A Hybrid Intelligent System for Arabic Handwritten Number Recognition

Reda M. Hussein
Faculty of computers & Information, Shiben El-Kom, Menoufia University, Egypt.
Reda3032@yahoo.com

W. F. Abd El-Wahed
Operations Research Dept. Faculty of computers & Information, Shiben El-Kom, Menoufia University, Egypt.

Fawzy Torkey *Operations*
Prof. of Computer Science & Engineering and President of Kafer El-Sheekh University, Egypt
Fatorkey@yahoo.com

Abstract

This paper shows how developments in the area of neural network combined with genetic algorithms can be used in the handwritten digit recognition. In this work, two approaches to the design of a feed-forward neural network that model the handwritten recognition system are discussed. The first approach focuses on constructing the network by using a trail-and-error method the second approach is responsible for determining the appreciate parameters of the neural network and its learning algorithm by the mean of genetic algorithms. Results show that using genetic algorithm for selecting the near optimal parameters of the neural network, is improving classification performance on handwritten digits.

Keywords*: Neural Networks; Genetic Algorithms; Handwritten numeral recognition*

1. Introduction

This work deals with neural network application in the field of pattern recognition and more specifically in handwritten recognition system. The application of NN in modeling non-linear systems, has a central drawback; the leak of a precise method to choose the appreciate topology, type of activation function, and parameters of learning algorithm. These tasks are usually based on a trail-and-error procedure performed by the developer of the model. In this approach, optimality or even near-optimality is not guaranteed, because the explored search space of the NN parameters is just a small portion of the whole search space and the type of search is rather random. To overcome this drawback, an automated method, based on the evolutionary properties of genetic algorithms (GAs), is developed. The role of GAs is to evolve several network architectures with different parameters so that the best possible combination is finally chosen.

Handwritten character recognition has been one of the most challenging tasks for artificial neural networks (ANN) designers. A number of researchers have recently applied neural network techniques to recognize the hand-written characters by using a genetic algorithm (GAs) approach. In [1] GA is used for optimally design the network architecture including number of hidden layers, number of neurons in each layer, connectivity and activation functions. And in [2] Except for the network architecture, the types of activation functions of the hidden and output nodes, as well as the type of the minimization approach of the back-propagation algorithm, are also included in the GA encoding. Also In [3] a genetic algorithm is used to obtain the optimal activation functions that vary according to some intermediate signals of the neural network. In [4], evolutionary programming was used for training neural networks, and in [5] a genetic algorithm was used for weight selection. Also In [6], changes in a neural network structure during training and operation were implemented by removing inactive synapses and inactive units. In [7] it is presented the evolution of neural networks for topology selection and weights through mutation although not directed to digit recognition.

In this paper improvements on the handwritten digit recognition accuracy are gained by a genetic selection of the parameters of the back-propagation learning algorithm (learning rate and momentum for all layers) in addition to connection weights.

The paper is organized as follow: next section presents the neural network architecture and its learning in with back-propagation algorithm. Section 3 introduces the Genetic Algorithms; Section 4 describes The Hybrid intelligent system used

for training the handwritten recognition system. Section 5 describes the dataset used for the handwritten digit recognition system. Section 6 presents a comparison to the two approaches and finally section 7 concludes the research study.

2. Neural Network

Consider a multi-layer feed-forward neural network with as shown in figure 1 with the following notation:

y_k : output of kth neuron of hidden layer

z_j : output of jth neuron of hidden layer

x_i : output of ith input to the neural network.

w_{kj} : weight between kth neuron and jth hidden neuron.

w_{ji} : weight between jth neuron and ith input.

net_k : output of linear combiner in kth neuron.

f_k , f'_k : activation function and its derivative of kth neuron

net_j : output of linear combiner in jth hidden neuron..

f_j , f'_j : activation function and its derivative of jth hidden neuron

d_k : desired output of kth neuron.

Each layer has units with sigmoid activation function that compute its output according to the following formula:

$$f(net) = \frac{1}{1+e^{-net}} \quad \text{..... (1)}$$

where, net is the weighted sum of the unit inputs plus a bias or offset term q .

