ISSN 2047-3338

Effect of Training Algorithms on the Accuracy in Iris Patterns Recognition using Neural Networks

M. Gopikrishnan[1] and Dr. T. Santhanam[2]

[1]Department of MCA, Prathyusha Institute of Technology and Management, Chennai-602 025, India
[2]Department of Computer Science, D. G. Vaishmav College, Chennai- 600 106, India
mgkc71@yahoo.com; santhanam_dgvc@yahoo.com

***Abstract*– A biometric system provides automatic identification of an individual based on a unique feature or characteristic possessed by the individual. Iris recognition is regarded as the most reliable and accurate biometric identification system available. An approach for accurate Biometric Recognition and identification of Human Iris Patterns using Neural Network has been illustrated in [10]. The same authors tried by reducing the size of the templates from 20 X 480 to 10 X 480 and concluded that this resulted in saving of computation effort with no loss in accuracy. In this paper, based on the accurate methodology [10[, we extend the work for optimization for Iris Patterns recognition using various neural training model algorithms. The results from the neural models trained by Levenberg– Marquardt algorithm is found to provide accuracy in recognition better than the methods presented in the literature.**

***Index Terms*– Iris Recognition, Biometric Identification, Pattern Recognition and Automatic Segmentation**

I. INTRODUCTION

BASED on the results reported in the literature [1] – [9] for Iris recognition, Gopikrishnan et al., [10] studied Hamming distance coupled with Neural Network based iris recognition techniques. Perfect recognition on a set of 150 eye images has been achieved through this approach ; Further, Tests on another set of 801 images resulted in false accept and false reject rates of 0.0005% and 0.187% respectively, providing the reliability and accuracy of the biometric technology. In subsequent papers same authors provided [11], [12] results of iris recognition performed on a reduced size template, applying Hamming distance, Feed forward back propagation, Cascade forward back propagation, Elman forward back propagation and perceptron. It has been established that the method suggested applying perceptron, using hardlim training function and learnp learning function, provides the best accuracy in respect of iris recognition with no major additional computational complexity. This paper uses the CASIA iris image database collected by Institute of Automation, Chinese Academy of Sciences [13].

In this paper, based on the accurate methodology suggested by Gopikrishnan et al., we extend the work for optimization for Iris Patterns recognition using various neural training model algorithms. The results from the neural models trained by Levenberg – Marquardt algorithm is found to provide accuracy in recognition better than the methods presented in the literature.

II. ARTIFICIAL NEURAL NETWORKS (ANN)

ANN's are biologically inspired computer programs to simulate the way in which the human brain process information. It is a very powerful approach for building complex and nonlinear relationship between a set of input and output data. The power of computation comes from connection in a network. Each neuron has weighted inputs, simulation function, transfer function and output. The weighted sum of inputs constitutes the activation function of the neurons. The activation signal is passed through a transfer function which introduces non-linearity and produces the output. During training process, the inter-unit connections are optimized until the error in prediction is minimized. Once the network is trained, new unseen input information is entered to the network to calculate the test output.

There are many types of neural network for various applications available in the literature. The most commonly used and simplest network architecture called Feed–Forward Back propagation neural network (FFBPNN) [14] and Cascade Forward Back propagation neural network (CFBPNN) [15] shown in Fig. 1 are used in this work.

FFBPNN and CFBPNN consist of three layers: an input layer, an output layer and an intermediate or hidden layer. The neurons in the input layer only act as buffer for distributing the input signals to neuron in hidden layer. Each neuron in hidden layer sums up its input signal after

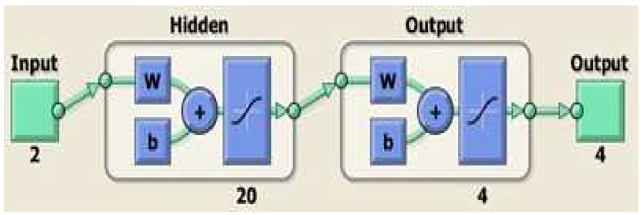


Fig. 1: Feed & Cascade– Forward Back propagation

weighting them and computes it outputs. Training a network consists of adjusting its weights using learning algorithms. The different training functions [14], [15] used in this work are explained briefly in Sections 2.1 to 2.14. All tthese algorithms scan train any network as long as the weight, net input and transfer functions have derivative functions.

