Available online at www.pelagiaresearchlibrary.com

Pelagia Research Library

Advances in Applied Science Research, 2013, 4(4):214-223

ISSN: 0976-8610
CODEN (USA): AASRFC

Artificial neural network for the modulus of rupture of concrete

Onwuka. O. David and Awodiji. T.G. Chioma

Civil Engineering Department, Federal University of Technology, Owerri, Imo State, Nigeria

ABSTRACT

In traditional method of concrete mix design, a normal concrete of required strength, can be achieved after several trials on mix proportion. This trial mix design approach has proven to be complex and difficult. This article demonstrates the applicability of artificial neural network (ANN) to the design of concrete of required modulus of rupture (MOR). The architecture of the neural network created is 4-20-1 (i.e. 4 input neurons, 20 neurons in the hidden layer and one output neuron). And, a feed forward back propagation learning algorithm was used for the training of the neural network. Sufficient set of mix proportions with their corresponding modulus of rupture, were generated experimentally and used for the training and testing of the neural network. The neural network toolbox of MATLAB software with TRAINGDM training function, LEARNGDM learning function and MSE regularization performance function, was utilized in creating the neural network. The results predicted by the network were in close agreement with corresponding experimental values. And a correlation coefficient of 0.9051 for all, shows that there is a linear relationship between the output and target during training.

Keywords: Artificial neural network.

INTRODUCTION

Concrete has been in use in construction for quite a long time now. It is the world's most utilized construction material and the need for infrastructural development in both the developing and developed countries has placed a great demand on it [8]. Its popularity as a construction material is due to the fact that it is made from commonly available ingredients and it can be tailored to functional requirements in a particular situation [10]. An exact determination of concrete mix proportions from tables or computer data is generally not possible. In consequence, all that is possible is to make an intelligent guess at the optimum relationship already established. Therefore, in order to obtain a satisfactory mix, not only are the proportions of available material estimated, but trial mixes are also made. These mixes are checked and adjustment in the mix proportions are made in the laboratory until a fully satisfied mix is obtained [7] .

An alternative method of designing concrete of required MOR has been developed using the artificial neural network (ANN) approach. The ANN approach uses mathematical formulations to model nervous system operations [9]. Many researchers have applied the ANN in making predictions. An ANN is configured for a specific application such as pattern recognition or data classification, through a learning process. In the past, considerable attention has been focused on the problem of applying neural network in diverse fields, including systems, fault diagnosis and controls. This is because ANNs are a good tool to model non-linear systems [13]. Many research activities have been carried out using supervised learning ANNs [9]. [5, 6] used the ANN in the prediction of global solar radiation of Uyo and Warri, Nigeria. [1] Used the ANN to predict ultrasonic velocities in binary oxide glasses while [2] used the ANN to predict the n-octanol/water partition coefficient of derivatives of the anti-HIV drug in quantum chemical calculations.

1.0 ARTIFICIAL NEURAL NETWORK

An artificial neural network is an information processing system that has certain performance characteristics in common with biological neural network [4]. It consists of a network of artificial neurons, which are processing units arranged in layers similar to the biological neurons in the brain. It developed as a generalization of mathematical models of human cognition or neural biology based on the assumption that;

(i) Information processing occur at many simple elements called neurons.
(ii) Signals are passed between neurons over connected links.
(iii) Each connected link has an associated weight which in a typical network multiples the signal transmitted.
(iv) Each neuron applies an activation function (usually non-linear) to its net input (sum of the weighted input signal) to determine its input signal.

In practice, artificial neural network can perform any computable function which a normal digital computer can handle.

Artificial neural network are specified by its architecture, training or learning algorithm, and activation function.

2.1 NETWORK ARCHITECTURE

The network architecture has to do with the arrangement of the neurons within the neural network. Fig 1 shows the architecture of the neural network adopted in the study. There are four neurons in the input layer corresponding to the mix proportions of water-cement ratio, cement, sand and granite chippings, and one neuron in the output layer, corresponding to the modulus of rupture of concrete. The main task here is to determine the number of neurons in the hidden layer. In order to determine the number of neurons in the hidden layer, lots of trials were carried out in the MATLAB neural network tool box environment. And finally, a network with twenty neurons in one hidden layer was selected and used in this study. Therefore, a configuration of 4 -20-1 was adopted for this network

