A Novel Evolutionary Algorithm For Block-Based Neural Network Training

Amin Niknam, Pourya Hoseini, Behbood Mashoufi, Abdollah Khoei

Microelectronics Research Laboratory
Urmia University
Urmia, Iran
st_a.niknam@urmia.ac.ir; st_p.hoseini@urmia.ac.ir; b.mashoufi@urmia.ac.ir; a.khoei@urmia.ac.ir

***Abstract*— A novel evolutionary algorithm with fixed genetic parameters rate have presented for block-based neural network (BbNN) training. This algorithm can be used in BbNN training which faces complicated problems such as simulation of equations, classification of signals, image processing and implementation of logic gates and so on. The fixed structure of our specific BbNN allows us to implement the trained network by a fixed circuit rather than utilizing a reconfigurable hardware which is usually employed in conventional designs. Avoiding the reconfigurable hardware leads to lower power consumption and chip area. All simulations are performed in MATLAB software.**

Keywords - Block-based neural network; Genetic algorithm; Evolutionary training.

I. INTRODUCTION

Artificial neural networks have been successfully applied to diverse engineering problems due to their model-free approximation capability to complex decision making processes [1]. Unlike in a multi-layer perceptron (MLP), in a BbNN the general structure can be altered by the training algorithm. In this paper BbNN is preferred to MLP because of its flexibility. One of the common algorithms used for training neural networks is back propagation that has problems, first the derivative of the activation function must be available, and second the mentioned algorithm is only useful in finding the optimum network weights and biases and cannot be used in training of the network's general structure. Therefore we have used genetic algorithm to train the BbNN that can find both of internal parameters such as weights and biases and the optimal general structure of the network while there is no need for the derivative of the activation functions. Wei.Jiang in [2] has also used a similar method but we have achieved a more accurate algorithm resulting in more optimal neural networks without using any reconfigurable hardware. Finally the XOR function is used to compare our algorithm results with Wei.Jiang's network in [5]. Our algorithm can also find trained BbNN for the most complicated functions. Implementation of the sinusoidal function have been tested to demonstrate the ability of our algorithm. This algorithm is used for BbNN training which can classify as couple of different ECG signals. For example, in this paper the BbNN used for classifying Sinus Tachycardia (ST) and Ventricular Tachycardia (VT) heart signals.

II. BLOCK-BASED NEURAL NETWORK

A BbNN can be represented by a two-dimensional array of blocks. Each block is a basic processing element that corresponds to a feed forward neural network with four variable input/output nodes. A block is connected to its four neighboring blocks with signal flow represented by an arrow between the blocks [4]. Fig. 1, illustrates the network structure of an $m \times n$BbNN. In this figure X1, …, Xn are network inputs and Y1, …, Yn are network outputs. The structure of BbNN is explained in [1]-[5] in detail.

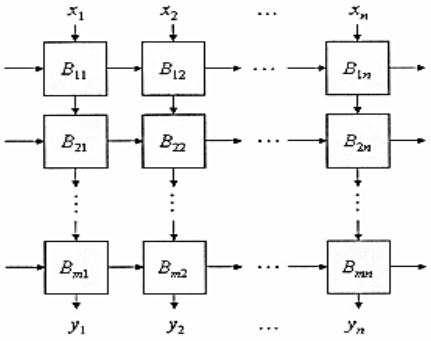

Fig. 1. An m×n BbNN

III. EVOLUTIONARY OPTIMIZATION OF BBNN

Common genetic algorithm has 4 major steps, namely, initialization, selection, reproduction, replacement. Algorithm parameters such as population size, tournament size, mutation rate, crossover rate and so on, are obtained by trial and error.

978-1-4673-6206-1/13/$31.00 ©2013 IEEE

A. Initialization

At this stage, program generates randomly an initial population of chromosomes. Chromosomes are the same thing as BbNN structures. Chromosomes formed in matrixes which show the direction of signal flows, and their internal structure of each block. Then fitness of each chromosome will be measured (how-to of this estimation has been explained in D section).

Fig. 2, shows the changing process in fitness function value for different population sizes.

