Improved Selection Method for Evolutionary Artificial Neural Network Design

G.V.R. Sagar[1], Bhallamudi Ravi Krishna[2], B. V. Chowdary[3], G. Y. Sagar[4], N. Phani Kumar[5]
[1]Associate Professor, Department of Electronics and Communication Engineering, G. Pulla Reddy Engineering College (Autonomous), Kurnool, A.P., India
[2]Associate Professor, Department of Artificial Intelligence and Data Science, Vignan Institute of Technology and Science, Hyderabad, T.S., India.
[3]Associate Professor, Department of Information Technology, Vignan Institute of Technology and Science, Hyderabad, T.S., India
[5]Department of Basic Sciences & Humanities, Vignan Institute of Technology and Science, Hyderabad, T.S., India
[4]Professor, Department of Statistics, College of Natural and Computational Sciences, Gambella University, Ethiopia.

Corresponding Author:
gysagar@gmail.com, gysagar@gmu.edu.et

Abstract— this paper improves the role of adaptive nature of new Evolutionary Algorithm (EA) [19] in designing Artificial Neural Network (ANN) using the proper selection mechanism. The proposed EA has been used for two purposes. One is generalization of architecture. In this, the optimal adaptive architecture is achieved by using evolutionary crossover and mutation. The adaptive strategy increased in the stage of selection process. This algorithm used the tournament selection method with minimum hamming distance. Unlike most previous studies, proposed EA puts emphasis on autonomous functioning in the design process of ANNs. The mathematical frame work is discussed in [19]. The proposed EA has been tested on a number of benchmark problems in machine learning and ANNs, including breast cancer, diabetes, heart problems and for time complexity N-Bit Parity is used. The experimental results show that proposed EA can design compact ANN architectures with good generalization ability, compared to other algorithms with good time complexity.

1 INTRODUCTION

ARTIFICIAL neural networks (ANNs) have been used widely in applications like system identification, signal processing, classification, and pattern recognition. Most applications were developed using feed-forward ANNs and the back-propagation (BP) learning algorithm [1] [17]. The important issue in using ANNs is to choose their architectures appropriately. Whereas a too large architecture may overfit the training data a too small architecture may under fit the training data. Both overfitting and under fitting cause bad generalizations of ANNs. So, it is necessary to design ANNs automatically and they can solve different problems efficiently. There are many algorithms available for designing ANNs automatically, such as constructive, pruning, constructive–pruning, and regularization algorithms [2]–[4]. A constructive algorithm [16] adds hidden layers, neurons, and connections to a minimal ANN architecture. A pruning algorithm deletes unnecessary hidden layers, neurons, and connections from an oversized ANN. A constructive–pruning algorithm is a hybrid approach. In addition, evolutionary approaches, such as genetic algorithms [5], evolutionary programming [6], [7], and evolution strategies [8], have been used extensively in designing ANNs automatically.

The main problem in designing ANNs using constructive, pruning, constructive–pruning, and regularization algorithms is that they use a predefined, fixed, and greedy strategy. Thus, these algorithms are susceptible to becoming trapped at architectural local optima [9]. The proposed work presents a new Evolutionary algorithm (EA) that uses one-point crossover and adaptive merging and adding process at the mutation level in designing ANNs given in [19]. The selection is done by minimum hamming distance in the tournament selection to calculate the error mean square error in training sets.

The main problem in designing ANNs using constructive, pruning, constructive–pruning, and regularization algorithms is that they use a predefined, fixed, and greedy strategy. Thus, these algorithms are susceptible to becoming trapped at architectural local optima [9]. The proposed work presents a new Evolutionary algorithm (EA) that uses one-point crossover and adaptive merging and adding process at the mutation level in designing ANNs given in [19]. The selection is done by minimum hamming distance in the tournament selection to calculate the error mean square error in training sets.

