Hindawi Publishing Corporation
Advances in Artificial Neural Systems
Volume 2014, Article ID 629137, 9 pages
http://dx.doi.org/10.1155/2014/629137

Research Article

Modeling Slump of Ready Mix Concrete Using Genetically Evolved Artificial Neural Networks

Vinay Chandwani, Vinay Agrawal, and Ravindra Nagar

Department of Civil Engineering, Malaviya National Institute of Technology Jaipur, JLN Marg, Jaipur, Rajasthan 302017, India

Correspondence should be addressed to Vinay Chandwani; chandwani2@yahoo.com

Received 29 August 2014; Accepted 20 October 2014; Published 11 November 2014

Academic Editor: Ping Feng Pai

Artificial neural networks (ANNs) have been the preferred choice for modeling the complex and nonlinear material behavior where conventional mathematical approaches do not yield the desired accuracy and predictability. Despite their popularity as a universal function approximator and wide range of applications, no specific rules for deciding the architecture of neural networks catering to a specific modeling task have been formulated. The research paper presents a methodology for automated design of neural network architecture, replacing the conventional trial and error technique of finding the optimal neural network. The genetic algorithms (GA) stochastic search has been harnessed for evolving the optimum number of hidden layer neurons, transfer function, learning rate, and momentum coefficient for backpropagation ANN. The methodology has been applied for modeling slump of ready mix concrete based on its design mix constituents, namely, cement, fly ash, sand, coarse aggregates, admixture, and water-binder ratio. Six different statistical performance measures have been used for evaluating the performance of the trained neural networks. The study showed that, in comparison to conventional trial and error technique of deciding the neural network architecture and training parameters, the neural network architecture evolved through GA was of reduced complexity and provided better prediction performance.

1. Introduction

Cement concrete is one of the most widely used construction materials in the world today. The material modeling of concrete is a difficult task owing to its composite nature. Although various empirical relationships in the form of regression equations have been derived from experimental results and are widely in use, these do not provide accuracy and predictability wherein the interactions among the number of variables are either unknown, complex, or nonlinear in nature. Artificial neural networks (ANNs), touted as the next generation of computing, have been the preferred choice since the last few decades for modeling unstructured problems pertaining to material behavior. The notable applications of ANN in modeling properties of concrete are implementations in predicting and modeling compressive strength of high performance concrete [1], self-compacting concrete [2], recycled aggregate concrete [3], rubberized concrete [4], fibre reinforced polymer (FRP) confined concrete [5], durability of high performance concrete [6], predicting drying shrinkage of concrete [7], concrete mix design [8], and prediction of elastic modulus of normal and high strength concrete [9].

Known for its design mix precision leading to better quality concrete and ease of transportation and laying at the construction site, ready mix concrete (RMC) has emerged as a preferred concrete construction product catering to the requirement of the end user. One of the important properties of concrete that play an important role in the success of RMC is its workability. Workability of concrete is determined by the effort to lay and compact the freshly prepared concrete at construction site with minimum loss of homogeneity. The workability of concrete quantitatively measured in terms of slump value is an important quality parameter in RMC industry. The slump of concrete depends on the concrete's design mix proportion. As noticed for other properties of concrete, the slump value also exhibits a highly nonlinear and complex functional relationship with concrete's constituents. Recent studies, like prediction of slump and strength of ready mix concrete containing retarders and high strength concrete

containing silica fume and plasticizers [10], predicting slump of fly ash and slag concrete [11], modeling slump of high strength concrete [12], modeling slump of high performance concrete [13], and modeling and analysis of concrete slump using laboratory test results [14], have proved the effectiveness of ANN in modeling slump of concrete.

Besides ANN applications in modeling behavior of concrete discussed above, there are many multidisciplinary applications of ANN which are beyond the scope of this paper. The reason for this rapid growth in the field of neural networks is attributed to its "black box" nature which allows it to be applied to almost every available problem, without seeking the knowledge of underlying relationships among the input and output variables. In spite of its popularity as a universal function approximator, the neural networks are still being designed using trial and error approach. Generally, iterative techniques using different combinations of number of hidden layers and hidden layer neurons are employed in conjunction with different learning rate, momentum coefficient, and transfer function to arrive at optimal neural network design. This technique of designing a neural network is therefore time consuming and relies heavily on the experience of the designer. In order to reduce the effort and time in designing the optimal neural network architecture and its training parameters, various studies for automatic design of neural network have been successfully performed in the past by harnessing the stochastic search ability of genetic algorithms [15–19]. The autodesign of neural network using genetic algorithms (GA) has not been so far employed for modeling the slump of concrete. The research paper presents a methodology for evolving an optimal architecture of neural network and its training parameters using GA for modeling the slump of concrete based on concrete's design mix proportions.

The study has been organized into sections. Section 2 deals with data collection and its description. Section 3 deals with the methodology, in which trial and error neural network modeling of concrete slump, evolving neural network architecture, and training parameters using genetic algorithms and statistical performance measures have been discussed. Results, discussions, and conclusions have been dealt with in Sections 4, 5, and 6, respectively.