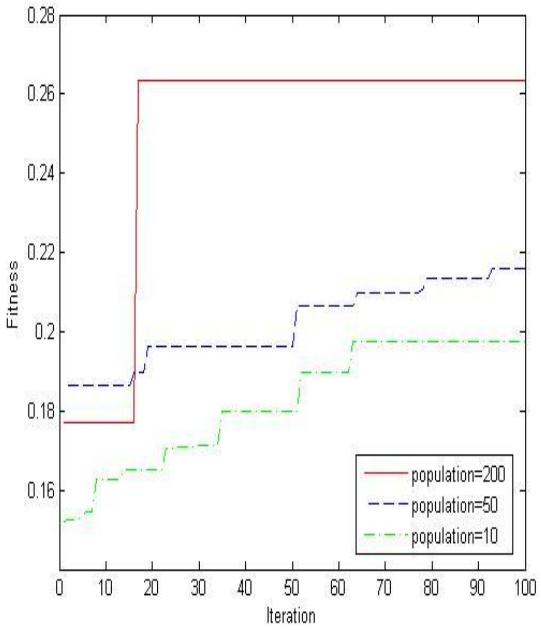

Fig. 2. Changing process in fitness function value for different populations

According to fig. 2, empirically a population size of 200 provides the best fitness in this algorithm.

B. Selection

At this stage, we have applied tournament selection method to randomly select chromosomes from the number of previous stage outputs; later there will be a winner chosen among those chromosomes which has the highest fitness. This process is repeated in number of the chromosomes that should be removed in each iteration.

Winner chromosomes resultant from application of tournament selection are called parents, which are expected to be in the same number as eliminated chromosomes in the replacement stage.

Fig. 3, shows the changing process in fitness function value for different tournament sizes.

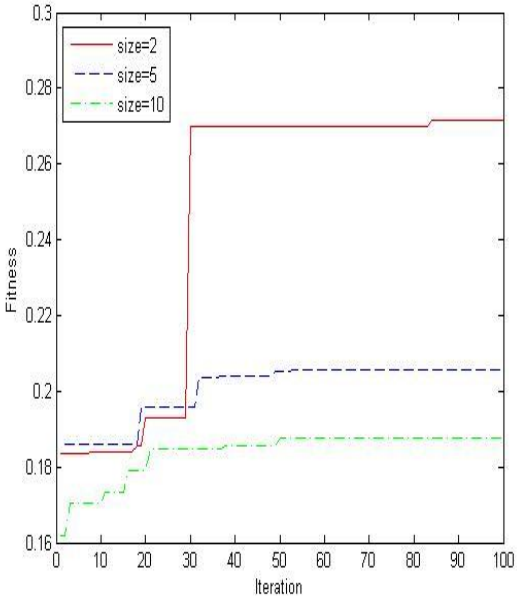

Fig. 3. Changing process in fitness function value for different tournament sizes.

As in fig. 3, is characterized, by increasing tournament size, the maximum fitness is decreased. So empirically a tournament size equal with 2 provides the best fitness in this algorithm.

C. Reproduction

In this stage the two operators of crossover and mutation which are known as the genetic operators are applied to the parents. Firstly the crossover operator is implanted, and then the mutation operator is implanted in series.

1) *crossover*

The crossover operator is only used in signal flow. Crossover operation can be done to different methods. In [2], Wei.Jiang has used random crossover where at first, select a pair of BbNN and then a group of signal flows corresponding in the two selected BbNN are exchanged randomly. But For the genetic algorithm of used in our algorithm we have used the uniform crossover which cause finding of the populations by better fitness. The performance of this method is thus in two parent neural network that are selected randomly, exchange their signal flows with 50 percent chance. This operator is applied on the parents with determining crossover rate value that will be asked in the beginning of the program. Fig. 4 and fig. 5, show changing process in fitness function value for uniform and random crossover respectively that by using of uniform crossover higher fitness value obtained.

