

The learning stage consists of learning algorithm. In this algorithm, the image is divided into rows. Each row is divided into number of parts of two pixels, to be considered as a vector (V). This means there are four states for these vectors as shown in Figure 3.

The vector (V)		Number of net
0	0	0
0	1	1
1	0	2
1	1	3

Figure 3: The four states for the vectors.

Therefore, the learning process in the EBP algorithm for any image will result in four sets of learning weights matrices at maximum. Each set consist of two matrices: W1 and W2. The W1 represents the weights from input layer to hidden layer, and the W2 represents the weights from hidden layer to output layer. There are four sets of biases arrays at maximum. Each set consists of two arrays, which are called Wh and Wo. The Wh represents the biases of nodes of hidden layer, and Wo represents the biases of nodes of output layer. Each matrix of W1 and W2 will be of size 2x2, and each array of Wh and Wo will be of size 2x1.

Hence, each vector V, as shown in Figure 3, will represent its own net. Each net has (W1 and W2, Wh and Wo). Therefore, we have only four nets as shown in Figure 4.

Each vector V will be replaced by a number. This means we will replace all the parts (the vectors) in the image with numbers. These numbers represent the net as shown in Figure 3 and Figure 4[8].

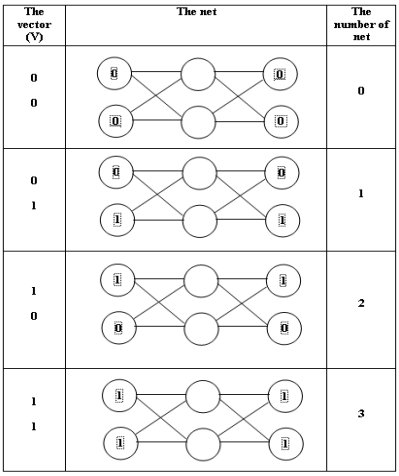

Figure 4: The four vectors with its own nets and numbers of nets.

The learning process of the EBP algorithm for all vectors in the image will pass through two steps:

1. Testing whether the system has learned current vector before or not: The binary vector is examined. If it is saved with its learning weights matrices, then it is learned. Therefore, we will replace this vector by the number of its own net; otherwise, the vector is not learned.
2. Learning the vector: If the vector is not learned, it is entered to the backpropagation learning algorithm, then save its learning weights matrices, and save the number of net instead of vector [8].

The convergence stage consists of two algorithms: the algorithm of identification and the algorithm of convergence. The identification algorithm extracts the nearest image for the unknown input image using the energy function, depending on the learning weight matrix W2 for both vectors of unknown input image and vectors of images in the database. Notice that the learning weight matrix W1 is similar to some of its parameters. Therefore, we use only the learning weight matrix W2 in identification algorithm. This similarity makes the identification process not active. The identification process is showed in Figure 5 [8].

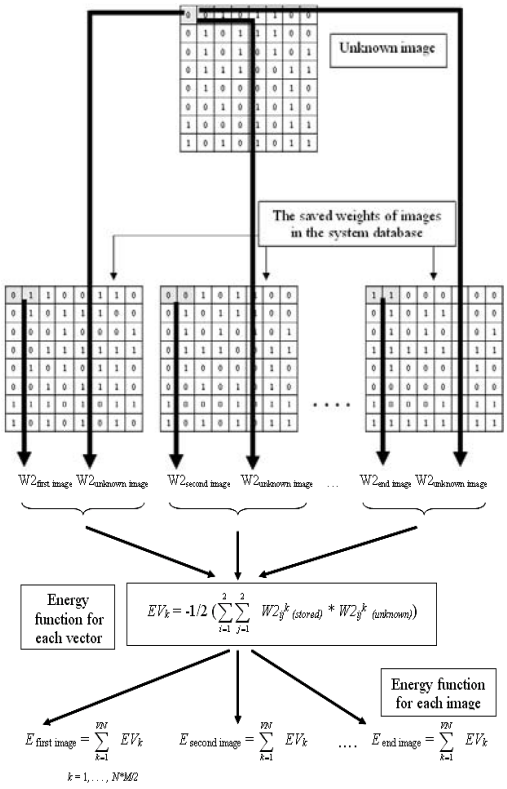

Figure 5: Operation of identification the nearest image by energy function.