A. Levenberg-Marquardt Algorithm [16]

Back propagation is used to calculate the Jacobian jX of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to Levenberg-Marquardt,

*jj = jX * jX*

*je = jX * E*

*dX = -(jj+I*mu) \ je*

where E is all errors and I is the identity matrix. The adaptive value MU is increased by MU_INC until the change above results in a reduced performance value. The change is then made to the network and mu is decreased by MU_DEC. The parameter MEM_REDUC indicates how to use memory and speed to calculate the Jacobian jX. If MEM_REDUC is 1, then TRAINLM runs the fastest, but can require a lot of memory. Increasing MEM_REDUC to 2, cuts some of the memory required by a factor of two, but slows TRAINLM somewhat. Higher values continue to decrease the amount of memory needed and increase training.

B. TRAINBFG Algorithm [16]

In this algorithm back propagation is used to calculate derivatives of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to the following:

*X = X + a * dX;*

where dX is the search direction. The parameter a is selected to minimize the performance along the search direction. The line search function searchFcn is used to locate the minimum point. The first search direction is the negative of the gradient of performance. In succeeding iterations the search direction is computed according to the following formula:

dX = -H\gX;

where gX is the gradient and H is an approximate Hessian matrix.

C. TRAINBR Algorithm [16]

Bayesian regularization minimizes a linear combination of squared errors and weights. It also modifies the linear combination so that at the end of training the resulting network has good generalization qualities. This Bayesian regularization takes place within the Levenberg-Marquardt algorithm. Back propagation is used to calculate the Jacobian jX of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to Levenberg-Marquardt,

*jj = jX * jX*

*je = jX * E*

*dX = -(jj+I*mu) \ je*

where E is all errors and I is the identity matrix.

The adaptive value MU is increased by MU_INC until the change shown above results in a reduced performance value. The change is then made to the network and mu is decreased by MU_DEC. The parameter MEM_REDUC indicates how to use memory and speed to calculate the Jacobian jX. If MEM_REDUC is 1, then TRAINLM runs the fastest, but can require a lot of memory. Increasing MEM_REDUC to 2 cuts some of the memory required by a factor of two, but slows TRAINLM somewhat. Higher values continue to decrease the amount of memory needed and increase the training times.

D. TRAINCGF Algorithm [16]

This algorithm is a network training function that updates weight and bias values according to the conjugate gradient back-propagation with Fletcher-Reeves updates.

E. TRAINCGP Algorithm [16]

This algorithm is a network training function that updates weight and bias values according to the conjugate gradient back propagation with Polak-Ribiere updates.

F. TRAINGD Algorithm [16]

This algorithm uses back propagation is used to calculate derivatives of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to gradient descent:

*dX = lr * dperf/dX*

F. TRAINGDM Algorithm [16]

In TRAINGDM algorithm backpropagation is used to calculate derivatives of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to gradient descent with momentum,

*dX = mc*dXprev + lr*(1-mc)*dperf/dX*

where dXprev is the previous change to the weight or bias.

G. TRAINGDA Algorithm [16]

In this algorithm back-propagation is used to calculate derivatives of performance DPERF with respect to the weight and bias variables X. Each variable is adjusted according to gradient descent:

*dX = lr*dperf/dX*

Each of epoch, if performance decreases toward the goal, then the learning rate is increased by the factor lr_inc. If performance increases by more than the factor max_perf_inc, the learning rate is adjusted by the factor lr_dec and the change, which increased the performance, is not made.

H. TRAINGDX Algorithm [16]

In TRAINGDX algorithm back-propagation is used to calculate derivatives of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to the gradient descent with momentum.

*dX = mc*dXprev + lr*mc*dperf/dX*

where dXprev is the previous change to the weight or bias. For each epoch, if performance decreases toward the goal, then the learning rate is increased by the factor lr_inc. If performance increases by more than the factor max_perf_inc, the learning rate is adjusted by the factor lr_dec and the change, which increased the performance, is not made.