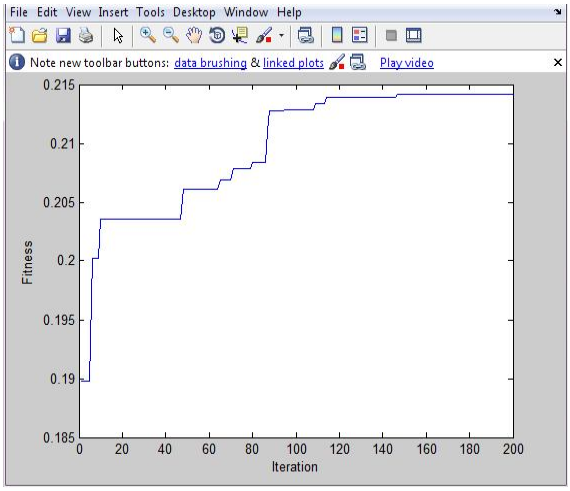

Fig 4. Changing process in fitness function value for uniform crossover.

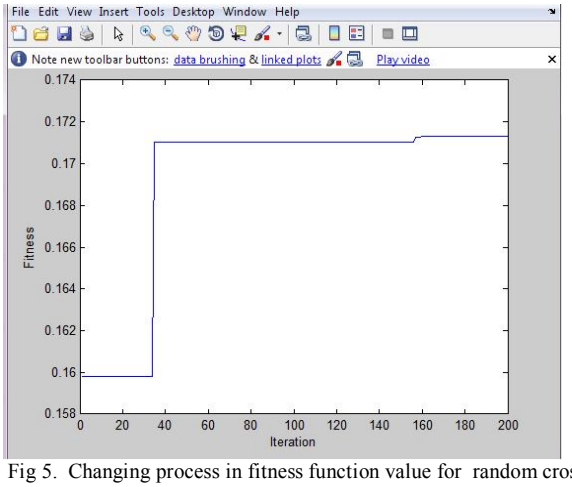

Fig 5. Changing process in fitness function value for random crossover.

Fig. 6, shows the changing process in fitness function values for different crossover rate.

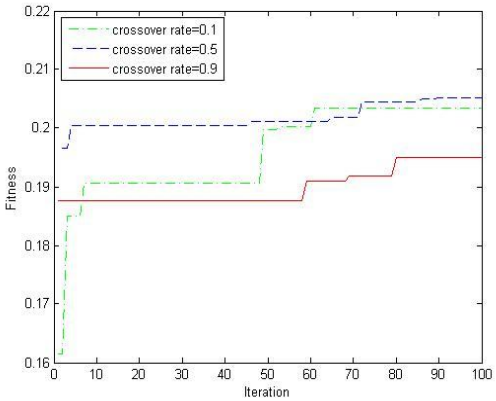

Fig 6. Changing process in fitness function value for different crossover rate.

According to fig. 6, empirically a crossover rate equal with 0.5 provides better fitness value in this algorithm.

2) mutation

This operator is applied on each outputs of crossover operator, by probability of mutation rate (the value requests from user in the beginning of program).

There are two types of mutation operator:

a) Structuralmutation

This operator applies on signal flows, thus at first selects one signal flow by probability of structural mutation rate value and then changes its direction. In this case, in the neighboring blocks of changed signal flow, generate the new weights and biases based on the random Gaussian distribution and the while not connected weights are removed [2].

Fig. 7, shows the changing process in fitness function value for different structural mutation rate.

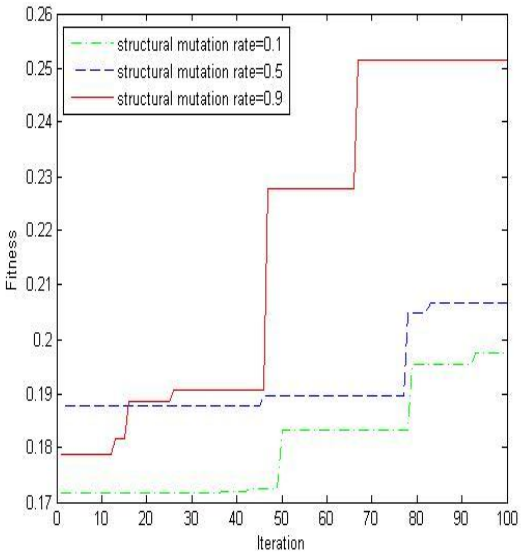

Fig. 7. Changing process in fitness function value for different structural mutation rate.