The identification process consists of the following steps:

1. Use the learning weights matrix W2 for current vector in the unknown input image, with each W2 for the vectors in the images saved in the database, and extracts the Energy function (*EV*) between them. Thus, each image in the database will have value of its own Energy function. Therefore, extract the value of the first minimum energy function and the value of the second minimum Energy function. Then, extract the value of difference between them (Difference Next Close (DNC)).
2. Examine whether the system has learned the unknown input image before or not using the DNC value. If it is large value, then the unknown input image is learned. Therefore, the stored image with the minimum energy function, representing the nearest image for the unknown input image, is converged. However, if the DNC is small value, the unknown input image is not learned. Therefore, exit from the system. The DNC value depends on the image density and size [8].

In the convergence algorithm, if unknown input image is learned, then converges the stored image with the minimum energy function (E). This is the nearest one for the unknown input image. In the same way of the learning algorithm, we divide the image into rows. Each row is divided into number of parts of two pixels size to be considered as a vector (V). Depending on the number of net and the backpropagation convergence algorithm, the vector (V) is converged. If the vector is converged by the backpropagation convergence algorithm before, it is converged by using number of net [8].

In Section 2, the EBP algorithm will be introduced by using net, which contains three nodes and four nodes. In Section 3, the performance of the EBP algorithm by using net, which contains two nodes, will be compared to the performance of the EBP algorithm by using net, which contains three nodes and four nodes. Section 4 concludes the work.

The EBP algorithm by using net contains (3 or 4) nodes:

In the EBP, we will use net, which contains three nodes and four nodes of input layer, hidden layer and output layer. The net which contain three nodes is shown in Figure 6.

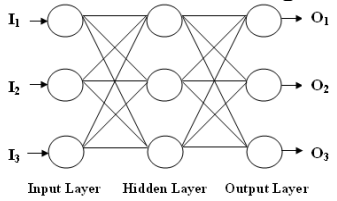

Figure 6: Net of small size which contains 3 nodes.

A. The learning Algorithm:

In the learning algorithm, we will divide the image into rows. Each row is divided into number of parts of three (or four) pixels, to be considered as a vector (V). This means there are eight states (or sixteen) for these vectors as shown in Figure 7 and Figure 8.

The vector (V)			Number of net
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Figure 7: The eight states for the vector (3 nodes) with its own number of nets.

The vector (V)				Number of net
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Figure 8: The sixteen states for the vector (4 nodes) with its own number of nets.

If the net contain 3 nodes (or 4 nodes), the learning process in the EBP algorithm for any image will result in eight (or sixteen) sets of learning weights matrices at maximum. Each set consist of two matrices: W1 and W2. There are eight (or sixteen) sets of biases arrays at maximum. Each set consists of two arrays, which are called Wh and Wo.

Each matrix of W1 and W2 will be of size 3x3 (or 4x4), and each array of Wh and Wo will be of size 3x1(or 4x1).

Hence, each vector V, as shown in Figure 7 (or Figure 8), will represent its own net. Each net has (W1 and W2, Wh and Wo). Therefore, we have only eight (or sixteen) nets.

Each vector V will be replaced by a number. This means we will replace all the parts (the vectors) in the image with numbers. These numbers represent the net as shown in Figure 7.

The following steps present the learning algorithm by using net contains 3 (or 4) nodes:

Step1: For x=1 to N {*N: length of image*}
Step2: For y=1 to M step 3 (or step 4)
{*M: width of image*}
{*step 3: when used net contains 3 nodes. step 4: when used net contains 4 nodes*}
Step3:
IF binary vector is learned
Then Save the number of net for this vector and return to take the other vector.
Else enter the vector to the backpropagation learning algorithm, save learning weights matrices W_1 & W_2, and save number of net for this vector and return to take the other vector.

B. The identification Algorithm

The following steps present the identification algorithm by using net contains 3 (or 4) nodes:

For x=1 to N {*N: length of image*}
For y=1 to M {*M: width of image*}
BEGIN
For all stored images do
Begin
For all rows of stored images do
For the vector in the input image and learning weight matrix W_2 do
Begin

$$EV_k = -1/2 \left(\sum_{i=1}^{t} \sum_{j=1}^{t} W_{2ij}{}^{k}{}_{(input)} * W_{2ij}{}^{k}{}_{(stored)} \right)$$

{*Energy function for each vector*}
{*t: t is 3 when used net contains 3 nodes and t is 4 when used net contains 4 nodes}*
End;

$$E = \sum_{k=1}^{VN} EV_k$$ {*Energy function for each image*}

End;
END
DNC= E1–E2
{E1: first minimum energy function,
E2: second minimum energy function}
IF DNC is large value
Then unknown input image is learned; therefore the stored image with the minimum Energy function (*E)* representing the nearest image for the unknown input image is converged.
IF DNC is small value
Then unknown input image is not learned, therefore exit from the system.