I. EVOLUTIONARY ALGORITHMS (EAS)

Evolutionary Algorithms (EAs) refer to a class of population based stochastic search algorithms developed [18] from ideas and principle of natural evolution. They are a class of stochastic optimization algorithms inspired by biological process that allows populations of neurons to adapt genetic inheritance and survival of the fittest. The architecture design is crucial in ANN because it has significant impact on network information processing capabilities. For a given learning task, an ANN with only a few connections and neurons may not be able to perform the task at all, due to its limited capability, while an ANN with a large number of nonlinear neurons and connections may overfit noise in the training data and fail to have good generalization ability.

The main advantage of this approach is that it can avoid the architectural local optima problem [9], [10]. However, the evolutionary approach is quite demanding in both time and user-defined parameters [2]. Moreover, it is necessary to find a set of optimal control parameters so that an evolutionary process can balance exploration and exploitation in finding good quality solutions. This can be overcome, by using proposed evolutionary learning process applied on feed-forward ANN architecture.

II. EVOLUTIONARY ANN MODEL

Designing architecture is an important issue in the evolution of an ANN. The proposed evolutionary algorithm used the one point or cutting point crossover to improve the behaviour between parents and off-springs was given in [19]. The proposed EA used an improved adaptive search strategy in designing ANN. The improved adaptive strategy is evolutionary merging, adding and selection of neurons based on the learning ability of hidden neurons or layers in ANN.

A population P of size m {1, 2,…., m} is considered and elementary steps of initial selection, partition and recombination are applied on P. The first elementary step is the initial selection is applied on P and new population P^l is generated.

$$P = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} \text{ with}$$

$$x_i \in \Omega$$

1.1

The fitness evaluation for individuals of P is

$$\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_m \end{bmatrix} \begin{matrix} \to f(x_1) \\ \to f(x_2) \\ . \\ . \\ . \\ \to f(x_{1m}) \end{matrix}$$

1.2

The 'selection' of individuals in P^l is based on probability rule $\frac{f(x_j)}{\Sigma_{l=1}^{m} f(x_i)}$ where f is the fitness function given as f = 1/mean square error.

$$P^1 = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_m \end{pmatrix}$$

1.3

The individuals having small fitness values are not allowed into P^1 at all. This is to institute the natural survival of the fittest principle.

A. One-point crossover Transform

The second elementary step is recombination or crossover. For one point crossover, let L_i and R_i are the crossover operations on two subspaces Q_j and Q_{j+1} with n layered architectures. The family of crossover transformations F is given as

$$\mathbf{F = \{L_i | 0 \le i \le n, L_i = L_{\{1,2,....i\}}\} \cup \{R_i | 0 \le i \le n, L_i = L_{\{1,2,....i\}}\}}$$

1.4

For instance, if a = (a_1, a_2, …, a_n), b=(b_1, b_2, ….,b_n), $\mathbf{a \in Q_j}$ **and** $\mathbf{b \in Q_{j+1}}$ then, each element of a and b are represents
the matrices of the layers in the two parent architectures. Each layer has m number of neurons with weight matrix W. The probability distribution of family F is

$$L_i(a,b) = (a_1, a_2, ...a_i, b_{i+1}, b_{i+2}, ...b_n)$$

1.5

$$R_i(a,b) = (b_1, b_2, ...b_n, a_{i+1}, b_{i+2}, ...a_n) \quad 1.6$$

The above process is called as one-point crossover and the new off-spring population is P^{11}. On completion of a cycle, the procedure starts all over again, with the initial population p^m and the cycle are repeated a number of times depending on the problem.

$$P^{11} = \begin{pmatrix} z_1 \\ z_2 \\ . \\ . \\ . \\ z_m \end{pmatrix}$$

1.7

B. Mutation

Finally, mutation is the process with small probabilities that replace z_i with $F(z_i)$ according to the evolutionary process given in the this section, for chosen F∈ M. This creates a new population P^{111}.