I. TRAINOSS Algorithm [16]

This algorithm uses back-propagation to calculate derivatives of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to the following:

*X = X + a*dX;*

where dX is the search direction. The parameter a is selected to minimize the performance along the search direction. The line search function searchFcn is used to locate the minimum point. The first search direction is the negative of the gradient of performance. In succeeding iterations the search direction is computed from the new gradient and the previous steps and gradients according to the following formula:

*dX = -gX + Ac*X_step + Bc*dgX;*

where gX is the gradient, X_step is the change in the weights on the previous iteration, and dgX is the change in the gradient from the last iteration.

J. TRAINR [16]

For each epoch, all training vectors (or sequences) are each presented once in a different random order with the network and weight and bias values updated accordingly after each individual presentation.

K. TRAINRP Algorithm [16]

Back-propagation is used in this algorithm to calculate derivatives of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to the following:

*dX = deltaX.*sign(gX);*

where the elements of deltaX are all initialized to delta0 and gX is the gradient. At each iteration the elements of deltaX are modified. If an element of gX changes sign from one iteration to the next, then the corresponding element of deltaX is decreased by delta_dec. If an element of gX maintains the same sign from one iteration to the next, then the corresponding element of deltaX is increased by delta_inc.

L. TRAINSCG Algorithm [16]

In TRAINSCG algorithm back propagation is used to calculate derivatives of performance PERF with respect to the weight and bias variables X. The scaled conjugate gradient algorithm is based on conjugate directions, as in TRAINCGP, TRAINCGF and TRAINCGB, but this algorithm does not perform a line search at each iteration.

III. EXPERIMENTAL RESULTS

The performance of the proposed improved methodology is evaluated with CASIA database (the institute of Automation, Chinese Academy of Sciences). The CASIA data base contain nearly 4500 iris images at (320X280). The experiments were carried out in Intel Core 2 Duo processor with templates. This is a real world application level simulation.

The goal of training is to find an optimum solution for network, so that output arises from real answer. In network training, each input vector and opposite output vector make a couple. Usually, a neural network trains with more couples. In neural network, primary weights are important because this comparison is dependent on different elements such as input data, weights, goal error parameter and the aim of network usage.

In Table (1-4) and Fig. (2-5) (using a 3D Column chart) the results of comparison between different training functions in neural network are shown. Levenberg–Marquardt algorithm is found to provide accuracy in recognition better than the methods presented in the literature and its performance is better than other training functions.

Table 1: Comparison between experimental Results using Layer 2 Feed Forward Back Propagation										
Training Function	**Learning Function**	**EXP 1 & 2**	**EXP 1 & 3**	**EXP 1 & 4**	**EXP 1 & 5**	**EXP 1 & 6**	**EXP 1 & 7**	**EXP 1 & 8**	**EXP 1 & 9**	**EXP 1&10**
BFG	GDM	0.18	0.15	0.18	0.16	0.18	0.17	0.17	0.18	0.18
CGB	GDM	0.18	0.14	0.17	0.18	0.17	0.17	0.16	0.16	0.22
CGF	GDM	0.23	0.15	0.19	0.24	0.21	0.18	0.17	0.17	0.19
CGP	GDM	0.17	0.18	0.17	0.15	0.22	0.17	0.16	0.16	0.18
GD	GDM	0.39	0.39	0.40	0.42	0.40	0.40	0.39	0.39	0.42
GDM	GDM	0.40	0.40	0.40	0.42	0.40	0.41	0.39	0.40	0.40
GDA	GDM	0.37	0.37	0.38	0.40	0.37	0.37	0.38	0.36	0.39
GDX	GDM	0.39	0.37	0.40	0.35	0.39	0.37	0.38	0.37	0.38
LM	GDM	0.16	0.12	0.15	0.14	0.16	0.16	0.15	0.14	0.15
0SS	GDM	0.18	0.15	0.19	0.17	0.19	0.18	0.18	0.18	0.17
R	GDM	0.21	0.18	0.22	0.19	0.21	0.21	0.20	0.20	0.20
RP	GDM	0.18	0.15	0.19	0.16	0.18	0.18	0.17	0.18	0.17
SCG	GDM	0.17	0.17	0.17	0.15	0.17	0.18	0.18	0.18	0.17