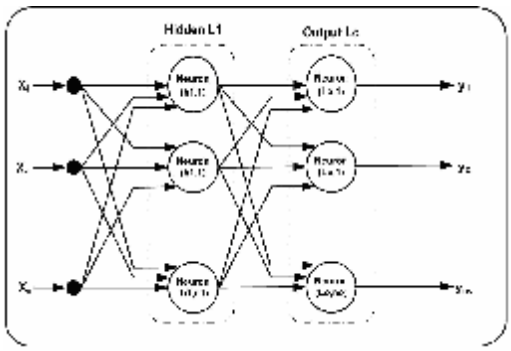

Figure 1:Multi-Layer feed forward Neural Network

The output of each neuron in hidden and output layer can be calculated as follow:

$$net_j = \sum w_{ji} x_i + q_j \quad \text{..... (2)}$$

$$z_j = f_j(net_j) \quad \text{..... (3)}$$

$$net_k = \sum w_{kj} x_j + q_k \quad \text{..... (4)}$$

$$y_k = f_k(net_k) \quad \text{..... (5)}$$

Once the network has been structured for a specific application, it is ready to be trained. Training a network means adapting its connection weights so that the network exhibits the desired computational behavior for all input patterns. Back-propagation known also as Error Back-Propagation or Generalized Delta Rule is the most widely used supervised training algorithm for neural networks. Connection weights of the neural network can be adapted by BP algorithm as follow:

Let the energy function chosen to be minimized is

$$E^p = \frac{1}{2}\sum_{k=1}^{m}(d_k^p - y_k^p)^2 \quad \text{..... (6)}$$

Update of output-layer weights

$$s_k^p = (d_k^p - y_k^p) f_k'(net_k^p) \quad \text{..... (8)}$$

$$\Delta w_{kj}^p(t+1) = h s_k^p z_j^p + a\Delta w_{kj}(t) \quad \text{..... (7)}$$

Updates of Hidden-Layer Weights

$$s_j^p = f_k'(net_k^p) \cdot \sum_{k=0}^{m} s_k w_{kj} \quad \text{..... (8)}$$

$$w_{ji}^p(t+1) = h s_j^p x_i^p + a\Delta w_{ji}(t) \quad \text{..... (7)}$$

where, g is the learning rate, which accelerates the learning procedure. Large values of g can cause oscillation, to avoid oscillation at large g, momentum term a is added to make the change in weights dependent on the past weight change.

The problem with ANN trained by back-propagation algorithm is that a number of parameters have to be set before any training can begin.

3. Genetic Algorithms

A GA is a mathematical search technique based on the principles of natural selection and genetic recombination [1]. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (i.e., minimizes the cost function). Some of the basic terminologies used in the field of genetic algorithms are:

- **Genotype** represents a potential solution to a problem, and is basically the string of values chosen by the user, also called chromosome.
- **Phenotype** is the meaning of a particular chromosome, defined externally by the user.
- **Chromosome** is a data structure that holds a "string" of task parameters, or genes. This string may be encoded as a binary bit-string or as an array of integers (floating point or real-coded representation).
- **Gene** is a subsection of a chromosome that usually encodes the value of a single parameter.
- The **fitness** of an individual is a value that reflects its performance (i.e. how well solves a certain task)
- **Recombination** or crossover produces new individuals in combining the information contained in the parents.
- **Mutation** occasionally injects a random alteration for one of the genes.

The process involved in GA optimization problems can be summarized as follows see figure 2:

1. Randomly generate an initial population of potential solutions.
2. Evaluate the suitability or 'fitness' of each solution.
3. Select two solutions biased in favor of fitness.
4. Crossover the solutions at a random point on the string to produce two new solutions.
5. Mutate the new solutions based on a mutation probability.
6. Goto 2.

Figure 2: Flowchart for Genetic Algorithm

4. The Hybrid Intelligent System

In this work a standard genetic algorithm is used [9, 15, 16, and 17]. The (learning rate and momentum for hidden and output layers) in addition to connection weights are the variable parameters of GA chromosome. The six steps followed by the genetic algorithm are as follows:

1- The initial population: This population includes 10 chromosomes that are obtained by randomly assigning parameters to each member.

2- Encoding: The real chromosomes of genomes representing the parameters

3- Training and testing. Each network of the population is trained by back-propagation for 500 iterations with training patterns.

4- The fitness. The fitness value for a given network is the Mean Square Error as in equation (6). The fitness value of the population is the average fitness value over all the members of the population.

5- The evolution stage. Three operators are applied over the population: selection, crossover and mutation operators. The operators are applied to the chromosomes of the input population, to produce the evolved new population.