$$P^{111} = \begin{pmatrix} w_1 \\ w_2 \\ . \\ . \\ . \\ w_m \end{pmatrix}$$

1.8

Neuron Merging: Created off-springs carried from crossover consist of M number hidden neurons in the hidden layer. Where M is selected by the user. All connection weights remain same. An epoch counter (μ_i) is initialized with zero, this count is used to count the number of epochs of hidden neurons and is trained. Mean Square Error (E) value is computed on the training set. If the termination criterion is satisfied, the training process is stopped and final network architecture gives the optimized ANN. Each hidden neuron is labelled with significance η_i. The significance of each hidden neuron

h_i [11] is computed. Train the ANN on the training set for a number of epochs (μ). Increment the epoch count as follows for

$i = 1,2\ldots..,N, \quad \mu_i = \mu_i + \square$ 1.9

Where N is the number of hidden neurons in the existing architecture. Initially N and M are the same.

The neuron merging in EA is based on significance $\boldsymbol{\eta_i}$. It is computed using equation (2.0) for the hidden neuron h_i.

$$\boldsymbol{\eta_i = \frac{\sigma_i}{\sqrt[3]{\mu_i}}}$$ 2.0

Where $\boldsymbol{\sigma_i}$ is the standard deviation.

Significance $\boldsymbol{\eta_i}$ is small when its standard deviation $\boldsymbol{\sigma_i}$ and/or its number of training epochs μ_i is large. The smaller the value of $\boldsymbol{\eta_i}$, the less significant h_i is. A less significant hidden neuron delivers constant information to the neurons of output layer. The next process of merging is to compute the correlations between s-labeled hidden neuron and other neurons in ANN. The EA uses the Pearson product-moment correlation coefficient to measure the correlation and it is denoted by C_{ij}. It is the correlation between s-labeled hidden neuron i and the unlabeled hidden neuron j and is given as

$$\boldsymbol{C_{ij} = \frac{\sum_{p=1}^{P}(h_i(p)-\bar{h}_i)\,(h_j(p)-\bar{h}_j)}{\sigma_i\sigma_j}}$$ 2.1

Where $h_i(p)$ and $h_j(p)$ are the outputs of hidden neurons i and j for the example p in the training set, the variables $\boldsymbol{\bar{h}_i}$ **and** $\boldsymbol{\bar{h}_j}$ are the mean values and $\boldsymbol{\sigma_i}$ **and** $\boldsymbol{\sigma_j}$ are standard deviation of h_i and h_j, respectively.

The EA merges two correlated neurons, for instant h_a and h_b, produces a neuron h_m. The algorithm assigns the input and output connection weights of the h_m as follows

$$\boldsymbol{w_{mi} = w_{ai} + w_{bi} \qquad i = 1, 2, \ldots..p} \qquad 2.2$$
$$\boldsymbol{w_{jm} = w_{ja} + w_{jb} \qquad j = 1, 2, \ldots..q}$$

Where p and q are the number of neurons in the predecessor and successor layers of the ANN. The neuron merging is carried out based on, whose contribution is negligible with respect to the overall network output and the decision is based on the selection criterion given in the above equation 2.1.

Neuron Addition: The EA uses a simple criterion to add a hidden neuron in an ANN. This is based on the training error progress of the ANN. When the error of the ANN does not reduce by an amount $\boldsymbol{\epsilon}$ after training epochs $\boldsymbol{\tau}$, EA assumes that, it is necessary to add hidden neurons in the ANN. Here $\boldsymbol{\varepsilon}$ and $\boldsymbol{\tau}$ are two user specified parameters. The neuron addition criterion is given as

$$\boldsymbol{E(t) - E(t+\tau) \le \varepsilon \qquad t = \tau, 2\tau, 3\tau \ldots\ldots..}$$ 2.3

Where $\boldsymbol{E(t)}$ **and** $\boldsymbol{E(t+\tau)}$ are the training errors at epochs t and$\boldsymbol{(t+\tau)}$, respectively. EA adds a hidden neuron by splitting an existing hidden neuron of an ANN. Two neurons are created by splitting and have the same number of connections as off-spring neuron.