According to fig. 7, empirically a structural mutation rate equal with 0.9 provides better fitness value in this algorithm.

b) Weight-bias mutation

This operator selects one of network weights or biases by probability of weight-bias mutation rate and updates their values based on (1):

$$W^* = W + r \qquad (1)$$

where r denotes Gaussian random variable.

Fig. 8, shows the changing process in fitness function value for different weight-bias mutation rate.

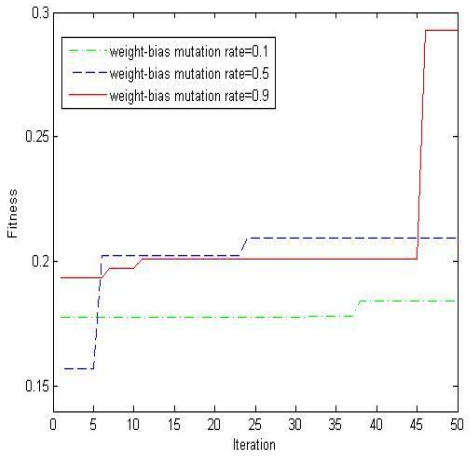

Fig 8. Changing process in fitness function value for different weight-bias mutation rate.

According to fig. 8, empirically a weight-bias mutation rate equal with 0.9 provides better fitness value in this algorithm.

D. replacement

In this stage, fitness values of offspring that were produced by genetic operators are calculated and then are placed at worst chromosomes. Genetic algorithm to number of iterations that is determined in the beginning of program is repeated. After final iteration the best founded chromosome (the chromosome with higher fitness value) is selected.

- fitness estimation

In this stage, the fitness of each chromosome is measured, thus at first all chromosomes apply to network input and their outputs are calculated (real output) that will compare with desired output (target output); the differences between them are regarded as error.

And finally the fitness of each population is measured by (2):

$$f = \frac{1}{1 + \sum e^2} \qquad (2)$$

Where e is difference between real output and target output.

IV. COMPARISON OF ALGORITHM RESULTS

In reference [5], Wei.Jiang has used the XOR function to test his training algorithm. Its final trained BbNN reached to fitness value equal with 0.95 that has 2 rows and 3 column as shown in fig. 9, but in this paper with our algorithm, trained BbNN has less blocks (2 rows and 2 column) and while the BbNN has fitness value equal with 1. This BbNN shown in fig.10. The changing process in fitness function value shown in fig.11.

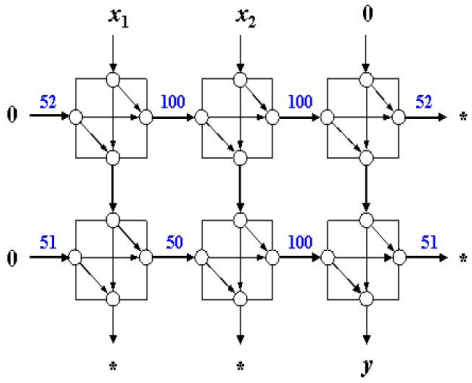

Fig 9. The evolved BbNN for XOR classification by W.Jiang algorithm

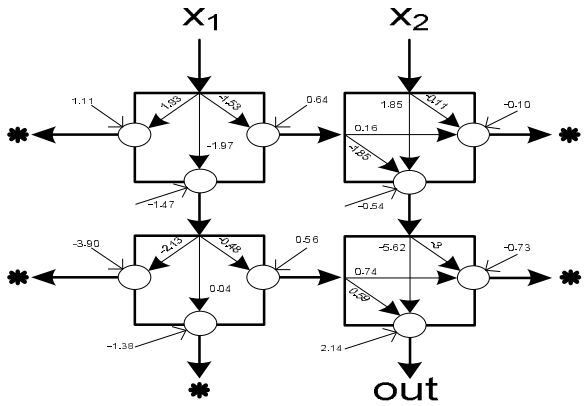

Fig 10. The evolved BbNN for XOR classification by wroted algorithm in this paper