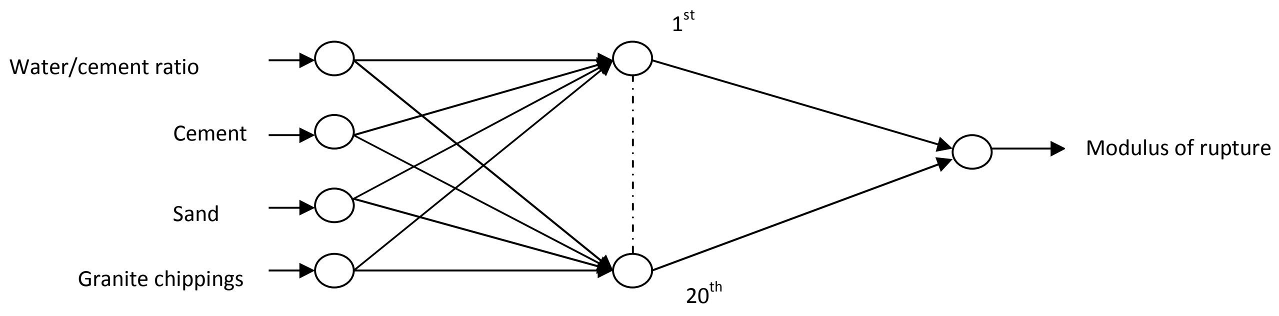

Fig 1: Architecture of the 3-layer neural network

2.2 LEARNING OF THE ANN

Artificial Neural Network has the ability to learn a given data and generalize the relationship between the input data (i.e. weights of the connection), even when the input data are incomplete or contain error. Like people, artificial neural network learns by example. It requires these learning examples as prior knowledge. These examples consist of the collections of input and output patterns, where the patterns are representative of patterns of activation.When the neural network is presented with examples of the computation to be performed, it learns the output that corresponds to the input specified. In order to ensure that a network learns, the input data should contain the specific information from which the desired output is derived. During the learning phase of the network, the same data is processed many times as the connection weights are refined. The following steps are involved in the learning phase:

(i) Feed forward of the training data.
(ii) Back propagation of the associated error.
(iii) Adjustment of the weights.

2.2.1 FEED FORWARD COMPUTATION

In feed forward computation, information flows in one (i.e. forward) direction. In general, each neuron in the network operates by summing up all the weighted inputs it received and then passes the result through a non-linear activation function. The operations of the different layers of neurons are as follows:

(a) Each input neuron, (X_i, i = 1,…,n) receives input signals, x_i and broadcasts this signal to all units in the layer above (i.e. hidden neurons).

(b) Each hidden neuron H_j(j = 1, 2,…,m) sums up its weighed input signals. The net input signal to the hidden unit, H_j is described as:

$$net_{pj} = \psi_{oj} + \sum_{i=1}^{n} w_{ij} I_i \qquad (1)$$

where ψ_{oj} = bias on hidden neurons j; w_{ij}= weight from neuron i (source) to neuron j (destination); I_i = input value from neuron i. And the output, h_j, of the hidden unit (H_j) as a function of its net input is given by:

$$h_j = f(net_{pj}) \qquad (2)$$

where f = sigmoid function = 1/[1 + Exp(-x)]

$$\text{But, } x = net_{pj} \qquad (3)$$

$$\text{Thus, } h_j = 1/ [1+ Exp(-net_{pj})] \qquad (4)$$

The signal given by Eqn. (4), is sent from the hidden neuron, H_j to all neurons in the output layer.

(c) Each output neuron, (O_k(k = 1, 2,…,p)) sums its weighted input signals. The net input to the output neuron (net_{pk}) is given as:

$$net_{pk} = w_{ok} + \sum_{j=1}^{m} w_{jk} h_j \qquad (5)$$

Where w_{ok}= bias on output neuron, O_k; w_{jk}= weights from neuron j to neuron k.

The output signal from neuron, O_k, is obtained by applying its activation function to its weighted input signals

$$o_k = f(net_{pk}) \qquad (6)$$

Where f = sigmoid (activation) function

$$\text{Therefore, } o_k = 1/[1 + exp(-net_{pk})] \qquad (7)$$

The set of calculation that results in obtaining the output state of the network is carried out the same way for both training and testing phases. However, the test mode just involves presenting the input units and calculating the resulting output in a single forward pass.