C. Tournament Selection

Finally the fittest ANNs is trained by the training sets based on the tournament selection procedure which improves the adaptability and fitness of the ANN. In this method, p individuals are selected as a group and arrange 'r' number of groups from the pool of 2M individuals (both parent and off-spring population). Two individuals are selected from each group and arrange 'r' number of tournament; the best one is selected based on Hamming distance method. Let n be the number of unique fitness values and $f_1 < \ldots\ldots. < f_{n-1}$ $(n \le N)$, where N is the number of independent trails, the lower fitness f_1 is denoting the worst fitness occurring in the population and f_n denoting the best fitness in the population. The pseudo code for this selection process is given in Fig (1) The cumulative fitness distribution $S(f_i)$ of individual with fitness value f_i is calculated as

$$\boldsymbol{S(f_i) = \begin{cases} 0 & i < 1 \\ \sum_{j=1}^{j=i} S(f_i) & 1 \le i \le n \\ N & i > n \end{cases}}$$ 2.4

Therefore for a given maximum fitness f, a individual x' $= (s'_1,\ldots.s'_n)$ is called an optimum if and only if, for all x that satisfy $H(x, x') = 1, \quad f(x') > f(x)$ holds.

For instant, The Hamming distance between two individuals X and Y and the distance is measured based on fitness values of the neurons and is given by

$$\boldsymbol{H(X, Y) = \sum_{i=1}^{n} |S_i - S'_i|}$$ 2.5

Where $X = (s_1, s_2, \ldots..s'_n)$ $\quad Y = (s'_1, s'_2, \ldots..s'_n)$

Therefore the generations of individuals from the 'r' number of groups is given as

$$\boldsymbol{P(a'_1) = \frac{1}{r^p}(\,((r-1)+1)^{\,2} + (r-1)^{\,2}}$$ 2.6

The last generation or rth group gives the optimum generation and is the best solution.

D. Termination Criterion

The training process is used to achieve the global minima. The training error of an ANN may reduce as its training process progress. In the proposed EA a simple criterion that terminates the training process of the ANN when it's

```
%Tournament selection
For r =1:2*pop size;
    pick P number of Challengers randomly, where P = 10% of pop size; arrange the tournament and find the minimum hamming distance using the error mean square for N number of training sets between r^th solution and selected P challengers.;
    Define score (Hamming distance) of tournament for r^th solution
End
Arrange score of all solutions in ascending order;
  Sp=pick up the best half score position;
  Select next generation solution as solution corresponding to position sp;
```

Figure 1: Pseudo code of Tournament Selection

Training error increases for T successive times measured at the end of T successive strips. At each strip EA adds one hidden neuron/layer and retrain the modified ANN architecture, or prunes several hidden neurons and retains the modified ANN or trains the existing ANN architecture. This criterion can be expressed as

$$\mathbf{E}(\mathbf{i}) < \mathbf{E}(\mathbf{i}+\mathbf{j}) \qquad \mathbf{j} = \mathbf{1}, \mathbf{2}, \ldots \ldots \mathbf{T} \tag{2.7}$$

Where $\mathbf{E}(\mathbf{i})$ **and** $\mathbf{E}(\mathbf{i}+\mathbf{j})$ are the errors of the ANN at epochs i and i + j respectively and T is a parameter specified by the user. The termination criterion is to stop the training process of the ANN when its training error increases not just once but during T consecutive times. The proposed EA computes these errors on the training set and test the termination criterion after completion of every generation.

III. EXPERIMENTAL STUDIES

The proposed method of optimization algorithm is applied to benchmark data classification problems using the feed-forward architecture with a bias of +1 input for hidden layer and output layer. The three classification problems are

i) Pima-India-Diabetes dataset problems.
ii) SPECT Heart data set
iii) Breast-Cancer dataset problem

All the data, applied to the training and test sets, are acquired from the UCI Machine Learning Repository [12]. The number of training and test data sets is given in Table 1. Testing error rate (TER) is one more paraeter in real time data classification problems, which refers to the percentage of wrong classification produced by ANNs on the testing set.

