Pattern Recognition for Data Retrieval using Artificial Neural Network

Jyoti R Munavalli*
Associate Professor, Department of Electronics and Communication Engineering, BNM Institute of Technology, Bangalore, India
jyotirmunavalli@bnmit.in

Rashmi R Deshpande
Research Scholar, Visvesvaraya Technological University, Karnataka, India.
rrdeshpande01@rediffmail.com

Abstract: *Data retrieval is an important aspects of data management. In this paper, we design an ANN to recognize the learned patterns. We use three-layer feed forward network for training of patterns (bitmap data). We implement two kinds of recognition: forced recognition and custom specific recognition. The ANN model developed recognizes the pattern even if there is variation in the applied test patterns from the learned/trained patterns. In particular, we discuss the fault tolerance offered by neural network. The characteristic of fault tolerance depends upon the type of distribution taken from random number. The conventional network is based on the concept of memorization whereas the neural network is based on the concept of generalization.*

Keywords: **Artificial Neural Network, Data Retrieval, Pattern Recognition, Fault Tolerance**

1. INTRODUCTION

An Artificial Neural Network (ANN) is a computing system, inspired by the biological nervous systems, such as the brain. The structure of information processing is the important aspect of ANN. It consists of huge number of neurons that are highly interconnected processing elements and these work in parallel to solve particular problem. ANNs are configured for a specific application, such as voice recognition, intelligent searching, financial forecasting, pattern recognition, signal processing and classification etc., through a process called "learning". ANNs has a different approach to problem solving compared to that conventional computing systems. Conventional computers follow a set of rules (rule-based approach)., but neural networks are not so. Neural nets process the information similar to human brain. The nets learn just like biological systems where the synaptic connections between the neurons are adjusted [1].

Artificial Intelligence (Machine Learning, Deep Learning and ANN) have been applied to a number of real-world complex problems. Conventional technologies are algorithm based whereas ANN is based on the abstraction of biological brain. ANNs are propitious to problems in which people are good at solving. At first, the neural net has to be designed. Then the learning process and the weights are used to identify the faults. The limitations of the current technology like intelligence, fault detection and tolerance and the processing speed, leads to implementation of ANN [2, 3]. As neural nets are parallel and distributed systems, processing is fast and they are adaptable. Because of parallel processing, they are capable of providing excellent fault tolerance [4].

The demand for high performance processor and computing algorithms will be ever increasing. In today's world, data plays a vital role in lot of applications. Huge amount of data is been generated by various applications through IoT. One facet of data is its collection and analysis and another is its storage and retrieval. Data gets corrupted at various stages from host to storage medium and retrieving this data is a challenging task.

Neural nets are appropriate for applications like classification and recognition of patterns. For this, simulation is the best option. Post simulation, physical implementation should be followed. The important parameters for the quality of solution are reliability and fault tolerance.

In this paper, we design an ANN for pattern recognition where we propose forced recognition and custom recognition. In this, we train the ANN and the input of bitmap image of characters (numbers and alphabets) is processed. Based on the proposed recognition model, the character is recognized. Though the data is corrupted, the proposed model recognizes the character. The paper is organized as follows: Section 2 presents the materials and methods, in which we describe the overview of the ANN, the proposed method and Pattern Recognition model. Section 3 presents the Results from the simulation, and Discussion where we discuss in detail about the fault tolerance and Section 4 is the conclusion of the paper.

2. MATERIALS AND METHODS

2.1. Overview of ANN

Neural network simulations depict the neurons in the brain. The way the brain processes the information still remains unknown. We briefly explain the basics of the neural nets with neuron feature and interconnections.

Model of an Artificial neuron

Artificial neuron model has three elements as shown in Fig. 1.

1. A set of synapses or connection links that are characterized by a weight (or strength) of its own. Input signal 'X_i' is multiplied with the weight value, 'W_{ij}'.
2. A summing junction to add the input signals that are multiplied by their respective weights $(net)_i$
3. An activation function that defines the output and helps the network learn complex patterns in the data. It limits the amplitude of the output signal (O_i) to some finite value [1, 2].