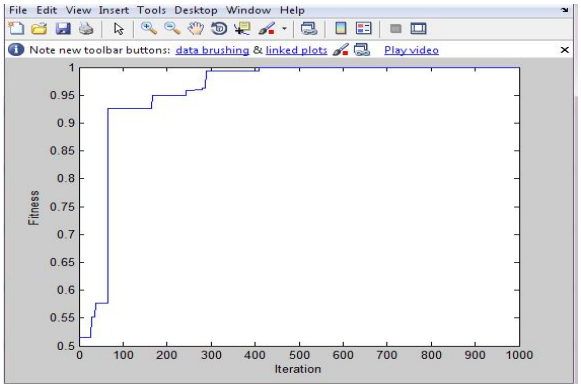

Fig 11. Changing process in fitness function value for training of XOR BbNN

In this paper we tested our algorithm for sinusoidal function problem that is more complicated than XOR problem.

We used 20 samples from sinusoidal function in the interval 0 to π as training patterns and considering the genetic algorithm parameters as table. I. We achieved to trained BbNN for sinusoidal function as shown in fig. 12. Changing process in fitness function value illustrated in fig. 13. Fig. 14, shows comparing between sinusoidal function and network output.

TABLE I. CONSIDERED ALGORITHM PARAMETERS FOR SINUSOIDAL FUNCTION PROBLEM.

Algorithm parameters	Best values
Network row	2
Iteration number	1000
Generation number	200
Tournament size	2
Deletion number	8
Crossover rate	0.5
Structural mutation rate	0.9
Weight-bias mutation rate	0.9
Activation function	sigmoid

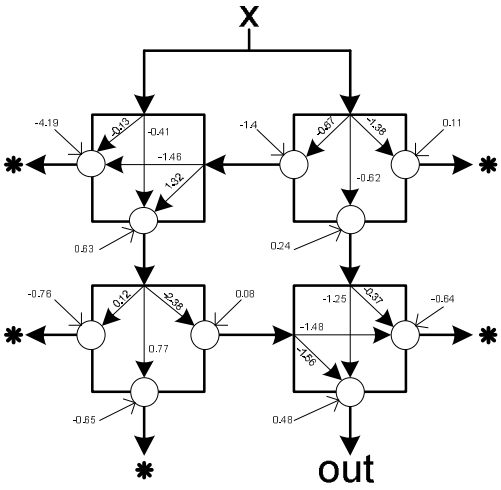

Fig 12. Sinusoidal function BbNN

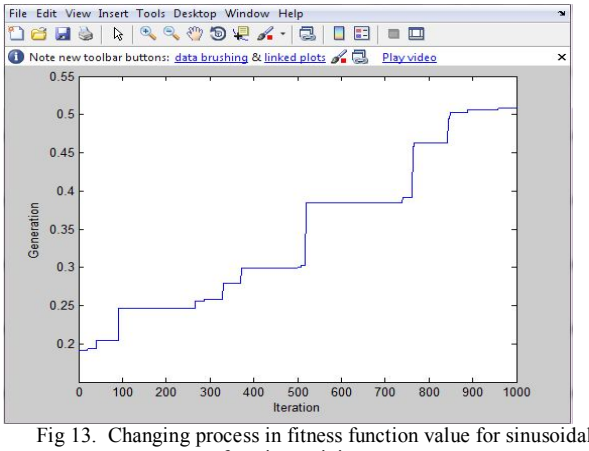

Fig 13. Changing process in fitness function value for sinusoidal function training.

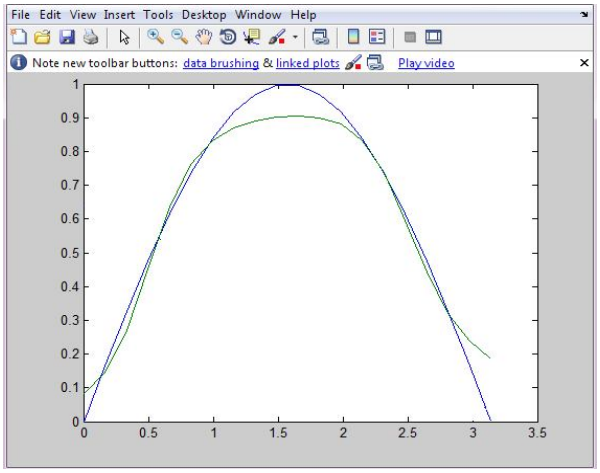

Fig 14. Comparing between sinusoidal function and network output

- Using algorithm for ECG signals classification

As mentioned earlier, in this paper the genetic algorithm is used to detect two signals ST and VT. Fig. 15, shows these two signals [6].