A. Experimental Setup of EANN

The evolutionary process attempts to crossover and mutate weights before performing any structural or topology crossover and mutation. The weights of ANNs were initialized to random values in the range between −0.5 and +0.5.For evolving connection weights population size in EA is taken as 20 and 10 independent trials have been conducted to get the generalized behaviour. For architecture the number of training generations for partial training is set to 40, the values ε and η are set in between 1E-02 to 1E-06 and0.03 to 0.06 respectively. The value of T used in the termination criterion is set to 2 (for real classification problems).

B. Experimental Result

In this work, the data sets of different problems were partitioned into three sets: a training set, and testing set. The number of examples in these sets is shown in the Table I. The training set was used to train and modify ANN architectures. The testing set for measuring their generalization ability. The performance results on three benchmark real data classification problems are given in Table 1 and Table 2. All the performances of the classification problems are averaged over 10 independent trail runs on training sets. First, for Pima Indians the average training error rate (TER) of 23.5 is obtained. The resultant architecture is optimized with an average of 3.02 hidden neurons and 1.2 average hidden layers and the average percentage of MSE performance of the architecture is equal to 77.50. Second by, for SPECT Heart problem the average TER is equal to 13.5 over 10 trial runs. The resulting architecture is tested using 67 test sets and minimum average number of hidden neurons is 3.41 with average number of hidden layers of 1.2. The average percentage of performance of proposed EA on the architecture is equal to 85.84. Third by, for the Breast Cancer problem, the optimized network with average hidden neurons and layers are 2.01 and 1.1 is obtained. The best average TER of 3.0 is achieved. The optimized architecture is tested with 240 test set. The average percentage of performance of proposed EA on the architecture is equal to 98.5.

The mean square error (MSE) performance for three real data classification problems are shown in Fig (2) for thousand generations. These performances were obtained over training sets averaged over ten independent trail runs, the averaged values of MSE are given in Table 3. Table 3 gives the comparison of Gradient algorithm with the proposed EA in terms of epochs count, MSE and total generation/iteration time. The proposed EA takes the more generations and the time in the training level but during testing it shows good performance over larger number of test sets with minimum time.

Table 1 Performance of EA on Three Real Time Dataset Classification Problem.

Parameter	Experimental Results		
	Pima-Indians	SPECT-Heart	Cancer
Number Of Runs	10	10	10

Average Number Of Generations	61	103	52
Number of Training patterns used	500	200	400
Average Training Set Accuracy in percentage	**76.5**	**87.2**	**97.0**
Number of Test patterns used	268	67	240
Average Test Set Accuracy in percentage	**77.5**	**85.12**	**98.02**
Initial Number of Hidden layers / Neurons	**2 / [5 4]**	**2 / [5 4]**	**2 / [5 4]**
Final Number of Hidden layers / Neurons (Resulted NN)	**1.2 / [3.02]**	**1.4 / [3.41]**	**1.1/ [2.01]**
Population size	50	50	50
Number of inputs	09	14	11
Number of outputs	01	01	01

Table 2 Performance of Proposed EA on 3 bench mark problems. All results were averaged over 10 independent trails with ε is 1E-06, η is taken as 0.03 and T is taken as 2

Problem	Number of		TER
	Hidden Neurons	Epochs	
Pima Indians Diabetes	3.2	358.7	23.5
SPECT Heart	3.41	190.6	15.1
Breast Cancer	2.01	228.2	2.86

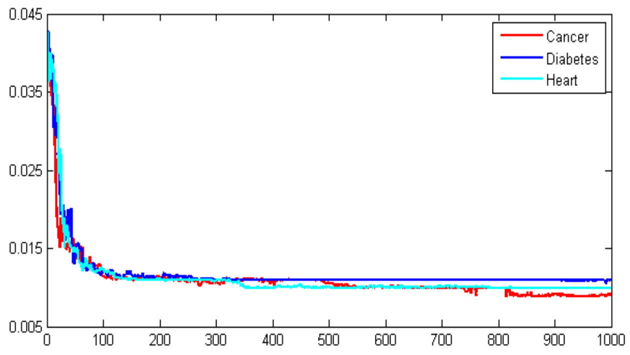

Figure 2 MSE performance of the Three Real Data Classification for 1000 generations. All are averaged over 10 Independent runs