2. Collection of Data and Its Description

The exemplar data for ANN was collected from the same RMC plant to mitigate any chance of change caused in the slump data due to change in physical and chemical properties of the concrete design mix constituents. The collected data comprised 560 concrete design mix proportions and their corresponding slump test values. The design mix proportions included weight per m^3 of cement, pulverized fly ash (PFA), sand (as fine aggregate), coarse aggregate 20 mm, coarse aggregate 10 mm, admixture, and water-binder ratio. The range (maximum and minimum values) of the RMC data used in the study is shown in Table 1.

Table 1: Range of RMC data used for neural network modeling.

RMC data	Maximum	Minimum
Cement (kg/m^3)	425	100
Fly ash (kg/m^3)	220	0
Sand (kg/m^3)	900	550
Coarse aggregate 20 mm (kg/m^3)	788	58
Coarse aggregate 10 mm (kg/m^3)	771	343
Admixture (kg/m^3)	5.5	1.0
Water-binder ratio	0.76	0.36
Concrete slump (mm)	175	110

3. Methodology

For conducting the study, the Neural Network Toolbox and Global Optimization Toolbox included in the commercially available software MATLAB R2011b (version 7.13.0.564) were used to implement the BPNN and GA, respectively.

3.1. Dividing Data into Training, Validation, and Test Data Sets. ANN derives learning capabilities through training using input-output data pairs and subsequent generalization ability when subjected to unseen data. The training and generalization of neural networks are accomplished using a training data set and a validation data set, respectively. The robustness of the trained and validated neural network is tested using a test data set. This procedure is accomplished by dividing the entire data into three disjoint sets, namely, training data set, validation data set, and test data set. The available data was randomized and 70% of the data was designated as training data set and the remaining 30% data was equally divided to create the validation and test data sets, respectively.

3.2. Normalization of Data. The data used for training, validation, and testing of neural networks comprise inputs and corresponding output features of different identities, which normally have minimum similarities. Moreover the range (minimum–maximum values) of the data used for each input and output component is also quite different. In order to scale down all the inputs and outputs in a particular bound range preferably −1 to +1 or 0 to +1, data normalization is performed. This type of normalization has the advantage of preserving exactly all relationships in the data and it does not introduce any bias [20]. This facilitates the learning speed, as these values fall in the region of sigmoid transfer function where the output is most sensitive to the variations of the input values [21]. Linear scaling in the range −1 to +1 has been used in the present study having function

$$x_{\text{norm}} = \frac{2 * (x - x_{\min})}{(x_{\max} - x_{\min})} - 1, \tag{1}$$

where x_{norm} is the normalized value of the variable x and $x_{\max}$ and $x_{\min}$ are the minimum and maximum values of variable x, respectively.

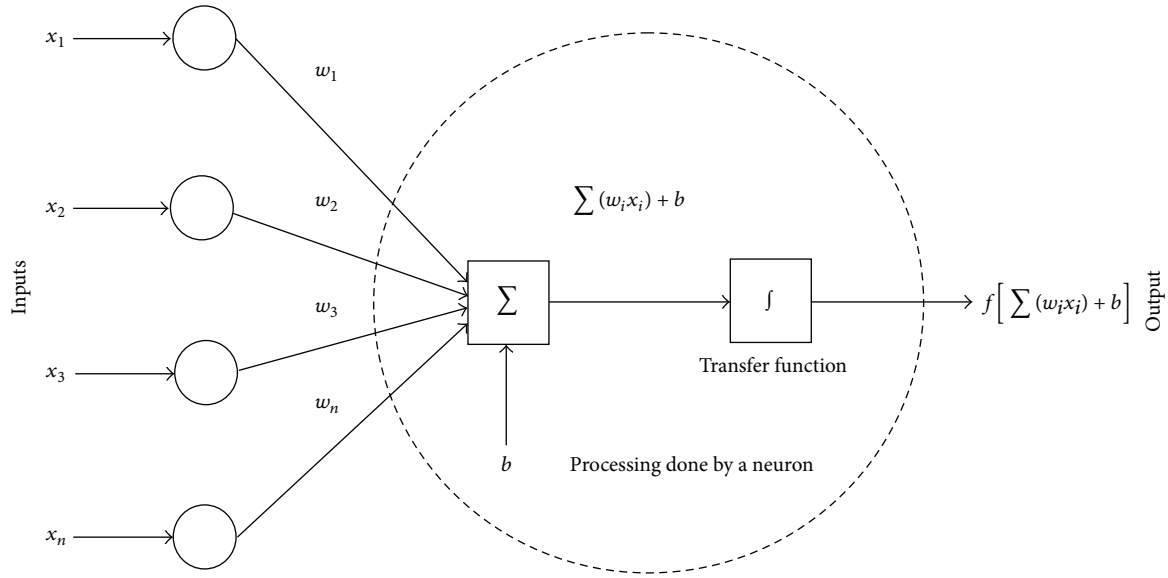

Figure 1: Mathematical model of an artificial neuron.

3.3. Neural Network Architecture and Training Parameters. An artificial neural network is an information processing paradigm which presents a computational analogy inspired by human brain. ANN consists of processing elements called the artificial neurons which are arranged in layers. The computational structure of an artificial neuron comprises several inputs $x_1, x_2, \ldots, x_n$ and an output y. Every input is assigned a weight factor $(w_1, w_2, \ldots, w_n)$ signifying the importance of the input. The synaptic weights carry both positive and negative values. A positive weight value leads to the forward propagation of information, whereas a negative value inhibits the signal. The neuron has an additional input called bias b and is interpreted as an additional weight. Bias has a constant value and its incorporation within the neuron helps in learning of the neural network. The inputs multiplied by corresponding weights are summed up and form the argument of the neuron's activation function. Figure 1 represents the mathematical model of an artificial neuron.