2.2.2 BACK PROPAGATION OF ERROR

The training algorithm adopted in this work, is the gradient descent with momentum constant. The back propagation computation involves the use of the chain rule of calculus. According to [11], error measure, E_p, known as the mean square error, (mse) is defined as

$$E_p = {}^1/_n \sum_{k=1}^{n} (f_k \quad k = 1 \qquad (8)$$

Where
f_k = target (desired) value of O_k, output neuron
o_k = actual output obtained from O_k, output neuron
n = the n^{th} values of the outputs obtained from O_k, output neuron

But the change in weight, $\Delta w_{jk,}$ from hidden unit, H_j to output unit, O_k , is directly proportional to the negative gradient of the error function, E_p with respect to the weights.

$$\Delta w_{jk} \ \alpha \ (-\partial Ep/\partial w_{jk}) \qquad (9)$$

$$\text{Therefore,} \quad \Delta w_{jk} = -\alpha E_p/w_{jk} \qquad (10)$$

Where α = learning rate

The Eqn. (10) is known as the delta rule.

$$\text{But, } \Delta w_{jk} = -\alpha \partial_k h_j \qquad (11)$$

Where ∂_k = portion of error correction weight for w_{jk} (i.e. the error at the output unit, O_k)

$h_j = \partial net_{pk} / \partial w_{jk}$

$$net_{pk} = w_{ok} + \sum_{j=1}^{m} w_{jk} h_j = \text{ net output to the output unit} \qquad (12)$$

Where, w_{ok}= bias on output neuron, k; w_{jk}= weights from neuron j to neuron k.

Using product rule, ∂_k, is derived as;

$$\partial_k = (f_k - o_k) f'(net_{pk}) \qquad (13)$$

In order to improve generalization in this work, the performance function was modified by adding a term that consists of the mean of the sum of squares of the network weights and biases (msw). Therefore, the mean square error with regularization 'msereg' can be defined as:

$$msereg = \kappa mse + (1 - \kappa) msw \qquad (14)$$

where κ = performance ratio;

$$msw = 1/n \sum_{i=1}^{n} w_j^2 = \text{ mean of the squares of the network weights and biases} \qquad (15)$$

2.2.3 TRAINING ALGORITHM
This is the procedure for modifying the weights on the connection links in a neural network. There are many algorithms for training neural networks; most of them can be viewed as a straight forward application of optimization theory and statistical estimation. They include: Back propagation, conjugate gradient descent, levenberg-marquardt, simulated annealing, evolutionary computation methods, particle swarm optimization and other swarm intelligence techniques. But the best known example of a neural network training algorithm, is the back propagation [4]. This is because it is the easiest algorithm to understand.

Back propagation (of error) or the generalized delta rule, is simply a gradient descent method to minimize the total squared error of the output computed by the network. The gradient vector of the error surface, is calculated. This vector points along the line of the steepest descent from the current point. So we know that if we move along it a 'short' distance, we will decrease the error. A sequence of such moves will find a minimum of some sort. A very general nature of the back propagation training method means that a back propagation network (a multilayer feed forward, network trained by back propagation) can be used to solve problems in many areas. Applications using back propagation and its variations can be found in virtually every field that uses neural network for problems that involves mapping a given set of inputs to a specific set of target outputs (i.e. networks that use supervised learning). As is the case with most neural networks, the aim is to train the network to respond correctly to input patterns that are used for training (memorization) and have the ability to give reasonable (good) responses to input that is similar, but not identical, to that used in training (generalization).

Numerous variations of back propagation have been developed to improve the speed of the training process. Example of this variation can be seen in the work of [12]. In their work, they combined the features of the feed forward neural networks and the genetic algorithms to develop a hybrid neural network model for the design of concrete beams. The effect of this hybridization of neural networks resulted to considerable improved efficiency (i.e. enhanced speed of training) of the network. Back propagation algorithm was used in training the neural network.

2.2.4 ACTIVATION FUNCTION.

An activation function is a function that transforms the net input into a neuron to a value the neuron transmits. The basic operation of an artificial neuron involves summing its weighted input signal and applying an output or activation function. For the input units, this function is the Identity function. Typically, the same activation is used for all neurons in any particular layer of a neural network, although this is not required. In most cases, a non-linear activation function is used. In order to achieve the advantages of multilayer nets, compared with the limited capabilities of single layer network, non-linear functions are required.