Fig. 2: Verification of Performance using 3D Column Chart
(Layer 2 Feed Forward Back Propagation)

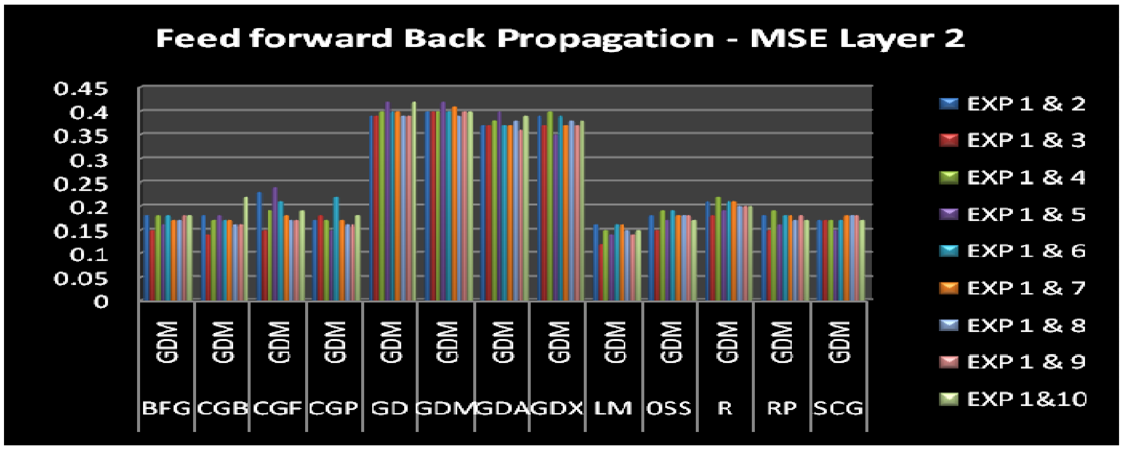


Table 2: Comparison between experimental Results using Layer 2 with Cascade Forward Back Propagation										
Training Function	**Learning Function**	**EXP 1 & 2**	**EXP 1 & 3**	**EXP 1 & 4**	**EXP 1 & 5**	**EXP 1 & 6**	**EXP 1 & 7**	**EXP 1 & 8**	**EXP 1 & 9**	**EXP 1&10**
BFG	GDM	0.13	0.13	0.18	0.14	0.16	0.16	0.16	0.16	0.15
CGB	GDM	0.18	0.11	0.20	0.15	0.16	0.20	0.15	0.17	0.15
CGF	GDM	0.21	0.14	0.20	0.17	0.18	0.18	0.19	0.19	0.17
CGP	GDM	0.17	0.12	0.24	0.14	0.16	0.16	0.15	0.15	0.16
GD	GDM	0.38	0.39	0.39	0.39	0.38	0.39	0.40	0.41	0.38
GDM	GDM	0.40	0.38	0.40	0.42	0.39	0.39	0.39	0.40	0.39
GDA	GDM	0.35	0.33	0.35	0.33	0.37	0.35	0.34	0.35	0.42
GDX	GDM	0.36	0.34	0.37	0.35	0.36	0.35	0.36	0.36	0.36
LM	GDM	0.12	0.08	0.12	0.09	0.11	0.11	0.11	0.11	0.11
0SS	GDM	0.19	0.13	0.12	0.15	0.18	0.18	0.17	0.17	0.15
R	GDM	0.18	0.15	0.18	0.16	0.18	0.18	0.18	0.18	0.18
RP	GDM	0.16	0.12	0.16	0.13	0.15	0.15	0.15	0.15	0.15
SCG	GDM	0.15	0.19	0.15	0.18	0.15	0.17	0.20	0.23	0.14

Fig. 3: Verification of Performance using 3D Column Chart
(Layer 2 with Cascade Forward Back Propagation)