A. Effect of Parameter Values On Proposed EA

We performed a set of new experiments using proposed EA with three different values for τ , ε, η, and T given in Table 4. The values for τ and ε were chosen in the range of 10–40 and 1E-02–1E-06, respectively, while Those for η and T were chosen in the range of 0.03–0.07 and 1–5, respectively. The average results of the new experiments over 10 independent runs are presented in Table 3. It is seen that a small or moderate value of τ, η, and T, and a large value of ε are beneficial for both training epochs and TER. The proposed EA performed very badly when the value of T was chosen very large (e.g., 5). This is reasonable because EA terminated only when the training error increased five consecutive times in each of the five consecutive strips. Thus, EA needed a large number of training epochs to satisfy the termination criterion. This allowed ANNs to learn very detailed information from the training data, resulting in poor generalization ability, i.e., a large TER.

The proposed EA also compared on number of hidden neurons, generations and TER with some different algorithms like BCA, BTA and BCPA [13], [[14], [15] given in Table 5. All the values are obtained for ε is 1E-06, η is taken as 0.03 and T is taken as 2.

IV. Conclusion and Future scope

The generalization ability of ANNs is greatly dependent on their architectures. The proposed EA performed well with respect to the mean square error, adaptive optimal architecture and generalization when compared to the different algorithms in the literature. The time complexity shows that the initial training takes the selection I and the after some epochs to reach along the path, higher selection pressure like selection II is quite effective. One of the future improvements to proposed EA would be to reduce the number of parameters or make them adaptive. In addition, the use of a different significance criterion in the merging operation of EA would also be an interesting future research topic. Since proposed EA has been applied to the classification problems, it would be interesting to study how well EA would perform on regression problems**.**

Table 3.0 Average Performance ANN using EA on Three Real Time dataset problem w.r.t the time, number of iterations / gene-rations and fitness averaged over 10 independent runs.

Name of the benchmark problem	**Gradient Algorithm**			**Evolutionary Algorithm**			
	Training process			**Training process**			**Total test time(s)**
	Itera-tions	**Total time(s)**	**MSE**	**Generatio ns**	**Total time(s)**	**MSE**	
Pima Indians Diabetes	635	23.9591	0.3948	358.7	918E+3	8.6214E-3	67.4160
SPECT Heart	356	15.0294	0.2661	190.6	487E+3	7.7264E-3	16.750
Breast Cancer	315	9.1792	0.2417	228.2	586E+3	5.3614E-3	60.361

Table 4 Performance of Proposed EA on Three Benchmark Classification Problems with T = 5. All Results Were Averaged Over 10 Independent Runs

Problem	Parameter			Number of		
		ε	η	Hidden Neurons	Epochs	TER
Pima Indians Diabetes	10	1E-03	0.03	3.21	350.2	24.61
	20	1E-05	0.04	3.26	385.1	23.75
	40	1E-06	0.06	4.23	652.4	25.36
SPECT Heart	10	1E-03	0.03	4.27	235.4	19.57
	20	1E-05	0.04	4.85	342.8	21.41
	40	1E-06	0.06	5.73	526.4	23.18
Breast Cancer	10	1E-03	0.03	3.15	180.1	2.41
	20	1E-05	0.04	3.27	223.4	2.86
	40	1E-06	0.06	4.53	292.4	3.15

Table 5 Comparison of different methods in literature based on architecture size and TER for three real dataset classification problems.