The architecture of a neural network consists of three basic layers denoted by "input layer," "output layer," and a number of intermediate "hidden layer/s". In case of multilayer feedforward neural networks (MFNN), the neurons in each layer are connected in the forward direction only and no intralayer connections between the neurons are permitted. The input features form the neurons of the input layer and output features are represented by the neurons of the output layer. The number of hidden layers and hidden layer neurons depends on the number of training cases, the amount of noise, and the degree of complexity of the function or the classification desired to be learnt [22]. Hornik et al. [23] concluded that a three-layer feedforward neural network with backpropagation algorithm can map any nonlinear relationship with desired degree of accuracy. Some "rules of thumb" acting as initial guidelines for choosing neural network architecture have been suggested by [24–27]. Nevertheless, the selection of hidden layers and hidden layer neurons is a trial and error process and generally started by choosing a network with a minimum number of hidden layers and hidden neurons.

MFNNs trained using backpropagation (BP) algorithms are commonly used for tasks associated with function approximation and pattern recognition. Backpropagation algorithm, in essence, is a means of updating neural network synaptic weights by backpropagating a gradient vector in which each element is defined as the derivative of an error measure with respect to a parameter [28]. BP algorithm is a supervised learning algorithm wherein exemplar patterns of associated input and output data values are presented to the neural network during training. The information from input layer to output layer proceeds in forward direction only and with the help of BP algorithm, the error computed is passed backwards and the weights and biases are iteratively adjusted to enable learning of neural network by reducing network error to a threshold value.

A suitable learning rate and momentum coefficient are employed for efficient learning of the network. A higher learning rate leads to faster training, but by doing so it produces large oscillations in the weight change which may force the ANN model to overshoot the optimal weight values. On the other hand, a lower learning rate makes convergence slower and increases the probability of ANN model to get trapped in local minima. The momentum term effectively filters out the high frequency variations of the error surface in the weight space, since it adds the effect of the past weight changes on the current direction of movement in the weight space [29]. A combined use of these parameters helps the BP algorithm to overcome the effect of local minima. The utility of transfer functions in neural networks is to introduce nonlinearity into the network. A consequence of the nonlinearity of this transfer function in the operation

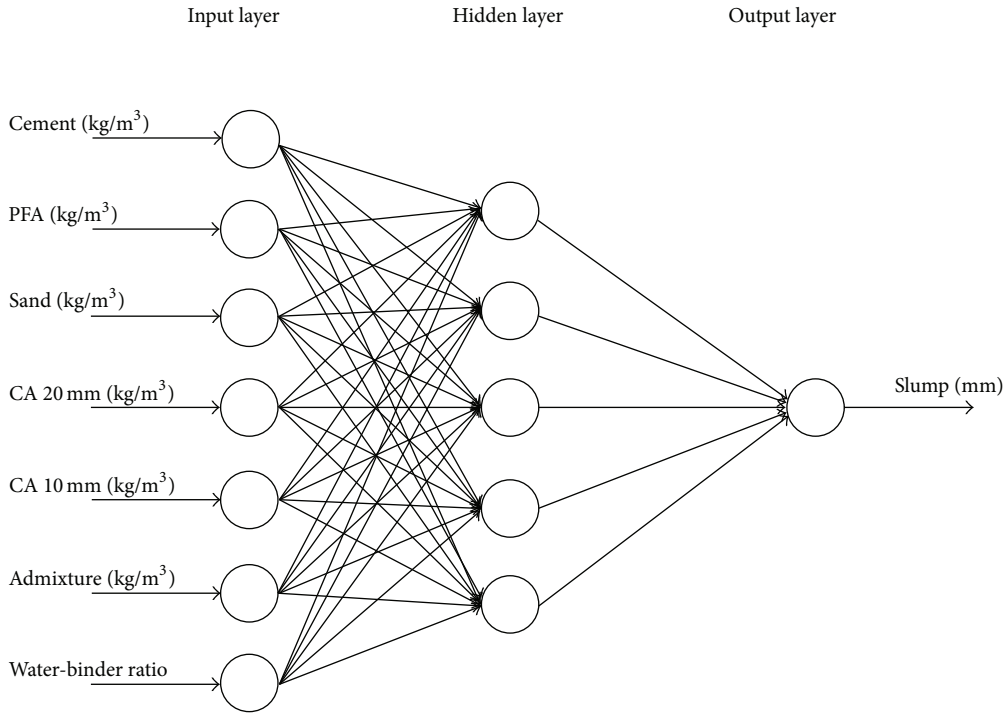

Figure 2: Single hidden layer neural network with five hidden layer neurons.

of the network, when so introduced, is that the network is thereby enabled to deal robustly with complex, undefined relations between the inputs and the output [30].