6- The stopping criterion is satisfied when either the maximum number of generations is achieved

5. Training and Simulation

5.1. Data Description

Training and testing was carried out using the "Optical Recognition of Handwriting Digits" database, which is made available by E. Alpaydin, and C. Kaynak. The database can be downloaded at [18]. It consists of 3823 training pattern of Arabic digits. Patterns are 32x32 bitmap images see figure 3 for the digit 2, which are converted to a 4x4 block size to reduce the dimensionality of the inputs to the neural network to 8x8 bitmap images.

Figure (2) shows an example of converting 32x32 bitmap to 8x8 block bitmap. The conversion is achieved by taking the 32x32 binary images as inputs and groups neighboring pixels in blocks.

The sum of "ON" pixels in each block is used to create a pixel in resulting image. The resulting image is then converted to a vector of input values and fed to the neural network as a single input pattern.

Output patterns are divided to ten classes from 0 to 9 as shown in Figure (3). The testing sets, different from all training sets, are composed of 1797 patterns.

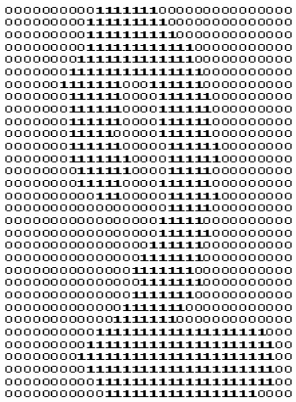

Figure 3: 32x32 bitmap of digit 2

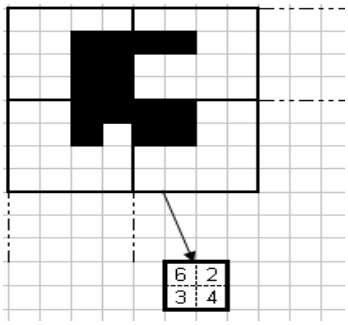

Figure 4: 32x32 to 8x8 image conversion using 4x4 block size.

5.2. The Model

Neural network used has 64 inputs, one hidden layer containing 15 neuron and output layer of 10 neurons. In addition to, two pairs of learning rate and momentum for each layer. All parameters of the network including weights were randomly initialized.

6. Experimental Results

In order to quantify the recognition capability of neural network configured with genetic algorithm, the analysis was performed in two stages as follow :

6.1. **Trail-and-Error Training** In training each problem using BP, learning rates and momentums of each layer could have been manipulated to find the best network configuration. The different combinations of the learning rates and momentums are used to try to find the right combination that will allow the solution to escape local minima but not skip over the global solution. The epoch is defined as one complete pass through the data set. For the handwritten number recognition system, the learning rate was set at 1.0 and the momentum factor set to 1.0. for each layer Both of these parameters were reduced with a reduction factor of 0.1 for every 500 epochs. This procedure was repeated twice with different initial weights. The total number of obtained network was 2x10x10=200 networks. This was done in order to keep the solution from oscillating and therefore helping to converge upon a solution. The best network of all 200 networks was then chosen and tested.

6.2. **Training with Genetic Algorithm** The model that described in section 5 was fed into the GA process. The basic parameters of GA that must be explored are the population size, the probability of crossover p_c , the probability of mutation

p_m and the number of generations. To determine the best possible values of these parameters, a number of experiments were carried out. The type of crossover used was the most common one, the one-point crossover [21]. The best values of p_c and p_m obtained are 0.9 and 0.01. the runs of the GA process were made with a population size of 10 networks trained with BP for 500 epoch for 20 generation (i.e. 200 network). The best network was then tested.

6.3. **Results** The performances of the two approaches discussed later were compared using two measures the MSE and the percentage of correction.
For the Trail-and-Error approach, figure 5 illustrates that the final MSE has a large variation over all networks. This means that the optimal is not guaranteed.