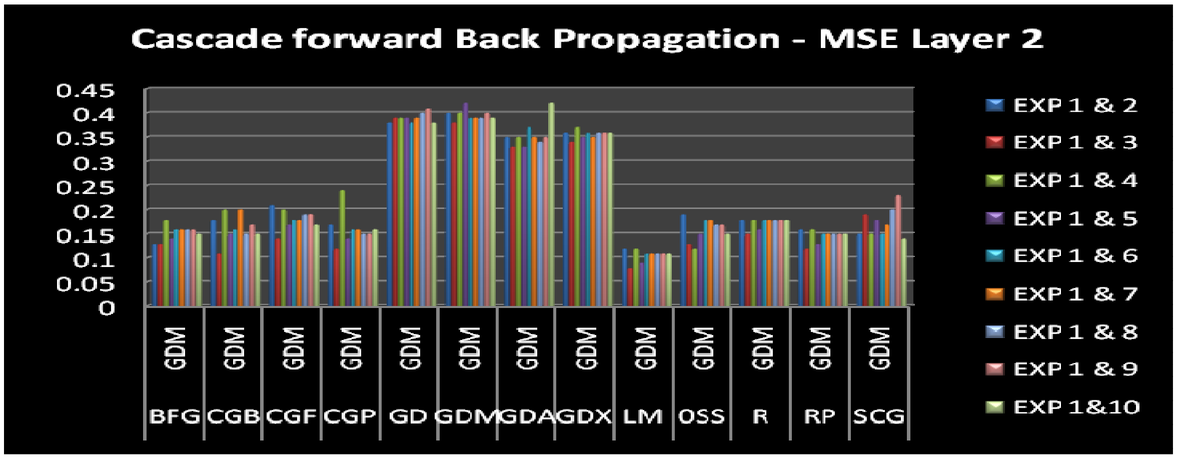


Table 3: Comparison between experimental Results using Layer 10 with Feed Forward Back Propagation

Training Function	Learning Function	EXP 1 & 2	EXP 1 & 3	EXP 1 & 4	EXP 1 & 5	EXP 1 & 6	EXP 1 & 7	EXP 1 & 8	EXP 1 & 9	EXP 1&10
BFG	GDM	0.18	0.15	0.19	0.16	0.17	0.18	0.17	0.17	0.17
CGB	GDM	0.17	0.14	0.17	0.15	0.17	0.16	0.17	0.16	0.21
CGF	GDM	0.18	0.15	0.19	0.25	0.18	0.18	0.18	0.17	0.18
CGP	GDM	0.17	0.14	0.17	0.25	0.18	0.16	0.16	0.16	0.16
GD	GDM	0.41	0.40	0.38	0.39	0.40	0.41	0.40	0.41	0.41
GDM	GDM	0.40	0.41	0.40	0.41	0.41	0.40	0.40	0.40	0.41
GDA	GDM	0.37	0.33	0.38	0.35	0.36	0.36	0.37	0.39	0.38
GDX	GDM	0.38	0.39	0.40	0.36	0.40	0.37	0.38	0.38	0.35
LM	GDM	0.16	0.13	0.16	0.14	0.20	0.15	0.15	0.15	0.15
0SS	GDM	0.18	0.15	0.19	0.17	0.18	0.18	0.18	0.18	0.18
R	GDM	0.21	0.18	0.22	0.20	0.20	0.20	0.21	0.22	0.20
RP	GDM	0.19	0.15	0.18	0.16	0.18	0.18	0.18	0.17	0.17
SCG	GDM	0.17	0.14	0.18	0.15	0.17	0.16	0.17	0.17	0.17

Fig. 4: Verification of Performance using 3D Column Chart
(Layer 10 Feed Forward Back Propagation)