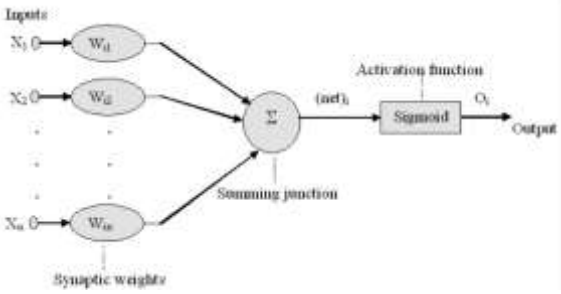


Fig. 1 Model of an Artificial Neuron

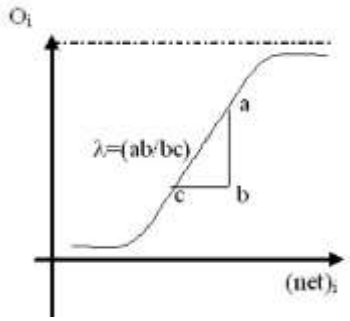


Fig. 2 Sigmoid Function

Activation function

The behavior of ANN depends on both the connection weights and activation function defined for the nodes. *'Sigmoid'* function is the most popular activation function used in ANN. It is an increasing function which is limited by a soft threshold. It balances between the linear and non-linear characteristics. Sigmoid graph is shown in Fig. 2 and is represented as in Equation1.

$$O_i = 1/ (1+\exp (-\lambda. net_i)) \quad (1)$$

where 'λ' is the slope of the sigmoid function near to origin, variation in which gives sigmoid functions of different slopes. As the slope parameter approaches ∞, the function simply becomes a threshold function, which assumes the value of a 0 or 1. But, the sigmoid function assumes a continuous range of values from 0 to1, if unipolar and –1 to +1 if bipolar. Because the sigmoid function is continuous, it is differentiable which is an

important requirement in ANN training.

2.2. Proposed method

The proposed method is as shown in Fig. 3 flowchart.

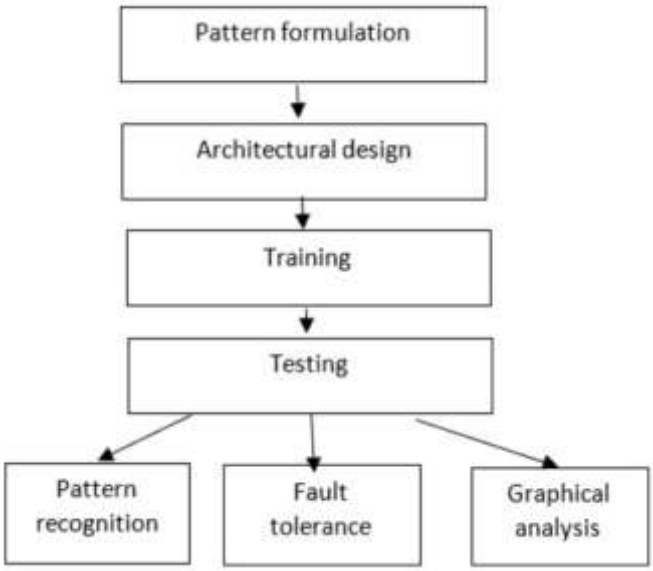
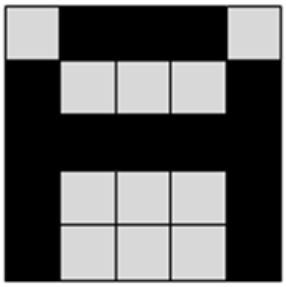


Fig. 3. Flowchart of proposed method

A. Pattern formulation*:*

Patterns are converted to binary format using grid transformation. This format is used in digital application so this format is selected. Any pattern's segment, if present in the grid is taken as 1 otherwise 0. This is shown in the example figure given below.



01111101001010010100101111

Fig. 4. Input pattern and its bitmap form

B. Architecture Design

Two types of neural nets are feed-forward and feedback networks. Feedforward neural nets are unidirectional. That is data flows in one way. From input to output. Feedback neural nets are the nets that have feedback connection, that is a part of the output is feedback to the input. Feedback networks are very powerful, have memory and can get extremely complicated [6].