3.4. Evolving Neural Network Architecture and Training Parameters Using Trial and Error. In the present study, the input layer consists of seven neurons, namely, cement, fly ash, sand, coarse aggregate (20 mm), coarse aggregate (10 mm), admixture content, and water-binder ratio. The output layer comprises a single neuron representing the slump value corresponding to the seven input neurons defined above. In this study eleven single hidden layer neural network architectures of different complexities with hidden layer neurons varying in the range 5 to 20 have been used for evolving the optimal neural network architecture. The neural network architecture with five hidden layer neurons for the present study is shown in Figure 2. The learning rate and momentum coefficient have been varied in the range 0 to +1. Hyperbolic tangent and log-sigmoid transfer functions have been used in the hidden layers along with linear transfer function in the output layer. Different combinations of learning rate and momentum coefficient with hidden layer transfer function have been tried for effective training of neural networks.

The neural networks were trained using the training data set. The information to the neural network was presented through input layer neurons. The information propagated in the forward direction through hidden layers and was processed by the hidden layer neurons. The network's response at the output layer was evaluated and compared with actual output. The error between the actual and the predicted response of the neural network was computed and propagated in the backward direction to adjust the weights and biases of the neural network. Using the BP algorithm the weights and biases were adjusted in a manner to render error to a minimum value. In the present study, Levenberg-Marquardt backpropagation algorithm which is the fastest converging algorithm preferred for supervised learning has been used as training algorithm. During training process, the neural network has a tendency to overfit the training data. This leads to poor generalization when the trained neural network is presented with unseen data. The validation data set is used to test the generalization ability of trained neural at each iteration cycle. Early stopping of neural network training is generally undertaken to avoid overfitting or overtraining of the neural network. In this technique the validation error is also monitored at each iteration cycle along with training error and the training is stopped once the validation error begins to increase. The neural network architecture having the least validation error is selected as the optimal one.

3.5. Evolving Neural Network Architecture and Training Parameters Using Genetic Algorithms. Genetic algorithm (GA) inspired by Darwin's theory "survival of the fittest" is a global search and optimization algorithm which involves the use of genetic and evolution operators. GA is a population based search technique that simultaneously works on a number of probable solutions to a problem at a time and uses probabilistic operators to narrow down the search to the region where there is maximum possibility of finding an optimal solution. GA presents a perfect blend of exploration and exploitation of the solution search space by harnessing computational models of evolutionary processes like selection, crossover, and mutation. GAs outperform

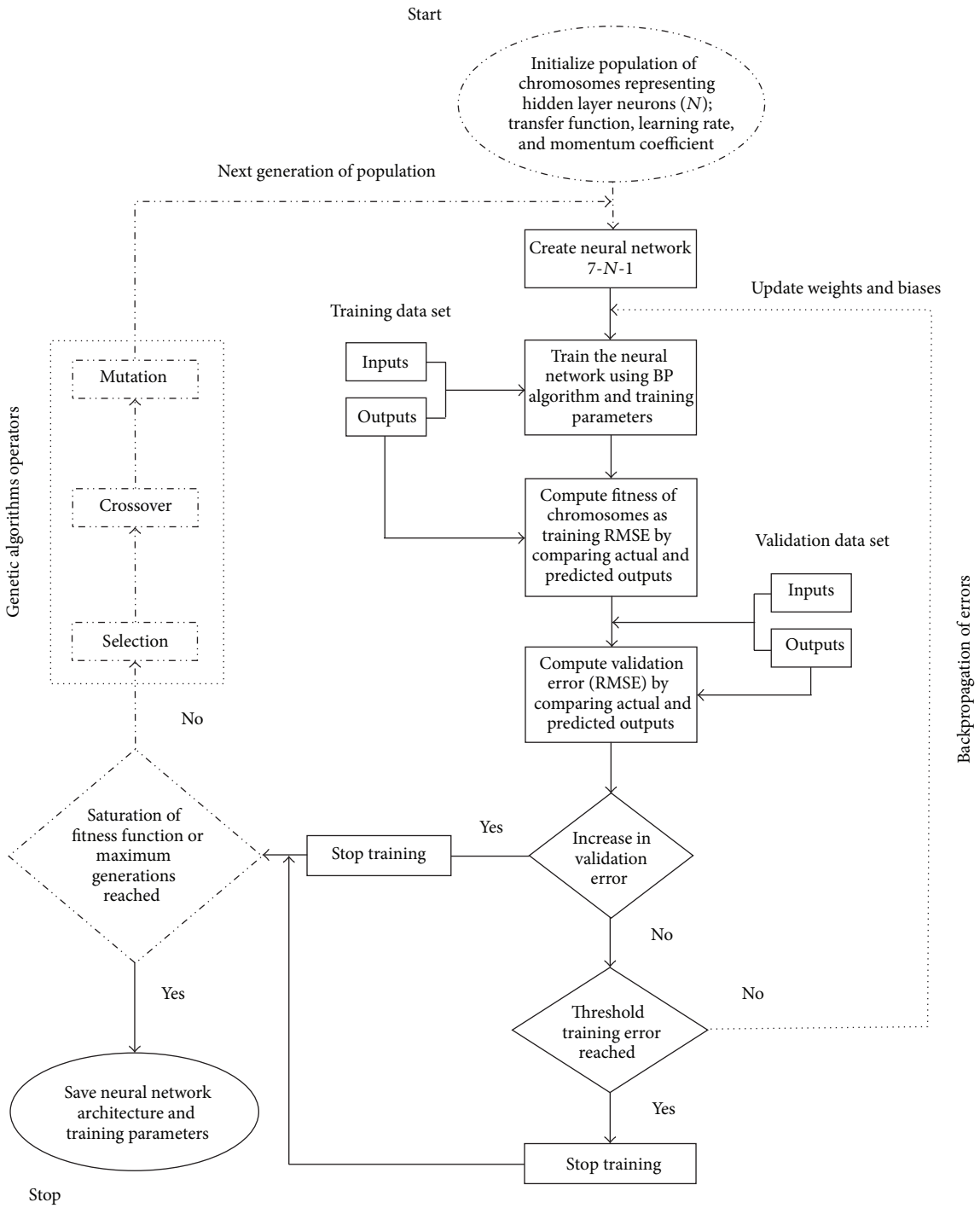

Figure 3: Evolving neural network architecture and training parameters using GA.