Algorithm or Method	**Diabetes problem**			**SPECT-Heart problem**			**Cancer problem**		
	Number of		**TER**	**Number of**		**TER**	**Number of**		**TER**
	Hidden neurons	**Genera-tions**		**Hidden neurons**	**Genera-tions**		**Hidden neurons**	**Genera-tions**	
BCA	5.96	467.5	26.04	3.42	173.4	20.34	2.20	290.2	1.92
BPA	5.56	409.6	26.25	3.12	161.4	19.93	1.60	263.3	1.89
BCPA	5.80	501.3	26.22	3.26	190.7	20.43	2.12	311.5	1.95
OBD	16.0	--	31.4	--	--	--	8.0	--	7.5
OBS	26.0	--	34.60	--	--	--	7.0	--	10.0
EP NET	3.4	--	22.37	--	--	--	2.0	--	1.37
Proposed EA	**3.02**	**358.7**	**23.5**	**3.41**	**190.6**	**15.1**	**2.01**	**228.2**	**2.86**

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in Parallel Distributed Processing, vol. I, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge,MA: MIT Press, 1986, pp. 318–362.

[2] T. Y. Kwok and D. Y. Yeung, "Constructive algorithms for structure learning in feed forward neural networks for regression problems," IEEE

Trans. Neural Netw., vol. 8, no. 3, pp. 630–645, May 1997.

[3] R. Reed, "Pruning algorithms—A survey," IEEE Trans. Neural Netw., vol. 4, no. 5, pp. 740–747, Sep. 1993.

[4] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks Architectures," Neural Comput., vol. 7, no. 2, pp. 219–269, Mar. 1995.

[5] J. H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: Univ. Michigan Press, 1975.

[6] L. J. Fogel, A. J. Owens, and M. J. Walsh, Artificial Intelligence Through Simulated Evolution. New York: Wiley, 1966.

[7] D. B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. New York: IEEE Press, 1995.

[8] H.-P. Schwefel, Numerical Optimization of Computer Models. Chichester, U.K.: Wiley, 1981.

[9] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," IEEE Trans. Neural Netw., vol. 5, no. 1, pp. 54–65, Jan. 1994.

[10] X. Yao, "Evolving artificial neural networks," Proc. IEEE, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.

[11] Md.Monirul et al., "A new adaptive merging and growing algorithm for designing artificial Neural," IEEE Trans. Man, and Cybernetics., vol. 39, no. 3, Apr. 2009, pp. 705–722.

[12] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases,1998.

[13] R. Reed, "Pruning algorithms—A survey, "IEEE Trans. Neural Netw., vol.4,no.5,pp.740–747,Sep.1993.

[14] Y. LeCun, J.S.Denker, and S.A.Solla, "Optimal brain damage," in Proc. Advances Neural Inform. Process. Syst., D.S. Touretzky, Ed.SanMateo, CA.Morgan Kaufmann,1990,vol.2, pp.598–605.

[15] B.Hassibi and D.G.Stork, "Second-order derivatives for network pruning: Optimal brain surgeon," in Proc. Advances Neural Inform. Process. Syst., C.Lee, S.Hanson, andJ.Cowan, Eds.San Mateo, CA:Morgan Kaufmann,1993,vol.5,pp.164–171.

[16] H. C. Andersen and A. C. Tsoi, "A constructive algorithm for the training of a multilayer perceptron based on the genetic algorithm," Complex Syst., vol. 7, no. 4, pp. 249–268, 1993.

[17] Y. Hirose, K. Yamashita, and S. Hijiya, "Back-propagation algorithm which varies the number of hidden units," Neural Networks, vol. 4, no. 1, pp. 61–66, 1991.

[18] M. W. Hwang, J. Y. Choi, and J. Park, "Evolutionary projection neural networks," in Proc. 1997 IEEE Int. Conf. Evolutionary Computation, ICEC'97, pp. 667–671.

[19] G.V.R. Sagar, K. Anitha sheela "Architecture design and time complexity of artificial neural network using evolutionary algorithm" Elixir International Journal, Vol.65, pp 19736 – 19744, 2013.