In this study, we propose to use feed forward neural net. ANN is also known to as "parallel distributed processing system". Here all the nodes in the same layer process the data independently and simultaneously. The output from each node is distributed to all the nodes of next layer The weights associated to the connections are multiplied. That is either the value is increased or decreased based on the values of the weights. These weights are updated using learning algorithm till the desired output is obtained. In this study, we use the feedforward architecture that has 25 nodes for input layer, 10 nodes for hidden layer and 1 node for output layer. (View Fig. 5)

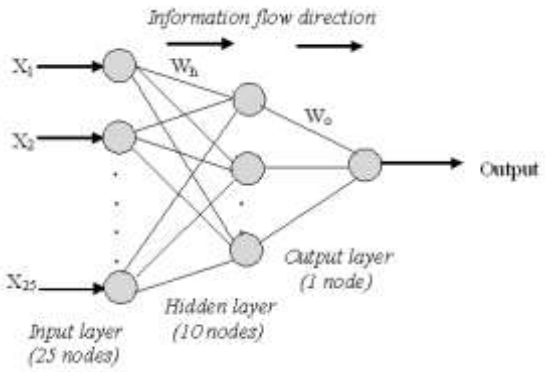


Fig. 5 Proposed feed forward ANN architecture

Now, we explain the process flow. (Fig. 6) Pattern of interest after passing through the grid transformation gets converted into a binary format. The connection strength from input layer to the hidden layers are initially taken to be random values, so the co-relation between the connection strength and input pattern is initially minimized (or can say zero co-relation). The two distributions through which we can fetch the random numbers from MATLAB library, one by means of Gaussian (Normal) distribution and other by means of Uniform distribution.

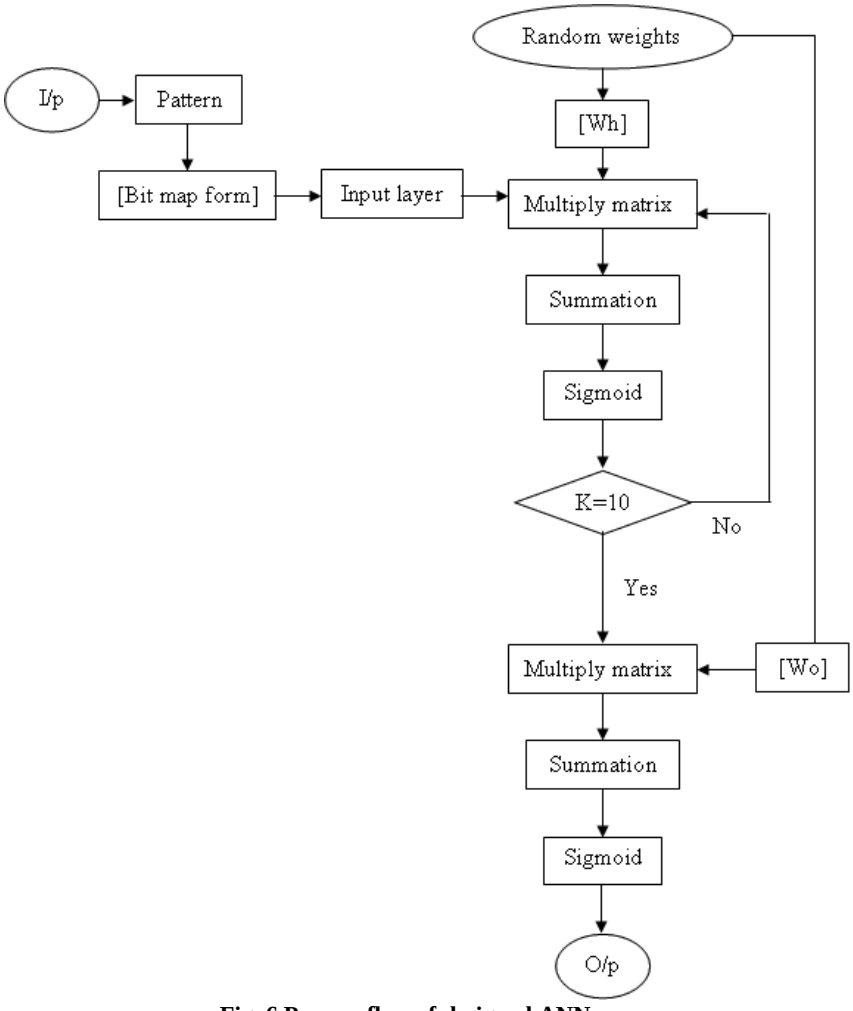


Fig. 6 Process flow of designed ANN

Here, Gaussian distribution is selected as it offers high fault tolerance capability compared to uniform distribution. Initially, input data is multiplied with the weights and the result is fed to all the nodes of the hidden layer. The active hidden layer performs accumulation and mapping operation on these inputs. The mapping of the accumulated result is required so that it could remain a bound number.