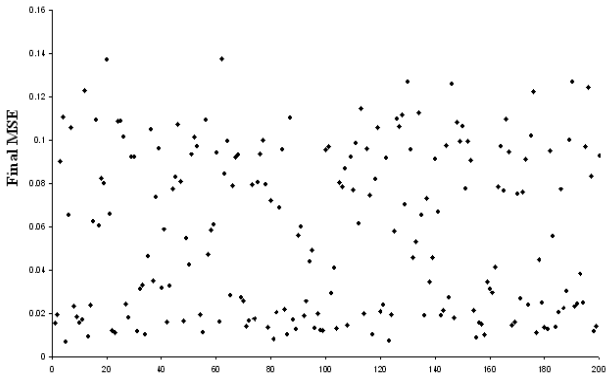

Figure 5: Best MSE over 200 run using Trail-and-Error

By selecting the best network obtained from this approach, and applying the training data as a test set, the achieved MSE is varying from 0.0013 to 0.0076 with average MSE 0.0043 as shown in figure 6. And the percentage of correction is varying from 96.3% to 100% with average value of 98.69%. the confusion matrix of the desired output versus the actual output is shown in figure 7

Perf.	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
MSE	0.00303	0.00683	0.0013	0.0041	0.0043	0.00398	0.0043	0.0016	0.0076	0.0066
% Correct	98.6702	97.9434	99.474	98.9717	99.742	99.2021	98.939	100	96.316	97.644

Figure 6: MSE and Percentage correction of training data trained with BP

Output / Desired	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
O(0)	371	3	1	0	0	0	1	0	0	1
O(1)	0	381	0	0	0	0	1	0	5	1
O(2)	0	0	378	0	0	1	1	0	0	0
O(3)	0	0	0	385	0	1	0	0	0	1
O(4)	3	0	0	0	386	0	0	0	3	1
O(5)	0	0	0	2	0	373	0	0	2	1
O(6)	2	1	1	0	1	0	373	0	1	0
O(7)	0	1	0	0	0	0	0	387	0	0
O(8)	0	1	0	0	0	0	1	0	366	4
O(9)	0	2	0	2	0	1	0	0	3	373

Figure 7: Confusion Matrix of training data trained with BP

For the test set, the achieved MSE is varying from 0. 0.0040 to 0.0344 with average MSE 0.016 as shown in figure 8. And the percentage of correction is varying from 77% to 98.3% with average value of 91.78%. the confusion matrix of the desired output versus the actual output is shown in figure 9

Perf.	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
MSE	0.0069	0.0225	0.0040	0.0167	0.0151	0.0175	0.0074	0.0128	0.0344	0.0228
% Correct	95.5056	90.1099	98.3051	90.7104	95.0276	95.0549	96.1326	89.944	77.011	90.000

Figure 8: MSE and Percentage correction of testing data trained with BP

Output / Desired	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
O(0)	170	2	0	0	3	2	1	0	0	0
O(1)	0	164	0	0	2	1	1	0	15	2
O(2)	0	1	174	1	0	2	0	0	0	0
O(3)	0	0	1	166	0	0	0	0	5	7
O(4)	8	1	0	0	172	0	4	1	1	2
O5)	0	0	0	1	0	173	0	13	10	4
O(6)	0	0	0	0	0	1	174	0	0	0
O(7)	0	0	1	0	1	0	0	161	0	0
O(8)	0	8	1	9	3	0	1	2	134	3
O(9)	0	6	0	6	0	3	0	2	9	162

Figure 9: Confusion Matrix of testing data trained with BP

For the GAs approach, as can be seen, figure 10shows the best MSEs found after each generation. The best solution was found after 15 generation. In figure 11, the corresponding average MSEs of the entire population after each generation are shown. One can see that the average performance of the population generally improves with the comparison of figure 5.

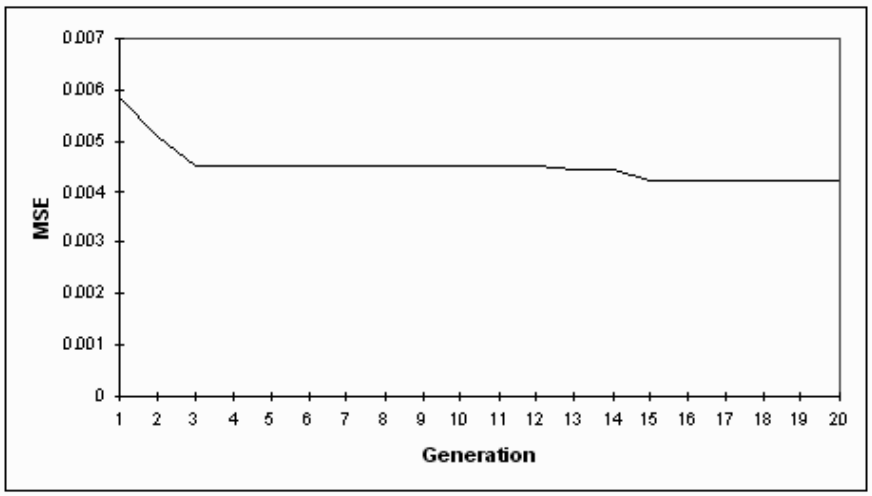

Figure 10:Best Fitness (MSE) versus Generation

The best network obtained from this approach, For the test set, the achieved MSE is varying from 0.0012 to 0.0055 with average MSE 0.0035 as shown in figure 12. And the percentage of correction is varying from 98.42% to 100% with average value of 99.5%. the confusion matrix of the desired output versus the actual output is shown in figure 13