C. The convergence Algorithm

In convergence algorithm, in the same way of the learning algorithm, we divide the image into rows. Each row is divided into number of parts of two pixels size to be considered as a vector (V).

The following steps present the convergence algorithm by using net contains 3 (or 4) nodes:

For x=1 to N {*N: length of image*}
For y=1 to M step 3 (or step 4)
{*M: width of image*}
{*step 3: when used net contains 3 nodes.*
step 4: when used net contains 4 nodes}
For i=1 to NC {*NC: the counter for numbers of net saved instead of the vectors. NC=n*m/3 if used net contains 3 nodes, and NC=n*m/4 if used net contains 4 nodes}*
IF the current vector is not converged by the backpropagation convergence algorithm before,
Then enter the current vector into the backpropagation convergence algorithm and save the produced vector into the array of Converged Image (CI) and return to take the other vector.
CI(x,y) = Vi(1,1)
CI(x,y+1) = Vi(1,2)
CI(x,y+2) = Vi(1,3)
CI(x,y+3) = Vi(1,4)

Else the vector is obtained by the number of net. Then, save the produced vector into the Converged Image (CI).
CI(x,y) = Vi(1,1)
CI(x,y+1) = Vi(1,2)
CI(x,y+2) = Vi(1,3)
CI(x,y+3) = Vi(1,4)
Endfor;
Endfor;
Endfor.

Experiments and Results

This section will be devoted to demonstrate the implementation of the EBP by using the nets which contain (2,3 and 4) nodes with experimental results and discussions. The experimental results are implemented on a Pentium IV, 2400 MHz using programs written in MATLAB. In our experiments, some samples of images are captured using database of fingerprint images [11, 12] and images of Baghdad city from space [13]. These images are of BMP format. Figure 9 shows images samples, which are used in the experiments.

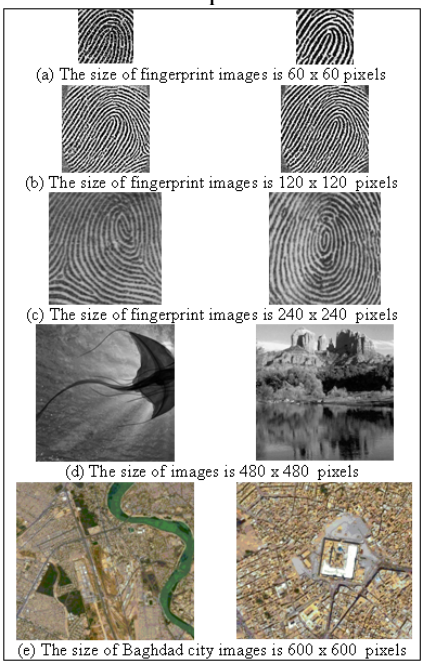

Figure 9: Samples of used images in the experiments.

The time of learning and time of identification and time of convergence in the EBP algorithm by using images with different sizes (60x60, 120x120, 240x240, 480x480 and 600x600 pixel) was measured. The net has only learned two images. The time averages were taken for five different readings. Table 1 shows the time averages of learning process when the used net contains 2, 3 and 4 nodes. Table 2 shows the time averages of identification process when the used net contains 2, 3 and 4 nodes. Table 3 shows the time averages of convergence process when the used net contains 2, 3 and 4 nodes. Table 4 shows the averages of the entire convergence time (identification time + convergence time) in the EBP algorithm when the used net contains 2, 3 and 4 nodes.

Table 1: The time averages of learning process in the EBP algorithm.

Size of image (Pixel)	The time averages of the learning process in EBP algorithm (Second)		
	The net contains 2 nodes	The net contains 3 nodes	The net contains 4 nodes
60x60	0.69	0.77	1.14
120x120	3	2.39	4.11
240x240	32.46	15.42	20.86
480x480	454.23	190.03	169.71
600x600	1118.4	511.66	385.71

Table 2: The time averages of identification process in the EBP algorithm.

Size of image (Pixel)	The time averages of the identification process in EBP algorithm (Second)		
	The net contains 2 nodes	The net contains 3 nodes	The net contains 4 nodes
60x60	0.97	2.04	5.44
120x120	4.12	8.33	22.72
240x240	16.09	33.25	102.53
480x480	65.22	133.81	368.6
600x600	85.93	160.01	458.43

Table 3: The time averages of convergence process in the EBP algorithm.