Again, the output of hidden layer nodes is multiplied with the random weights selected for the outer layer. Output node performs accumulation and mapping operations. One such complete processing from input node to output node is referred as epoch The co-relation is checked using the error function. If the error ≤ threshold, the learning rule will adjust the weights in right direction to reduce the error. The adjusted weights will become the new weights next epoch and this repeats till error value reaches the required criteria.

C. Training/ Learning of ANN

A neural net can be trained off-line. In that case it will have different learning phase and operation phase. It can be trained online, in that case has learning and operating phase at the same time. Generally, it is observed that supervised learning is performed off-line and unsupervised learning is performed on-line.

For a neural network to perform a specific task, selection of the units and the way they are connected to one another is very important. The weights on the connections must be set appropriately. The interconnections determine whether it is possible for one unit to influence another [5, 7]. The weights indicate how much a unit is influenced. A three-layer neural net can be trained to execute a particular task by the following procedure:

- The network with training set that consist of an activity pattern for the input units together with the desired pattern of activities for the output units.
- We compare the actual output of the network matches the desired output.
- Weights of each connection are modified or updated in order to produces a better approximation of the desired output.

Selection of learning algorithm

This aim of this study is to match the pattern so we make use of a Supervised and Generalized Delta Rule (GDR) used in so called Error Back Propagation Algorithm (EBPA). It is the most popular of the supervised learning techniques used to train a feed forward multi-layered artificial neural network The EBPA uses gradient descent to achieve training by error correction, where network weights are adjusted to minimize error based on a measure of the difference between desired and actual feed forward network output. Desired input/output behavior is given in the training set where the input & the target values {i, t} are predefined.

In 'Error Back Propagation', the error signal generated at the output node is propagated backwards. The algorithm tries to minimize an *error function* for that it starts from the output layer propagating towards the input layer. We require error function to adjust the weights in a proper direction, as to provide the maximum co-relation w.r.t the inputs, so that the network can converge towards a state that allows all the training patterns to be encoded, so this algorithm opted for training ANN in this paper.

D. Generalized Delta Rule

The *Error signal* **'e'** at the output of an output node neuron (network output) is defined by,

$$e=(T_i-O_i) \quad (2)$$

where T_i and O_i denote respectively the desired (target) and the actual network output. The GDR tries to minimize the scalar *Error function* **E**, defined as the *mean square of an Error* signal **e** (because learning rule demands a differentiable function) given by,

$$E=[(T_i-O_i)^2]/2 \quad (3)$$

Gradient of *Error function* **E** is then computed and is denoted as ∇E**,** given by,

$$\nabla E = \partial E/\partial W_{ij} \quad (4)$$

where W_{ij} denotes the strength of interconnection from node **i** to node **j**.

The Weight adjustment ΔW is then set proportional to ∇E, hence given by

$$\Delta W= - \eta * \nabla E \quad (5)$$

The negative of Gradient **E** (∇E) means the rate of decrement is maximum in negative direction, which implies decrement in error signal, thus serving the goal of training ANN. 'η' is the 'learning constant'. The effectiveness and convergence of the GDR depends significantly on the value of learning constant 'η'. However, the optimum value of η depends on the problem being solved and there is no single learning constant value suitable for different training cases.

From chain rule, we obtain

$$\partial E/\partial W_{ij} = \partial E/\partial O_i * \partial O_i/\partial W_{ij} \quad (6)$$

From equation (3), we obtain

$$\partial E/\partial O_i = -(T_i - Oi) \quad (7)$$

$\partial O/\partial W_{ij}$ gives the how the individual weights are going to affect the network output. This affect is calculated by taking *Slope*.

$$\text{i.e., Slope} = \partial O_i/\partial W_{ij} \quad (8)$$

Hence from Equations (6), (7) & (8) we arrive at

$$\partial E/\partial W_{ij} = -(T_i - O_i) * \text{Slope} \quad (9)$$

Therefore, from equations (5) & (9)

$$\Delta W = \eta * (Ti - Oi) * \text{Slope} \quad (10)$$

If error is high, it means the current weights are very far from the required value and hence a large change in weights is needed. If the error value is less meaning the current weights are near to the required value, so the change in weights required for the next iteration is also small. Therefore, change in weight is proportional to error:

$$\Delta W \propto (\text{Error}) \quad (11)$$

Each and every weight has a role in getting the output, but in a different proportion, where the hidden layer weights (W_h) have less effect on the output as compared to the outer layer weights (W_o), which can be justified. So, individual weights have to be adjusted in accordance to their affect at the output in order to minimize the error. Hence, we can say,

$$\Delta W \propto (\text{Slope}) \quad (12)$$

Derivations for ΔW

We have

$$\Delta W = \eta * (\text{Error}) * \text{Slope}$$

$$= \eta * (T_i - O_i) * \text{Slope}$$

From Equations (11) and (12)