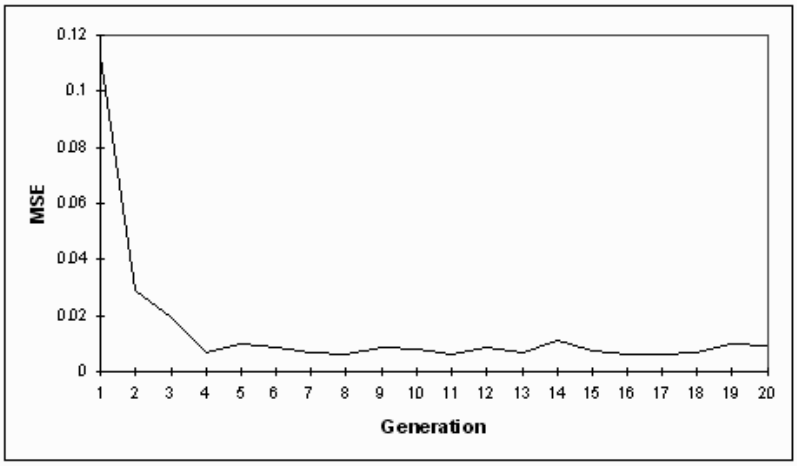

Figure 11:Average Fitness (MSE) versus Generation

For the test set, the achieved MSE is varying from 0.0031 to 0.0207 with average MSE 0.0104 as shown in figure 14. And the percentage of correction is varying from 89.01% to 98.87% with average value of 95.91%. The confusion matrix of the desired output versus the actual output is shown in figure 15

Perf.	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
MSE	0.00123	0.00465	0.003	0.00441	0.0035	0.00257	0.00299	0.002	0.0055	0.0051
Percent Correct	99.734	99.4859	99.737	98.9717	100	99.734	99.7347	100	98.421	99.215

Figure 12: MSE and Percentage correction of training data trained with GBP

Output / Desired	*O(0)*	*O(1)*	*O(2)*	*O(3)*	*O(4)*	*O(5)*	*O(6)*	*O(7)*	*O(8)*	*O(9)*
O(0)	375	0	0	0	0	0	0	0	2	0
O(1)	0	387	0	0	0	0	1	0	2	1
O(2)	0	0	379	0	0	0	0	0	0	0
O(3)	0	0	0	385	0	1	0	0	0	0
O(4)	0	0	0	0	387	0	0	0	0	1
O(5)	0	0	0	3	0	375	0	0	0	0
O(6)	1	0	1	0	0	0	376	0	2	0
O(7)	0	1	0	0	0	0	0	387	0	0
O(8)	0	0	0	0	0	0	0	0	374	1
O(9)	0	1	0	1	0	0	0	0	0	379

Figure 13: Confusion Matrix of training data trained with GBP

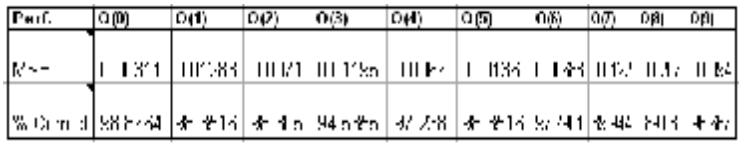

Figure 14: MSE and Percentage correction of testing data trained with GBP

Output / Desired	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
O(0)	176	0	0	0	0	0	0	0	0	0
O(1)	0	179	2	0	2	0	2	0	8	2
O(2)	0	0	174	4	0	0	0	0	0	0
O(3)	0	0	0	173	0	1	0	0	5	1
O(4)	0	1	0	0	176	0	1	1	0	0
O(5)	2	0	0	2	0	179	0	13	2	1
O(6)	0	1	0	0	0	0	177	0	0	0
O(7)	0	0	0	0	1	0	0	161	0	1
O(8)	0	1	1	2	1	0	1	1	155	1
O(9)	0	0	0	2	1	2	0	3	4	174

Figure 15: Confusion Matrix of testing data trained with GBP

7. Conclusions

The experimental results show that genetic algorithms have a strong potential to find good solutions for the neural network configuration problem, and therefore is a good alternative to select the most appropriate network for a given task. For the handwritten digit recognition problem, the average recognition accuracy of 95.2 % was found by the Trail-and-Error approach. The neural network was improved by selecting configuration parameters of the neural network using GA to 97.7%.