the efficiency of conventional optimization techniques in searching nonlinear and noncontinuous spaces, which are characterized by abstract or poorly understood expert knowledge [31].

Automatic design of neural network architecture and its training parameters is accomplished by amalgamating GA with ANN during its training process. The methodology uses GA to evolve neural network's hidden layer neurons and transfer function along with its learning rate and momentum coefficient. The BP algorithm then uses these ANN design variables to compute the training error. The training process is monitored at each iteration by computing the validation error. The training of neural network is stopped once validation error starts to increase. This process is repeated number of times till optimum neural network architecture and its training parameters are evolved. The steps of this methodology are presented as flow chart in Figure 3 and are summarized as below.

(1) Initialization of Genetic Algorithm with Population of Chromosomes. In GA the chromosomes contain the information regarding the solution to a problem. The chromosomes contain number of genes which represent potential solutions to a problem. Being a population based heuristic, the GA starts with a number of chromosomes representing initial guesses to the possible solutions to a problem. In the present study, the chromosomes represent the information regarding number of hidden layer neurons, transfer function, learning rate, and momentum coefficient. The numbers of hidden layer neurons are initialized in the range 5 to 20 neurons and are represented as discrete integer number. The GA is allowed to select the transfer function between tangent hyperbolic and log-sigmoid transfer functions. The range of learning rate and momentum coefficient is bounded in the range 0 to +1 and is represented by real numbers.

The size of the population is chosen in such a way to promote evolving of optimal set of solutions to a particular problem. A large initial population of chromosomes tends to increase the computational time, whereas a small population size leads to poor quality solution. Therefore population size must be chosen to derive a balance between the computational effort and the quality of solution. In the present study an initial population size of 50 chromosomes is used.

(2) Creating the Neural Network. The neural network is created using randomly generated hidden layer neurons. As discussed above, the number of input layer neurons is 7 and output layer neurons is 1; hence the neural network architecture is initialized as 7-N-1, where N is the number of hidden layer neurons determined using GA. The transfer function for hidden layers is also initialized using GA.

(3) Training of Neural Network and Evaluating Fitness of Chromosomes. The neural network is trained using Levenberg-Marquardt backpropagation training algorithm. The learning parameters, namely, learning rate and momentum coefficient, initialized using GA are used during the training process for systematic updating of neural network weights and biases. BP algorithm updates the weight and biases through backpropagation of errors. Early stopping technique is applied to improve the generalization of the neural network. The fitness function acts as measure of distinguishing optimal solution from numerous suboptimal solutions by evaluating the ability of the possible solutions to survive or biologically speaking it tests the reproductive efficiency of chromosomes. The fitness of each chromosome is computed by evaluating the root mean square error (RMSE) using

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(T_i - P_i\right)^2}, \tag{2}$$

where T_i and P_i denote the target or observed values and ANN predicted concrete slump values, respectively.

(4) Selecting the Fitter Chromosomes. The evolution operator of GA called as selection operator helps in selecting the fitter chromosomes based on the value of the fitness function. A selection operator performs the function synonymous to a filtering membrane, allowing the fitter chromosomes to survive to create new offspring. As one moves from one generation of population to the next, the selection operator gradually increases the proportion of the fitter chromosomes in the population. The present study uses roulette wheel selection strategy which allows probability of selection proportional to the fitness of the chromosome. The basic advantage of roulette wheel selection is that it discards none of the individuals in the population and gives a chance to all of them to be selected [32].

(5) Creating Next Generation of Population. The new generation of population is created through two genetic operators of GA called crossover and mutation. The crossover operator is a recombination operator and is applied to a pair of parent chromosomes with the hope to create a better offspring. This is done by randomly choosing a crossover point and copying the information before this point from the first parent and then copying the information from the second parent beyond the crossover point. The present study utilized the scattered crossover with probability 0.9 for recombining the two parent chromosomes for producing a fitter child.

The mutation operator modifies the existing building blocks of the chromosomes maintaining genetic diversity in the population. It therefore prevents GA from getting trapped at a local minimum. In contrast to crossover which exploits the current solution, the mutation aids the exploration of the search space. Too high mutation rate increases the search space to a level that convergence or finding global optima becomes a difficult issue whereas a lower mutation rate drastically reduces the search space and eventually leads genetic algorithm to get stuck in a local optima. The present study uses uniform mutation with mutation rate 0.02. The procedure for creating new population of chromosomes is continued till maximum generation limit is achieved or the fitness function reaches a saturation level. Maximum number of generations used for present study is 150.