<u>ΔW for output layer</u> (ΔW_o)

Slope = $\partial O_i/\partial W_{o1}$

$O_i \rightarrow f(\text{net})$

$\text{net} \rightarrow f(W_o)$

From chain rule,

$$\partial O_i/\partial W_{o1} = \partial O_i/\partial \text{net} * \partial \text{net}/\partial W_{o1}$$
$$= f`(S) * \partial[X_1W_{o1} + X_2W_{o2} + \ldots]/\partial W_{o1}$$
$$= f`(S) * X_1$$
$$= \text{Slope}$$

In general

$$\therefore \Delta W_o = \eta * (T_i - O_i) * f`(S) * X_i \quad (13)$$

Hence

$$W_{new} = W_{old} + \Delta W \quad \text{for output layer} \tag{14}$$

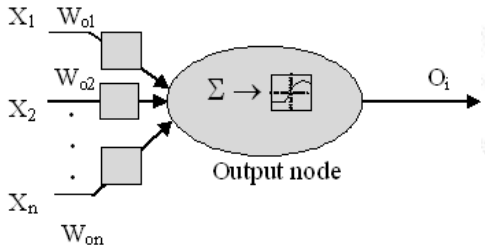


Fig. 7(a) Weight adjustment for output layer

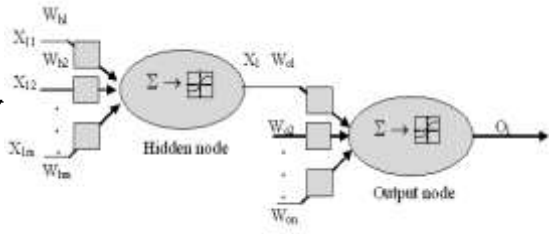


Fig. 7 (b) Weight adjustment for hidden layer

ΔW for hidden layer (ΔW_h)

Slope = $\partial O_i / \partial W_{h1}$

$O_i \rightarrow f\,(net)_o$

$(net)_o \rightarrow f\,(X_1)$

From chain rule, $\partial O_i / \partial W_{h1}$ $= \partial O_i / \partial (net)_o * \partial (net)_o / \partial X_1 * \partial X_1 / \partial W_{h1}$

$= f\,`_o\,(S) * W_{o1} * \partial [X_1]/W_{h1}$

$\partial [X_1]/W_{h1} = \partial [X_1] / \partial (net)_h * \partial (net)_h / \partial W_{h1}$

$= f\,`_h\,(S) * X_{11}$

$\therefore \partial O_i / \partial W_{h1} = f\,`_o\,(S) * W_{o1} * f\,`_h\,(S) * X_{11}$

$= (Slope)_h * (Slope)_o$

In general,

$$\therefore \Delta Wh = \eta * (T_i - O_i) * f\,`_o\,(S) * f\,`_h\,(S) * W_{oi} * X_i \tag{15}$$

Hence,

$$W_{new} = W_{old} + \Delta W_h \quad \text{for hidden layer} \tag{16}$$

2.3. Pattern Recognition

Decision about the pattern has to be taken only in the domain of learned patterns using maximum correlation principle. Two different approaches of recognition have been created namely forced recognition and custom specified recognition. Test input is applied to same ANN architecture [8].

A. Forced Recognition

In forced recognition, decision is taken in favor of things which has been learned previously, using maximum correlation method. Trained weights W_h / W_o are applied as connection strength. With each set of $[(X_t)i, (W_h)\ t\ /(W_o)\ t]$, a value is obtained at the output, this value is the degree of co-relation between the input and the selected weight set. For all different set of weights, corresponding co-relation value is obtained. Final decision taken in favor of set of weights with which maximum co-relation occurs.

B. Custom Specified Recognition

In custom specified recognition, a true correlation value set is compared with new set of correlation value. Decision is defined in terms of precision required in recognition. For very high precision, a small threshold is required, so precision can be changed by just

adjusting the threshold value. Here decision criteria are defined as "If in correlation error ≤ threshold, then maximum correlation is applied to decide the patterns otherwise a new pattern is classified.