MATERIALS AND METHODS

3.1 MATERIALS

Concrete used in this study was composed of cement, sharp river sand, granite chippings and water. Dangote cement, a brand of ordinary portland cement was used in this work. It conforms to the requirement of [3]. The sharp river sand was obtained from Otamiri River in Owerri, Imo State of Nigeria. It fell into the zone 3.The granite chippings were obtained from Ishiagu quarry site in Ebonyi State of Nigeria. And, the water used was portable water obtained from municipal water supply. Somesof the mix proportions of these constituent materials of concrete are shown in Table 1.

Table 1: Mix proportions of constituent materials used in producing the concrete beams

S/N0	Mix proportion	Water-cement ratio	Mixture Label	Water (Kg)	Cement (Kg)	Sand (Kg)	Granite (Kg)
1.	1 : 2: 4	0.55	S1	2.56	4.63	9.26	18.51
2.	1 : 2.5 : 6	0.50	S2	1.71	3.41	8.53	20.46
3.	1 : 1.5 : 3.5	0.60	S3	3.24	5.40	8.10	18.90
4.	1 : 2.25 : 5	0.52	S4	2.06	3.93	8.84	19.64
5.	1 : 2.5 : 4.75	0.49	S5	1.96	3.93	9.82	18.65
6.	1 : 1.75 : 3.75	0.58	S6	2.87	4.98	8.72	18.69
7.	1 : 2.75 : 5.75	0.48	S7	1.62	3.41	9.38	19.61
8.	1 : 2 : 4.75	0.55	S8	2.30	4.18	8.36	19.86
9.	1 : 2.25 : 4.5	0.52	S9	2.19	4.18	9.41	18.81
10.	1 : 2.375 : 4.875	0.51	S10	2.01	3.93	9.33	19.14
11.	1 : 2.25 : 4.75	0.53	S11	2.13	4.05	9.11	19.24
12.	1 : 2.125 : 4.625	0.54	S12	2.25	4.18	8.88	19.33
13.	1 : 1.75 : 4.125	0.58	S13	2.71	4.71	8.25	19.44
14.	1 : 2.25 : 4.375	0.52	S14	2.23	4.25	9.56	18.59

3.2 METHODS

The first part of the test was the laboratory testing in flexure of the 150x150x600mm prototype concrete beam specimens prepared from fourteen different concrete mixtures. These tests were conducted at the age of 28 days after curing the specimens in open water tanks. The two beams prepared from each mix, were tested and their average values recorded as the results

The second part was the development of the neural network. The historical data consisting of the four hundred ratios of concrete (inputs) and their corresponding four hundred modulus of rupture (outputs), were stored in a matrix form and then processed using Matlab software with the “mapminmax” and “removeconstantrows” processing functions. The processing functions were used for preventing the network from being saturated. Subsequently, a feed forward neural network with four input neurons, twenty hidden neurons and one output neuron, was created using the network command“newff”. Thereafter, the neural network was configured by specifying the network parameters such as activation functions for the different layers and training algorithm. The configuration commands are “net.inputs(1).processFcn”and“net.outputs(2).processFcns”. Then the weights and biases of connection paths between two processing neurons in the network, were initialized and reinitialized using the command “net = init(net)”. Finally, the network was trained for pattern classification and validated. The process of training required tuning the values of the weights and biases in order to optimize the network performance. Training was implemented using the batch mode. In all, four hundred data pair comprising of concrete mix proportions and moduli of rupture were used in the training of the network.

In validating the network, the performance of the network was checked in order to determine if any changes need to be made to the training process. The experimental data were used for this purpose.

RESULTS AND DISCUSSION

The experimental results i.e. modulus of rupture are presented in column 2 of Table 2. And the results predicted by the neural network, are given in column 3 of the same Table 2. A comparison of the experimental results and the neural network results show that the maximum percentage difference is 3.276, which is negligible. From Figs 2 and 3, it will also be seen that there is a close agreement between the experimental results and the neural network results.