3.6. Evaluating Performance of the Trained Models. The study uses six different statistical performance metrics for evaluating the performance of the trained models. The statistical parameters are mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE), coefficient of correlation (R), Nash-Sutcliffe efficiency (E), and root mean square to standard deviation ratio (RSR). The above performance statistics were evaluated using

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|T_i - P_i\right|,$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(T_i - P_i\right)^2},$$

$$\text{MAPE}\,(\%) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|T_i - P_i\right|}{T_i} \times 100,$$

$$R = \left(\frac{\sum_{i=1}^{N} \left(\left(T_i - \overline{T} \right) \left(P_i - \overline{P} \right) \right)}{\sqrt{\sum_{i=1}^{N} \left(T_i - \overline{T} \right)^2 \sum_{i=1}^{N} \left(P_i - \overline{P} \right)^2}} \right),$$

$$E = 1 - \frac{\sum_{i=1}^{N} \left(T_i - P_i \right)^2}{\sum_{i=1}^{N} \left(T_i - \overline{T} \right)^2},$$

$$\text{RSR} = \frac{\text{RMSE}}{\sqrt{(1/N) \sum_{i=1}^{N} \left(T_i - \overline{T} \right)^2}}, \tag{3}$$

where T_i and P_i denote the target or observed values and ANN predicted values and $\overline{T}$ and $\overline{P}$ represent the mean observed and mean ANN predicted values, respectively. N represents the total number of data. A lower value of MAE, RMSE, MAPE, and RSR indicates good performance of the model. A higher value of E and R statistics above 0.90 indicates good prediction of the model.

4. Results

The neural network architecture was evolved through trial and error process by analyzing 30 different combinations of hidden layer neurons, transfer function, learning rate, and momentum coefficient. The optimal neural network architecture (BPNN) was evolved as 7-11-1 having eleven hidden layer neurons with learning rate 0.45, momentum coefficient 0.85, and tangent hyperbolic hidden layer transfer function. The same operation was performed by incorporating GA during the training of ANN. The GA was able to evolve the optimal neural network architecture and training parameters in 92 generations (Figure 4). The time taken by GA to reach the saturation level of fitness function 2.2357 mm was evaluated as 305.3412 seconds. The GA evolved neural network architecture (ANN-GA) comprised 9 hidden layer neurons and tangent hyperbolic transfer function. The optimal learning rate and momentum coefficient for backpropagation neural network were computed as 0.3975 and 0.9385, respectively. Both ANN-GA and BPNN models subsequent to training were validated and tested. The results in terms of the performance statistics are presented in Table 2.

The entire RMC data was also used for evaluating the prediction ability of the trained models, namely, BPNN and ANN-GA. The regression plots showing the prediction of trained BPNN and ANN-GA models are exhibited in Figures 5(a) and 5(b), respectively. The statistical performance for the entire data set is tabulated at Table 3.

5. Discussions

The results of the study show that amalgamation of GA with ANN during its training phase leads to evolving of optimal neural network architecture and training parameters. In comparison to trial and error BPNN neural network having architecture 7-11-1, the hybrid ANN-GA automatically evolved a less complex architecture 7-9-1. Moreover the optimal training parameters evolved using GA were able to enhance the learning and generalization of the neural network. In comparison to BPNN model, the ANN-GA model provided a lower error statistics MAE, RMSE, MAPE, and RSR value of 1.506 mm, 2.2357 mm, 1.0479%, and 0.1837 during training; 1.6299 mm, 2.4687 mm, 1.0991%, and 0.2044 during validation; and 1.7769 mm, 2.6295 mm, 1.2382%, and 0.2039 during testing, respectively. The trained ANN-GA model gave higher prediction accuracy with higher values of statistics R and E during training, validation, and testing of trained models. The performance statistics computed for the entire data set using the trained ANN-GA model shows a lower MAE, RMSE, MAPE, and RSR value of 1.5754 mm, 2.3345 mm, 1.0841%, and 0.1902, respectively, and higher E and R values of 0.9819 and 0.9638, respectively, in comparison

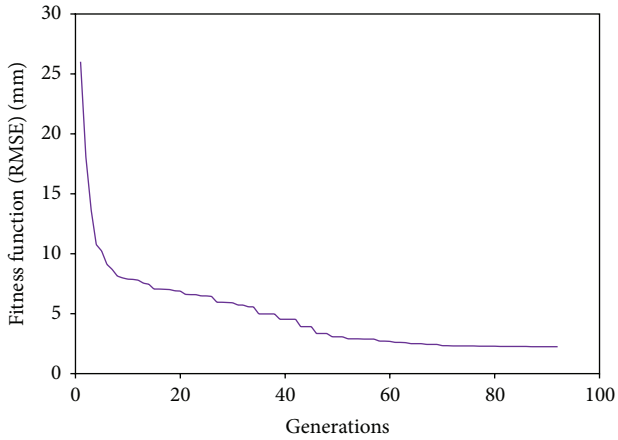

Figure 4: Fitness function (RMSE) versus generation.

Table 2: Statistical performance of ANN models for training, validation, and testing data sets.