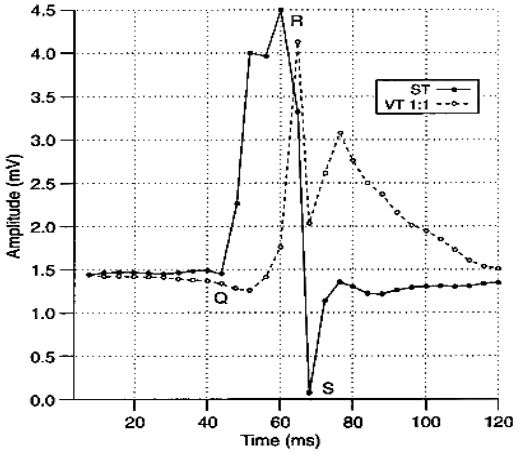

Fig 15. ST and VT signals

For training of network we used 20 different ST signals and 20 different VT signals available at www.ecgdatabase.com that these samples are cardiac signals for different people.

For BbNN training, at first extract samples from training signals by using sample and hold circuit, these samples are forty sets in form of 1×10 matrixes. 20 sets related to ST signals and 20 sets related to VT signals, and then use these samples for training of BbNN by wrote algorithm.

We got to trained BbNN as shown in fig. 16. It achieves fitness value equal with 1 after about 100 iterations. In this figure, unimportant outputs are shown with stars and S1-S10 are outputs of sample and hold circuit.

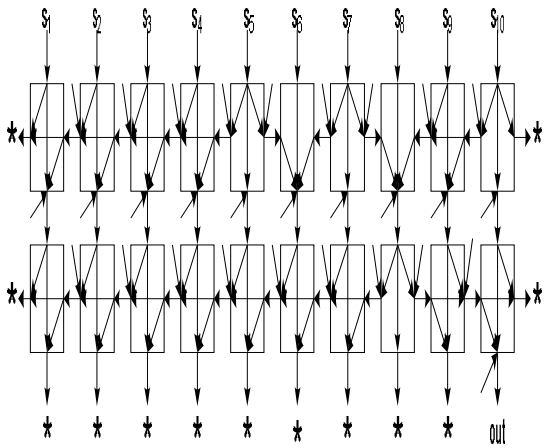

Fig 16. Trained BbNN for ST and VT classification (unimportant outputs marked with a star).

V. Conclusion

In this paper a novel evolutionary algorithm has been explained. The simulation results show that this algorithm can be used for training of BbNN for solving many problems such as, signal processing, image processing, simulation of logic gates and complicated functions and so on. Output BbNNcan be implemented by circuit design with no need to use reconfigurable hardware.

Acknowledgment

The authors have special thanks to Faculty of Microelectronics for financial supports.

References

[1] Sang-Woo Moon and Seong-Gon Kong, "Block-Based Neural Networks", IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 12, NO. 2, MARCH 2001.

[2] Wei Jiang, and Seong G. Kong, "Block-Based Neural Networks for Personalized ECG Signal Classification" IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 18, NO. 6, NOVEMBER 2007.

[3] Wei Jiang, Seong G. Kong, and Gregory D. Peterson, "Continuous Heartbeat Monitoring Using Evolvable Block-based Neural Networks" 2006 International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006.

[4] Wei Jiang, Seong G. Kong, and Gregory D. Peterson, "ECG Signal Classification using Block-based Neural Networks", Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4, 2005.

[5] Wei Jiang, "PERSONALIZED HEALTH MONITORING USING EVOLVABLE BLOCK-BASED NEURAL NETWORKS", Ph.D thesis, The University of Tennessee, Knoxville, Aug 2007.

[6] Richard Coggins, Marwan labri, Barry Flower, Stephen Pickard, "A Low-Power Network for On-Line Diagnosis of Heart Patients",vol 15, 1995 IEEE.