Table 2: Comparison of Experimental results vs. Neural Network Predictions vs. Percentage Errors

Mix Label	Experimental Results(Mpa)	Neural Network Prediction (Mpa)	error	% error
S1	6.00	6.0101	-0.1101	1.866
S2	5.30	5.1904	0.1096	2.068
S3	6.22	6.2761	-0.0561	0.9
S4	5.20	5.2687	-0.0687	1.321
S5	4.60	4.6898	-0.0898	1.952
S6	6.30	6.3482	-0.0482	0.765
S7	4.89	4.9267	-0.0367	0.751
S8	6.20	6.4031	-0.2031	3.276
S9	5.12	5.0453	0.0747	1.459
S10	4.70	4.7649	-0.0649	1.381
S11	5.20	5.2640	-0.0640	1.231
S12	5.70	5.7218	-0.0218	0.382
S13	6.15	6.2138	-0.0638	1.0374
S14	5.00	5.0260	-0.0260	0.521

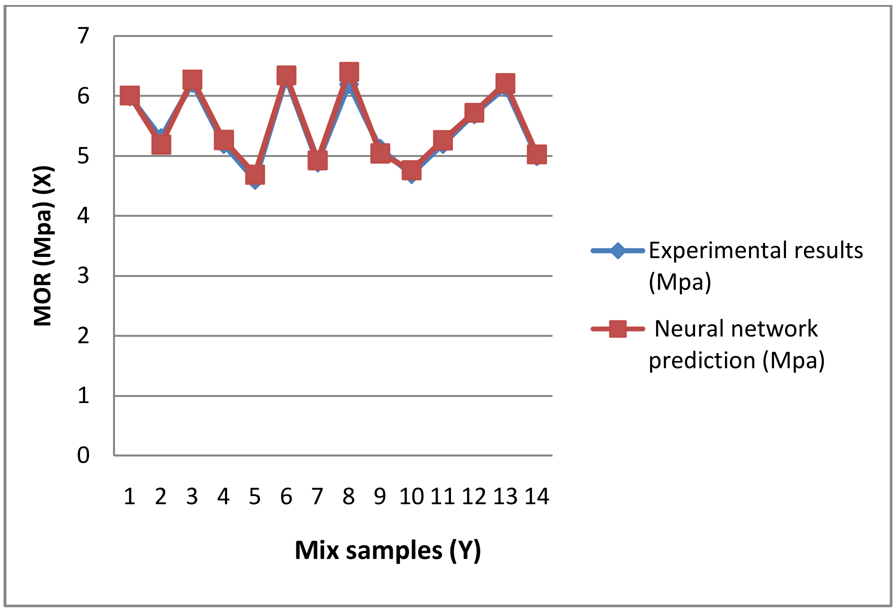

Fig 2: Line graph comparing experimental results with neural network predictions

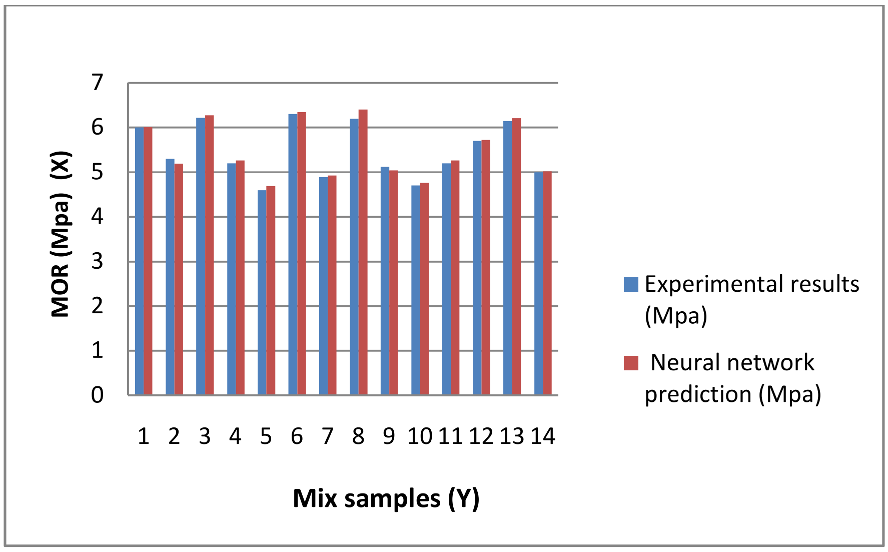

Fig 3: Bar chart comparing experimental results with neural network predictions

4.1 VALIDATION OF NETWORK PERFORMANCE

The validation of network performance is presented in Fig 4. The correlation coefficient R is 0.905, which shows that there is a linear relationship between the outputs and targets.