Model	MAE (mm)	RMSE (mm)	MAPE (%)	R	E	RSR
Training						
ANN	1.7378	2.4027	1.1862	0.9804	0.9610	0.1974
ANN-GA	1.506	2.2357	1.0479	0.9830	0.9663	0.1837
Validation						
ANN	1.9829	2.7489	1.3474	0.9746	0.9482	0.2276
ANN-GA	1.6299	2.4687	1.0991	0.9794	0.9582	0.2044
Testing						
ANN	2.0651	2.9582	1.3916	0.9735	0.9474	0.2294
ANN-GA	1.7769	2.6295	1.2382	0.9803	0.9584	0.2039

Table 3: Statistical performance of the trained ANN models for the entire data set.

Model	MAE (mm)	RMSE (mm)	MAPE (%)	R	E	RSR
Overall						
ANN	1.8236	2.5470	1.2412	0.9783	0.9569	0.2075
ANN-GA	1.5754	2.3345	1.0841	0.9819	0.9638	0.1902

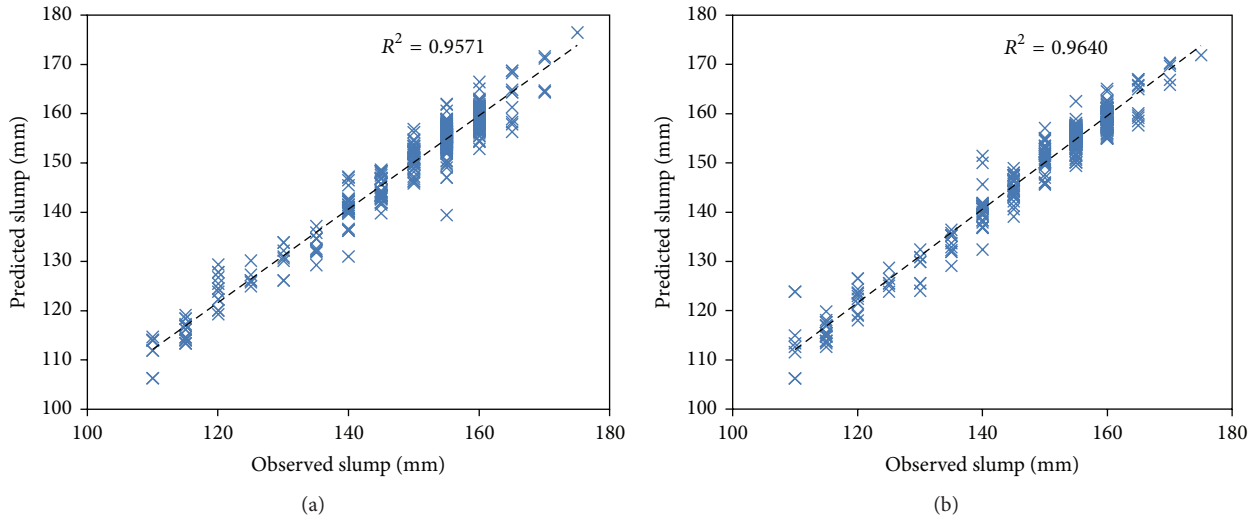

Figure 5: Regression plot of BPNN and ANN-GA predicted slump versus observed slump.

to trained BPNN model. Overall, the performance metrics show that the ANN-GA model has consistently outperformed the BPNN model.

6. Conclusions

The study presented a methodology of designing the neural networks using genetic algorithms. Genetic algorithms population based stochastic search was harnessed during the training phase of the neural networks to evolve the number of hidden layer neurons, type of transfer function, and the values of learning parameters, namely, learning rate and momentum coefficient for backpropagation based ANN.

The performance metrics show that ANN-GA model outperformed the prediction accuracy of BPNN model. Moreover the GA was able to automatically determine the number of hidden layer neurons which were found to be less than those evolved using trial and error methodology. The hybrid ANN-GA provided a good alternative over time consuming conventional trial and error technique for evolving optimal neural network architecture and its training parameters.

The proposed model based on past experimental data can be very handy for predicting the complex material behavior of concrete in quick time. It can be used as a decision support tool, aiding the technical staff to easily predict the slump value for a particular concrete design mix. This technique will considerably reduce the effort and time to design a concrete mix for a customized slump without undertaking multiple trials. Despite the effectiveness and advantages of this methodology, it is also subjected to some limitations. Since the mathematical modeling of concrete slump is dependent on the physical and chemical properties of the design mix constituents, hence the same trained model may or may not be applicable for accurate modeling of slump on the basis of design mix data obtained from other RMC plants deriving its raw material from a different source.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete Research*, vol. 28, no. 12, pp. 1797–1808, 1998.

[2] M. Uysal and H. Tanyildizi, "Estimation of compressive strength of self compacting concrete containing polypropylene fiber and mineral additives exposed to high temperature using artificial neural network," *Construction and Building Materials*, vol. 27, no. 1, pp. 404–414, 2012.

[3] Z. H. Duan, S. C. Kou, and C. S. Poon, "Prediction of compressive strength of recycled aggregate concrete using artificial neural networks," *Construction and Building Materials*, vol. 40, pp. 1200–1206, 2012.

[4] A. Abdollahzadeh, R. Masoudnia, and S. Aghababaei, "Predict strength of rubberized concrete using atrificial neural network," *WSEAS Transactions on Computers*, vol. 10, no. 2, pp. 31–40, 2011.

[5] H. Naderpour, A. Kheyroddin, and G. G. Amiri, "Prediction of FRP-confined compressive strength of concrete using artificial neural networks," *Composite Structures*, vol. 92, no. 12, pp. 2817–2829, 2010.