- Degree of maximum correlation is first calculated with each set of (X_t), (W_h) t / (W_o) t] and stored as shown in flow diagram.
- Any test input (X_t)i is applied.
- A threshold value has to be defined.

Degree of correlation value between each pair of [(X_t)i, (W_h) t /(W_o) t] is calculated and stored as shown in flow diagram. Now these $(DC)_2$ values are correspondingly compared with $(DC)_1$ value. If any error value doesn't satisfy the ≤ threshold criteria, a new pattern is announced otherwise among the values which satisfy the criteria, the pattern with minimum value is decided as the recognized pattern, because in this case correlation between the input pattern and the trained weight set will be maximum.

$(DC)_1 \rightarrow$ having actual correlation value for each [(X_t), (W_h) t /(Wo) t]
$(DC)_2 \rightarrow$ having correlation value for each [(X_t)i, (W_h) t /(W_o) t]

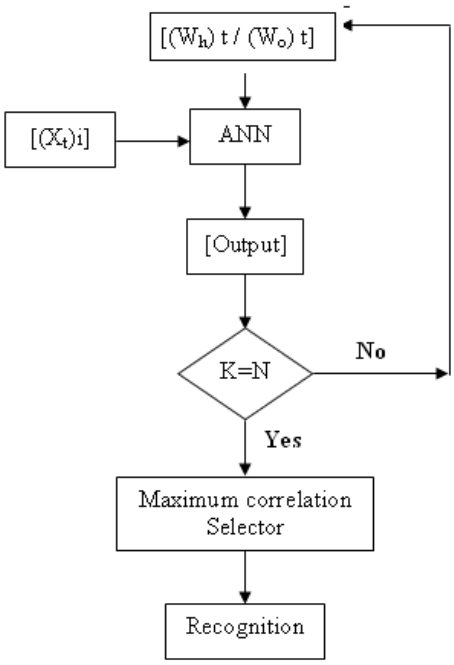
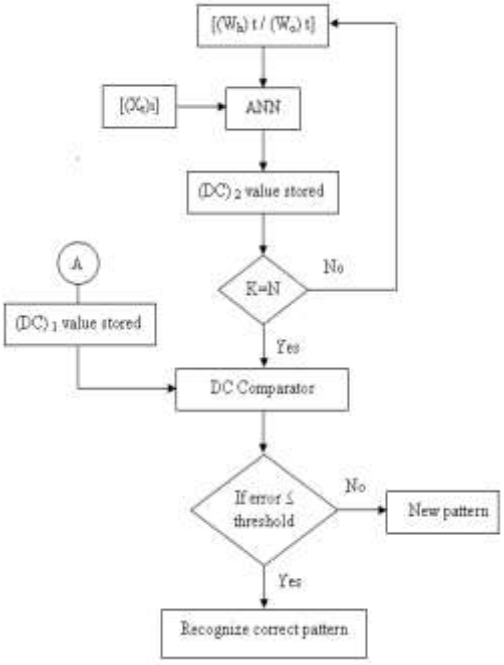


Fig. 8(a) Forced recognition flow chart

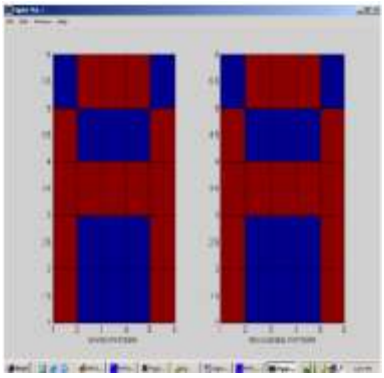


(b) Custom specific recognition flow chart

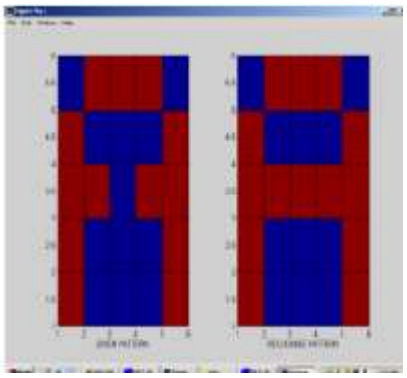
3. RESULTS AND DISCUSSION

The three-layer feed forward ANN can be simulated using the MATLAB. Some of the simulated results are shown here. The Fig. 9 shows the ability of this architecture in the fault tolerance and in pattern recognition.