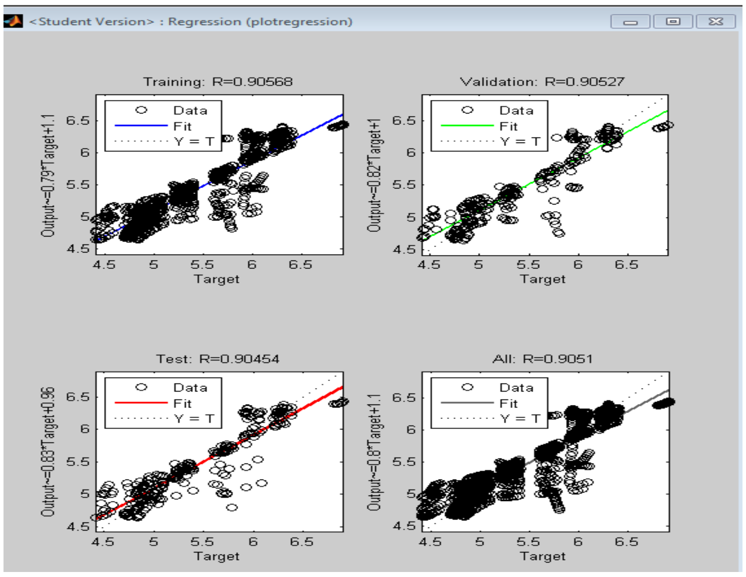

Fig 4: Validation of network performance

Fig 5 shows the msereg error convergence history. The mean square error with regularization (msereg) for the best validation performance occurred at 11.56 for 1000 epoch. In Fig 6, the gradient at 1000 epoch is 0.4779. There was no validation check at 1000 epoch as the training of the network stopped when the maximum epoch was reached.

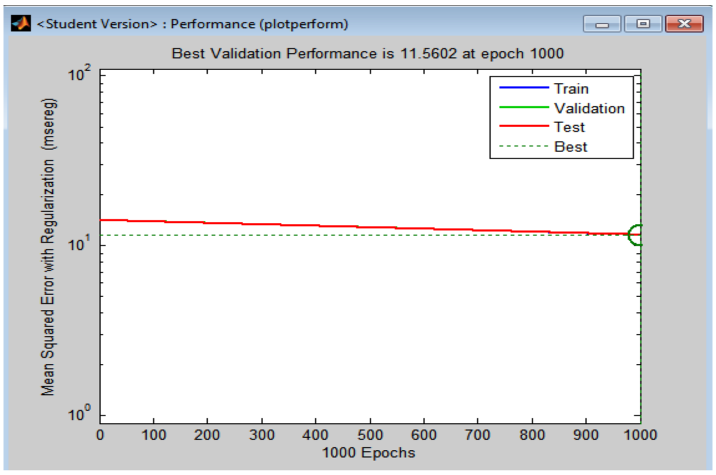

Fig 5: Mean square error convergence history

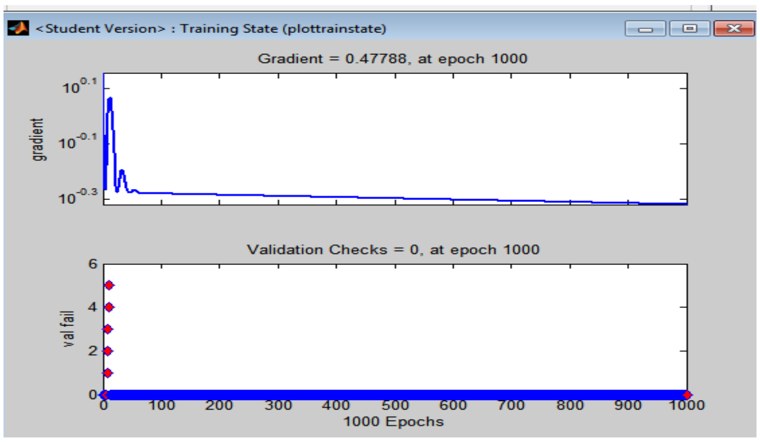

Fig 6: Plot of network gradient vs. number of epoch

4.2 TEST OF ADEQUACY OF THE NEURAL NETWORK
The students T-test was used in testing the adequacy of the neural network at a significance level of 0.05. The t-test computations are presented in Table 3.