Size of image (Pixel)	The time averages of the convergence process in EBP algorithm (Second)		
	The net contains 2 nodes	The net contains 3 nodes	The net contains 4 nodes
60x60	0.11	0.12	0.19
120x120	0.38	0.47	0.74
240x240	1.64	1.97	3.15
480x480	7.46	8.86	12.42
600x600	11.62	12.01	16.91

Table 4: The averages of the entire convergence time in the EBP algorithm.

Size of image (Pixel)	The entire time of converging (identification time + convergence time) in EBP (Second)		
	The net contains 2 nodes	The net contains 3 nodes	The net contains 4 nodes
60x60	1.08	2.16	5.63
120x120	4.5	8.8	23.46
240x240	17.73	35.22	105.68
480x480	72.68	142.67	381.02
600x600	97.55	172.02	475.34

The results show the learning process in the EBP algorithm by using the net which contains 2 nodes is faster than the EBP algorithm when the used net contains 3 and 4 nodes when the size of used image is 60x60 pixels and the results show the learning process in the EBP algorithm by using the net which contains 3 nodes is faster than the EBP algorithm when the used net contains 2 and 4 nodes when the size of used image is 120x120 and 240x240 pixels and the results show the learning process in the EBP algorithm by using the net which contains 4 nodes is faster than the EBP algorithm when the used net contains 2 and 3 nodes when the size of used image is 480x480 and 600x600 pixels.

The results show that the identification process and convergence process and the entire convergence time in the EBP algorithm by using the net which contains 2 nodes are faster than the EBP algorithm by using the net which contains 3 and 4 nodes at all the sizes of images.

Conclusion

The energy backpropagation algorithm (EBP) used the net, which contains two nodes of input layer, hidden layer and output layer. In order to develop the performance of the EBP algorithm, we used in this research the net which contains three nodes and four nodes of input layer, hidden layer and output layer.

This study compared among times of the learning, times of the identification and times of the converging by using the three nets (2, 3 and 4 nodes) in the Energy backpropagation Algorithm.

The EBP algorithm by using the three nets (2, 3 and 4 nodes) were implemented and tested on images of different sizes. The samples of the images were taken from a database of fingerprint images and images of Baghdad city. The images are of BMP format. The time of learning, identification and convergence processes of the EBP algorithm were measured. The results indicate the learning process in the EBP algorithm by using the net which contains 4 nodes has better performance when we use the large size images while the learning process in the EBP algorithm by using the net which contains 3 nodes has better performance when we use the smallest size images.

The results indicate the identification and convergence processes in the EBP algorithm by using the net which contains 2 nodes has better performance than the nets which contain three nodes and four nodes. However, we plan to make more experiments and practical implementations of the EBP algorithm.

References

1. Fausett, L., **1994**. *Fundamentals of Neural Networks*. (1st ed.), USA: Prenice-Hall.
2. Kulkarni, A., **2001.** *Computer Vision and Fuzzy-Neural Systems.* USA: Prentice Hall.
3. Vallurn, B., and Hayoriva, V., **1996**. *C++ Neural Network and Fuzzy Logic*. (2nd ed.), USA: MIT Publishing.
4. Rumelhart, D., Hinton, G., and Williams, R., **1986**. Learning internal representations by error propagation, Parallel Distributed Processing: Explorations into the Microstructure of Cognition. Cambridge, MA: MIT Press, vol. **2**:318-362
5. Stoeva, S., and Nikov, A., **2000**. A fuzzy backpropagation algorithm, *Fuzzy Sets and Systems*, **112**: 27–39.
6. Wu, J., **1994**. *Neural Networks and Simulation Methods*. New York: Marcel Dekker.
7. Kosko, B., **1992**. Neural Networks and Fuzzy Systems: *A Dynamical Systems Approach to Machine Intelligence*. USA: Prentice-Hall international.
8. Aal-Yhia, A. H., and Sharieh, A., **2007**. An energy backpropagation algorithm: *World Congress on Engineering*, London. **1**:122-127.
9. Kinnebrock, W., **1995**. *Neural Network. Fundamentals, Applications, Examples.* USA: Prentice-Hill, Inc.
10. Lippmann, R., **1987**. An introduction to computing with neural nets, *IEEE*: 4-22.
11. Biometrix 8th WWW Fp-images. **2003**. (2nd ed.) Retrieved August 10. Available: http://www.biometrix.at/fp-images.zip
12. Neurotchnologija 3th WWW Fingercell_Sample_DB. **2004**. Retrieved May 25. Available: www.neurotchnologija.com/download/fingercell_sample_DB.zip.
13. mage@2005 DigitalGlobe 8th HTTP sample.JPEG. **2005**. (n. d.) Retrieved August 15. vailable:http://earth.google.com/.