[6] R. Parichatprecha and P. Nimityongskul, "Analysis of durability of high performance concrete using artificial neural networks," *Construction and Building Materials*, vol. 23, no. 2, pp. 910–917, 2009.

[7] L. Bal and F. Buyle-Bodin, "Artificial neural network for predicting drying shrinkage of concrete," *Construction and Building Materials*, vol. 38, pp. 248–254, 2013.

[8] T. Ji, T. Lin, and X. Lin, "A concrete mix proportion design algorithm based on artificial neural networks," *Cement and Concrete Research*, vol. 36, no. 7, pp. 1399–1408, 2006.

[9] F. Demir, "Prediction of elastic modulus of normal and high strength concrete by artificial neural networks," *Construction and Building Materials*, vol. 22, no. 7, pp. 1428–1435, 2008.

[10] W. P. S. Dias and S. P. Pooliyadda, "Neural networks for predicting properties of concretes with admixtures," *Construction and Building Materials*, vol. 15, no. 7, pp. 371–379, 2001.

[11] I.-C. Yeh, "Exploring concrete slump model using artificial neural networks," *Journal of Computing in Civil Engineering*, vol. 20, no. 3, pp. 217–221, 2006.

[12] A. Öztaş, M. Pala, E. Özbay, E. Kanca, N. Çağlar, and M. A. Bhatti, "Predicting the compressive strength and slump of high strength concrete using neural network," *Construction and Building Materials*, vol. 20, no. 9, pp. 769–775, 2006.

[13] I.-C. Yeh, "Modeling slump flow of concrete using second-order regressions and artificial neural networks," *Cement and Concrete Composites*, vol. 29, no. 6, pp. 474–480, 2007.

[14] A. Jain, S. Kumar Jha, and S. Misra, "Modeling and analysis of concrete slump using artificial neural networks," *Journal of Materials in Civil Engineering*, vol. 20, no. 9, pp. 628–633, 2008.

[15] R. B. Boozarjomehry and W. Y. Svrcek, "Automatic design of neural network structures," *Computers & Chemical Engineering*, vol. 25, no. 7-8, pp. 1075–1088, 2001.

[16] J. S. Son, D. M. Lee, I. S. Kim, and S. K. Choi, "A study on genetic algorithm to select architecture of a optimal neural network in the hot rolling process," *Journal of Materials Processing Technology*, vol. 153-154, no. 1–3, pp. 643–648, 2004.

[17] M. Saemi, M. Ahmadi, and A. Y. Varjani, "Design of neural networks using genetic algorithm for the permeability estimation of the reservoir," *Journal of Petroleum Science and Engineering*, vol. 59, no. 1-2, pp. 97–105, 2007.

[18] P. G. Benardos and G.-C. Vosniakos, "Optimizing feedforward artificial neural network architecture," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 3, pp. 365–382, 2007.

[19] S. Wang, X. Dong, and R. Sun, "Predicting saturates of sour vacuum gas oil using artificial neural networks and genetic algorithms," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4768–4771, 2010.

[20] K. L. Priddy and P. E. Keller, *Artificial Neural Networks: An Introduction*, SPIE—The International Society for Optical Engineering, Bellingham, Wash, USA, 2005.

[21] M. M. Alshihri, A. M. Azmy, and M. S. El-Bisy, "Neural networks for predicting compressive strength of structural light weight concrete," *Construction and Building Materials*, vol. 23, no. 6, pp. 2214–2219, 2009.

[22] D. Sovil, V. Kvanicka, and J. Pospichal, "Introduction to multilayer feed forward neural networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43–62, 1997.

[23] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[24] A. Blum, *Neural Networks in C++: An Object-Oriented Framework for Building Connectionist Systems*, John Wiley & Sons, New York, NY, USA, 1992.

[25] M. J. A. Berry and G. Linoff, *Data Mining Techniques*, John Wiley & Sons, New York, NY, USA, 1997.

[26] Z. Boger and H. Guterman, "Knowledge extraction from artificial neural network models," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. 3030–3035, Orlando, Fla, USA, October 1997.

[27] K. Swingler, *Applying Neural Networks: A Practical Guide*, Morgan Kaufman, San Francisco, Calif, USA, 2001.

[28] M. H. Sazli, "A brief review of feed-forward neural networks," *Communications, Faculty of Science, University of Ankara*, vol. 50, no. 1, pp. 11–17, 2006.

[29] S. Rajasekaran and G. A. V. Pai, *Neural Networks, Fuzzy Logic and Genetic Algorithms: Synthesis & Applications*, Prentice-Hall of India Private Limited, New Delhi, India, 2003.

[30] A. Y. Shamseldin, A. E. Nasr, and K. M. O'Connor, "Comparsion of different forms of the multi-layer feed-forward neural network method used for river flow forecasting," *Hydrology and Earth System Sciences*, vol. 6, no. 4, pp. 671–684, 2002.

[31] A. Mellit, S. A. Kalogirou, and M. Drif, "Application of neural networks and genetic algorithms for sizing of photovoltaic systems," *Renewable Energy*, vol. 35, no. 12, pp. 2881–2893, 2010.

[32] N. M. Razali and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP," in *Proceedings of the World Congress on Engineering*, vol. 2, pp. 1134–1139, 2011.