Table 3: Student T-test computation

SN	YE	YM	Di =YM-YE	DA - Di	$(DA - Di)^2$
S1	6	6.0101	0.0101	0.053392857	0.002851
S2	5.3	5.1904	-0.1096	0.173092857	0.029961
S3	5.9	6.2761	0.3761	-0.312607143	0.097723
S4	5.2	5.2687	0.0687	-0.005207143	2.71E-05
S5	4.6	4.6898	0.0898	-0.026307143	0.000692
S6	6.3	6.3482	0.0482	0.015292857	0.000234
S7	4.89	4.9267	0.0367	0.026792857	0.000718
S8	6.2	6.4031	0.2031	-0.139607143	0.01949
S9	5.12	5.0453	-0.0747	0.138192857	0.019097
S10	4.7	4.7649	0.0649	-0.001407143	1.98E-06
S11	5.2	5.264	0.064	-0.000507143	2.57E-07
S12	5.7	5.7218	0.0218	0.041692857	0.001738
S13	6.15	6.2138	0.0638	-0.000307143	9.43E-08
S14	5	5.026	0.026	0.037492857	0.001406

$\sum D_i = 0.8889$
$\sum(D_A - D_i) = 0.17394$
$D_A = \sum D_i / N = 0.063492857$
$S^2 = \sum (D_A - D_i) / (N - 1) = 0.01338$
$S = \sqrt{S^2} = 0.11567$
$T_{cal} = D_A * (N) \wedge 0.5 / S = 2.05381$

From standard statistical table, the T-value is given as, $T_{(0.05, 13)} = 2.16$

Since, the calculated T-value, T_{cal} is less than the critical T-value from standard statistical table, (i.e. T = 2.16), the null (H_o) hypothesis is accepted as there is no significant difference between the neural network results and the experimental results. This affirms that the results from the neural network are reliable and so the neural network can be used to predict the 28th day flexural strength of concrete at 95% confidence level.

CONCLUSION

In this study, the historical results of 400 samples were applied to generate an artificial neural network (ANN) to predict the modulus of rupture of concrete. The concrete was made of different mix proportions of cement, sand, granite chippings and water. The outcome of the created ANN was compared with the results of the experimental work. The selected network and its parameters were;

i. The water-cement ratio, cement, sharp sand and granite were the inputs while the Modulus of Rupture of Concrete was the output of the network.
ii. The architecture of the selected network was 4 –20 -1.
iii. A total of 400 data were used for training and 67 data were used for verifying and testing respectively.
iv. The ultimate network to predict the Modulus of Rupture was the feed-forward back-propagation neural network, in which the training and transmission function were TRAINGDM and TANSIG respectively.
v. The outcome results of the created network were close to the results of the experimental effort.
vi. Momentum constant and the learning rate were kept constant at 0.9 and 0.2 respectively.
vii. The selected ANN can be used to predict the Modulus of Rupture of Concrete with minimum error below 4% and the maximum correlation coefficient close to 1.

REFERENCES

[1]Arulmozhi KT, Sheelarani R. *Advances in Applied Science Research*, **2011**,2(4):147-155.
[2] Bayat Z, Emadiyan M. *Der Chemica Sinica*, **2011**, 2(6):341-350.
[3]BS 12*, Specification for ordinary and rapidly hardening Portland cement*, **1978**. BSI- London.
[4]Fausett L. *Fundamentals of Neural Network*, New York, Prentice- Hall, Inc. **1994**,pp 289-295.
[5] Ibeh GF, Agbo G, Rabia S. *Advance in Applied Science Research*, **2012**, 3(1):619-624.
[6]Ibeh GF, Agbo GA, Agbo PE, Ali PA. *Advance in Applied Science Research*,**2012**, 3 (1):130-134
[7]Neville AM. *Properties of Concrete*,4th Edition, Dorling Kindersley (India) Pvt. Ltd. **2006**, pp 727-728.
[8] Osei DY, Jackson EN. *Advances in applied science research*, **2013**, 3(6):3658– 3662.
[9]Razavi SV, Jumaat MZ, Ahmed, HE. *International Journal of the Physical Sciences*, **2011**; 6(6), pp 1325 – 1331.
[10]RishiG.*Concrete Mix Design using Artificial Neural Network*. Masterdegree thesis, Civil Engineering Department, Thapar Institute of Engineering & Technology, India, **2003**.
[11]Rumelhart DE, Hinton GE, Williams R.*Learning Internal representations by back propagation error*, **1986**. Nature, 323:pp533 – 536.
[12]Sudarsana RH, Ramesh Babu B.*Hybrid Neural Network model for the design of Beam subjected to Bending and Shear*,Sadhana India. **2007**; 32(5), pp 577-586.
[13] YehIC. *Journals of Construction Engineering and Management*. **1998**, pp 374 -380.