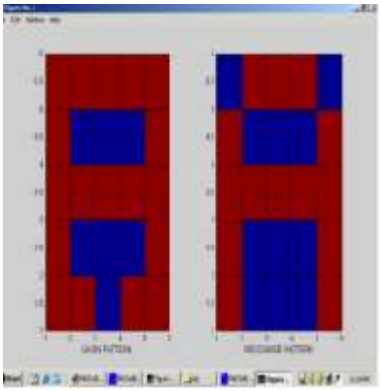
Wh plot supports the fundamental principle for pattern recognition i.e., maximum/minimum correlation. From the above graph, it's clear that learning rule maintains the fundamental form to achieve maximum correlation, i.e., with maximum value input, maximum weights are allocated and with minimum value input, minimum weights allocated. In the adjustment of hidden layer weights, outer layer weights play a very important role, in fact its appearance has proportionality relation in the formula. This is the reason that initially hidden layer random weights differ from outer layer random weights but after adjustment, W_h follows the same variation as W_o.



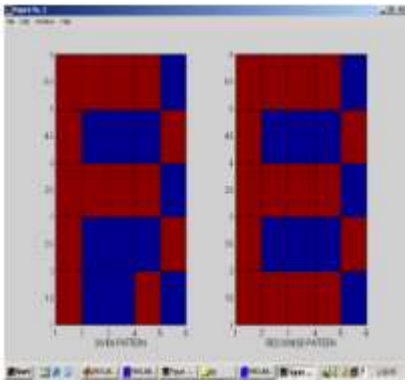
(a)Correct Recognition



(b)Correct Recognition



(c)Correct Recognition



(d)Incorrect Recognition

Fig. 9(a-d) Recognition/ Fault tolerance result for pattern 'A'

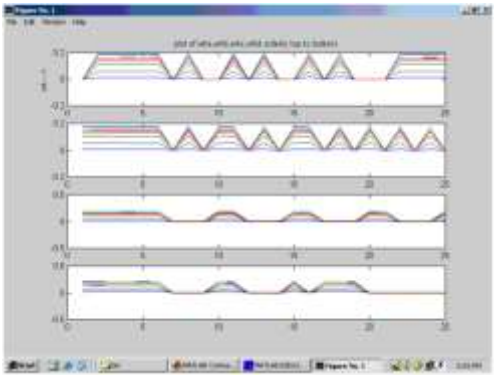
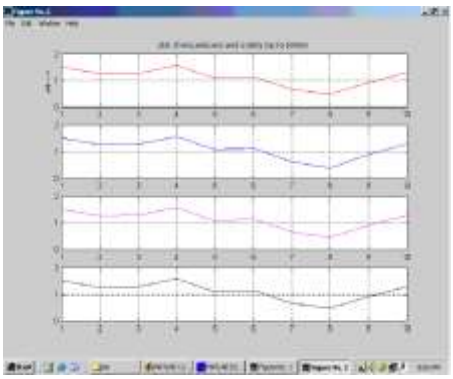
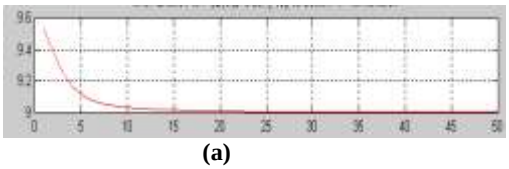


Fig. 10 (a) Input -Hidden layer weights (Wh) plot

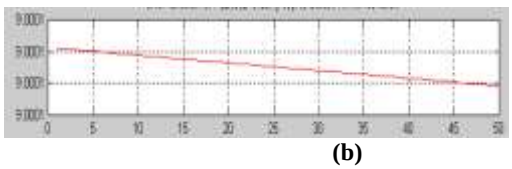


(b) Hidden-Output layer weights (Wo) plot

The Error function is considered as the figure of merit for learning rule, because weight adjustment takes place proportional to the error. From the graph (Fig.11), expected error in first iteration is highest because of initially assigned random weights or in other terms because of minimum correlation between weights and inputs, output is minimum. With iteration, adjustment approaches the required value and correlation value increases, so error starts to decrease. After number of iterations, error approaches a very small value and so adjustment of weight also followed by a very small amount because of maximally approaching correlation. This can be clearly observed in the above graph.



(a)



(b)

Fig.11. Error at (a)first and (b)last iteration for pattern A

The simulation model should be converted to physical implementation. For that reliability and fault tolerance are important parameters during design. Either extra hardware is requiring to be added to provide supplement path or there is a need to develop such an architecture that could handle fault without requiring any extra hardware. ANN have the potential of handling the faults because of parallel computations [9-14].