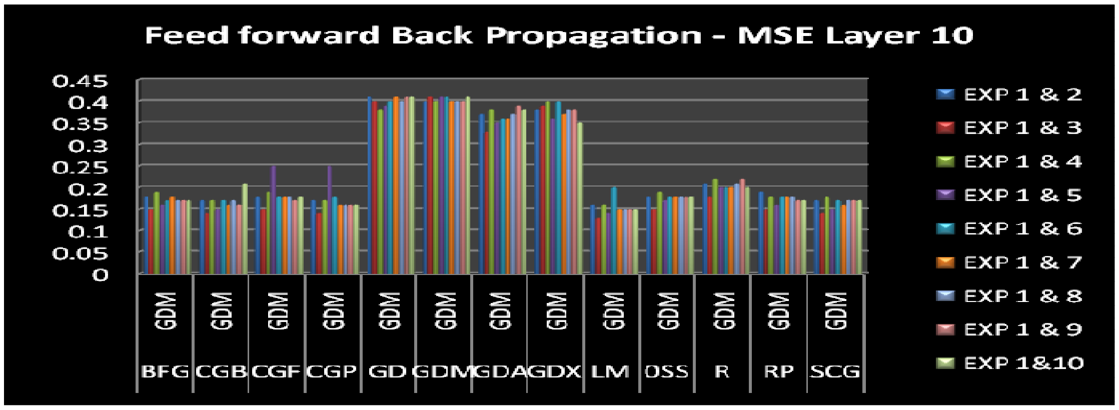
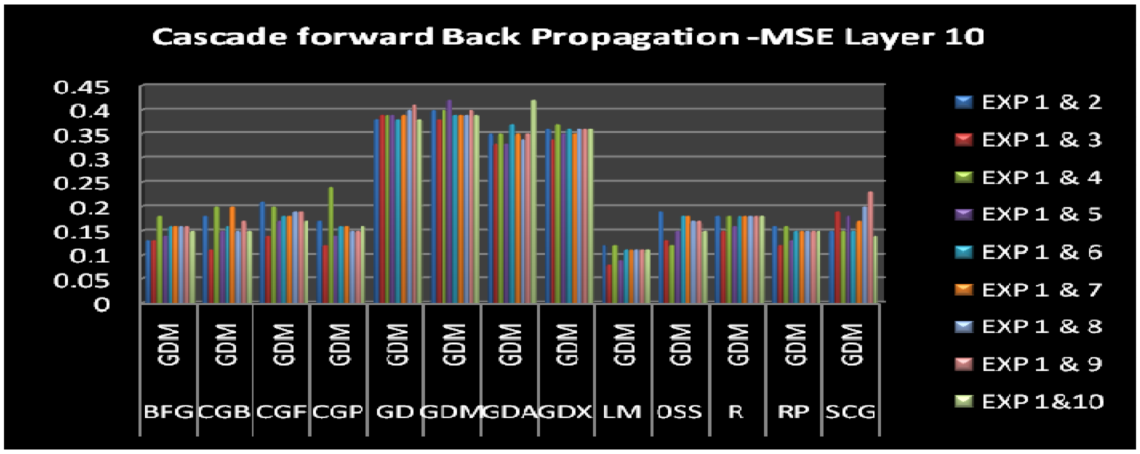


Table 4: Comparison between experimental Results using Layer 10 with Cascade Forward Back Propagation

Training Function	Learning Function	EXP 1 & 2	EXP 1 & 3	EXP 1 & 4	EXP 1 & 5	EXP 1 & 6	EXP 1 & 7	EXP 1 & 8	EXP 1 & 9	EXP 1&10
BFG	GDM	0.13	0.13	0.18	0.14	0.16	0.16	0.16	0.16	0.15
CGB	GDM	0.18	0.11	0.20	0.15	0.16	0.20	0.15	0.17	0.15
CGF	GDM	0.21	0.14	0.20	0.17	0.18	0.18	0.19	0.19	0.17
CGP	GDM	0.17	0.12	0.24	0.14	0.16	0.16	0.15	0.15	0.16
GD	GDM	0.38	0.39	0.39	0.39	0.38	0.39	0.40	0.41	0.38
GDM	GDM	0.40	0.38	0.40	0.42	0.39	0.39	0.39	0.40	0.39
GDA	GDM	0.35	0.33	0.35	0.33	0.37	0.35	0.34	0.35	0.42
GDX	GDM	0.36	0.34	0.37	0.35	0.36	0.35	0.36	0.36	0.36
LM	GDM	0.12	0.08	0.12	0.09	0.11	0.11	0.11	0.11	0.11
0SS	GDM	0.19	0.13	0.12	0.15	0.18	0.18	0.17	0.17	0.15
R	GDM	0.18	0.15	0.18	0.16	0.18	0.18	0.18	0.18	0.18
RP	GDM	0.16	0.12	0.16	0.13	0.15	0.15	0.15	0.15	0.15
SCG	GDM	0.15	0.19	0.15	0.18	0.15	0.17	0.20	0.23	0.14