8. References:

[1] Manojit Dam, Deoki N. Saraf, Design of neural networks using genetic algorithm for on-line property estimation of crude fractionator products, Computers and Chemical Engineering vol. 30 pp. 722–729, 2006

[2] Konstantinos P. Ferentinos,Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms,Neural Networks vol. 18 pp. 934–950, 2005 Elsevier Ltd.

[3] S.H. Ling, H.K. Lam, and F.H.F. Leung, A variable-parameter neural network trained by improved genetic algorithm and its application,¨ in Proc. Int. Joint Conf. Neural Networks 2005 (IJCNN2005), Montreal, Canada, 31 Jul.-4 Aug. 2005, pp. 1343-1348.

[4] EN-HUI Zheng', Min Yang, Tuning of Neural Networks based on Genetic Algorithm and Statistical Learning Theory, Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 August 2004

[5] Montana DJ, "Neural Network Weight Selection Using Genetic Algorithms, " in Intelligent Hybrid Systems, Edited by Goonatilake S & Khebbal S, John Wiley & Sons, pp.85-104, 1995.

[6] Odri, S.V., Petrovacki, D.P., Krstonosic, G.A., "Evolutional Development of a Multilevel Neural Network, " Neural Networks, Vo1.6, pp.583-595, 1993.

[7] Braun, H., Weisbrod, J., 1993, Evolving Neural Networks for Application Oriented Problems, Proceedings of the Second Annual Conference on Evolutionary Programming, eds. D.B. Fogel and W. Atmar, La Jolla, CA: Evolutionary Programming Society, pp.62-71.

[8] Haykin S, "Multilayer Perceptrons, " in Neural . Networks: A Comprehensive Foundation, John Griffin (ed), IEEE Press, Macmilan College Pu CO., pp. 138-229, 1994.

[9] Goldberg, D., 1989, Genetic Algorithms in Search, Optimization and Machine Learning, Reading MA: Addison-Wesley.

[10] Koza, J., 1992, Genetic Programming, Cambridge, MA, MIT Press.

[11] kbeiro, J.L. and Treleaven P.C., 1994, Genetic-Algorithm Programming Environments, Computer, pp.28-43.

[12] Fogel, D.B., "Theoretical and Empirical Properties of Evolutionary Computation," in Evolutionary Computation: Toward a New Philosophy of Machine Iritelligence, IEEE Press, pp.121-186, 1995.

[13] Porto, V.W., Fogel D.B., Fogel L.J., "Alternative Neural Network Training Methods, " IEEE Expert, Vol 10, No.3, pp.16- 22, 1995.

[14] Kosko B. Neural Networks and Fuzzy Systems – A Dynamical Systems Approach to Machine Intelligence. Prentice Hall, 1992

[15] Davis, L. (ed), 1987, Genetic Algorithms and Simulated Annealing, Pitman, London.

[16] Fogel, D.B., "Theoretical and Empirical Properties of Evolutionary Computation", in Evolutionary Computation:

Toward a New Philosophy of Machine Iritelligence, IEEE Press, pp.121-186, 1995.
[17] Davis, L. (ed), 1991, Handbook of Genetic Algorithms, New York: Van Nostrand Reinhold.
[18] "Optical Recognition of Handwriting Digits" database ftp://ftp.ics.uci.edu/pub/machine-learning- tabases/optdigits/
[19] Mandana Ebadian Dehkordi, Nasser Sherkat, Tony Allen, "Handwriting style classification" Volume: 6, Issue: 1, August, 2003. pp. 55 - 74.
[20] Seok; Suen, Ching Y.," A class-modular feedforward neural network for handwriting recognition" Volume: 35, Issue: 1, January, 2002. pp. 229-244
[21] Goldberg, D. E. " Genetic Al Algorithms in search, optimization and machine learning." Reading, MA: Addison , 1989.