Fig. 5: Verificationof Performance using 3D Column Chart
(Layer10 with Cascade Forward Back Propagation)



IV. CONCLUSIONS

Based on the improved iris recognition method already proposed in the literature in this paper optimization for Iris Patterns recognition using various neural training model algorithms is carried out. The results of the neural model trained by Levenberg – Marquardt algorithm is found to provide accuracy in recognition better than other training methods. A comparison of results obtained for the two experiments show the results of the improved method exhibit an encouraging performance as for as the accuracy is concerned especially on the CASIA data set [13]. The performance evaluation and comparisons indicate that the proposed method is a viable and very efficient method for iris recognition resulting in lesser time complexity and space requirement.

ACKNOWLEDGMENTS

The iris CASIA dataset is available on the web at http:// www. sinobiometrics.com / english/ Iris%20Data bases.asp[13].

REFERENCES

[1] S. Sanderson, J. Erbetta. Authentication for secure environments based on iris scanning technology. IEE Colloquium on Visual Biometrics, 2000.
[2] E. Wolff. Anatomy of the Eye and Orbit. 7th edition. H. K. Lewis & Co. LTD, 1976.
[3] R. Wildes. Iris recognition: an emerging biometric technology. Proceedings of the IEEE, Vol. 85, No. 9, 1997.
[4] Yingzi Du, Robert Ives, Bradford Bonney and Delores Etter, " Analysis of partial iris recognition" by spie – 2005.
[5] Woodward, N.M. Orlans, and P.T. Higgins, Biometrics, The McGraw-Hill Company, Berkeley, California, 2002.

[6] J. Daugman, “High Confidence Visual Recognition of Persons by a Test of Statistical Independence”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 11, pp. 1148-1161, 1993.

[7] R.P. Wildes, J.C. Asmuth, G.L. Green, S.C. Hsu, R.J. Kolczynski, J.R. Matey, and S.E. McBride, “A Machine Vision System for Iris Recognition”, Mach. Vision Application, Vol. 9, pp.1-8, 1996.

[8] W.W. Boles and B. Boashash, “A Human Identification Technique Using Images of the Iris and Wavelet Transform”, IEEE Transactions on Signal Processing, Vol. 46, No. 4, pp. 1185-1188, 1998.

[9] J.E. Siedlarz, “Iris: More detailed than a fingerprint”, IEEE Spectrum, vol. 31, pp. 27, 1994.

[10] M. Gopikrishnan , Dr. T.Santhanam “A tradeoff between template size reduction and computational accuracy in Iris Patterns Recognition using Neural Networks”, seec proceedings by , SEEC’ 2010..

[11] M. Gopikrishnan , Dr. T.Santhanam “Improved Biometric Recognition and identification of Human Iris Patterns using Neural Networks”, pp 610 – 615 by ICMCS 2010.

[12] M. Gopikrishnan , Dr. T.Santhanam and Dr.R.Raghavan, “Neural Network based accurate Biometric Recognition and identification of Human Iris Patterns” seec proceedings , pp154 – 157 , 2009.

[13] CASIA Iris Image Database, http:// www.sinobiometrics.com.

[14] Feedforward Network, Backpropagation, Neural Network Toolbox in MATLAB.

[15] Cascade-forward Network, Back propagation, Neural Network Toolbox in MATLAB.

[16] Neural Network User Guide, available from http:// www.mathworks.com/ help/ toolbox /nnet/2010