A. Place of faults inside the system

- ***Connection weights***

Fault tolerating capability is maximum irrespective of position of faults in W_h because the output isn't directly linked and the distribution of work is very high at this layer, but in case if faults occur in W_o, the tolerance depends upon where the fault occurs, since the distribution of work is very less.

There are two types of faults, viz., Open circuit type fault (0) and short circuit type fault (1). Former type of faults (0) is much sensitive than latter type of faults (1) or to say in other words fault tolerance is more for short circuit type of faults because the activation function (Sigmoid) provides maximum output value for such type of input faults and for faults of type 0 it provides half of the maximum output obtained for faults of type 1.

Because the number of hidden layer links with respect to output layer links is maximum, probability of fault occurrence is maximum in these links only and also tolerating capability is maximum. This means this layer is less sensitive compared to the output layer towards affecting the response.

- ***Processing elements (Active nodes)***

If we know the most sensitive nodes where sensitivity is defined in terms of affect at the output, it's a good information about fault tolerance capability of that system. It's an achievement of our project to provide knowledge of sensitivity of nodes before the start of experiment. This information is obtained by all the random values we selected as the output layer weights. The weight pattern at this layer remains same even after training irrespective of any kind of input patterns available. The hidden layer node having maximum random value at its output link will remain contain this maximum value & this node will become the most sensitive node after training. So, if any fault happens at this node, affect at the output will be maximum. This is enough information to reduce the sensitivity w.r.t faults in maximally affecting case of nodes.

For a solution, we have to create either a redundant node for a part of processing operation or we can provide a facility of global correction in terms of defining a node globally for a particular active node layer.

B. Place of faults outside the system

If any fault happens with the input itself, the same will be mapped onto by the hidden layer and because of the parallel distributed computation of ANN, maximum fault tolerance capability defined at hidden layer and hence the input faults are less sensitive towards affecting the final response.

C. Selection of random weights

Very less work has been done like what should be the nature of random weights from the point of view of fault tolerant capability. By experience, it is found that if the random values are taken from the Gaussian distribution [mean->0, variance->1], tolerance capability is more as compared to if data taken from uniform distribution. This is because the variation among data values is large in Gaussian distribution as compared to uniform distribution. So, if any faults happen at the low value weights, it can be more tolerable and the high value weights are much sensitive towards affecting the output whereas if data are from uniform distribution, since the variation among values is very less, fault in any data poses nearly equal affect at the output and is more

4. CONCLUSION

In this paper, the criteria selected for pattern recognition is the principle of maximum correlation, which for a particular set of input and weights is achieved by delta rule. Two different approaches of recognition have been created namely forced recognition and custom specified recognition. In forced recognition where decision must be taken in favor of things which has been learned previously, using just maximum correlation method. Where as in the latter case a true correlation value set is compared with new set of correlation value. Decision is defined in terms of precision required in recognition. For very high precision, a small threshold is required, so precision can be changed by just adjusting the threshold value. The result of pattern recognition is very satisfactory.

Analysis of fault tolerance for a system is a difficult task. All three possible places of fault inside the system, hidden layer weights W_h, outer layer weights Wo and combination of both has been experimented. Because Wo has more direct effect on output as compared to W_h, so sensitivity with respect to Wo is very high if any fault happens compared to W_h. But the probability of fault is high for W_h because there is more available space. Also, if fault occurs in the input, the same will be mapped onto by the hidden layer and due to parallel-distributed processing of the system, maximum fault tolerance capability exhibited by the hidden layer and hence overall probability of tolerating faults is very high in multilayer feed forward systems.

Random value of Wo plays a very high role in defining the sensitivity of place with respect to faults. It is shown that probability to tolerate the fault can be defined for each & every place just by using the analysis of random values of Wo. Some graphs have been used to analyze the system as well as to prove some assumptions. By analysis, it is found that if the data (initial random values) are taken from Gaussian distribution, tolerance capability is more as compared to if data taken from uniform distribution. The data corrupted can be retrieved by using this method. The result obtained is good because of fault tolerance. The recognition depends on the extent to which the data is corrupted. As a future work, more analysis needs to be carried out with different types of data representations like image.

REFERENCES

[1] *Simon Haykin, "Neural Networks: A Comprehensive Foundation", New York: Macmillan, (1998).*

[2] *Jacek M. Zurada, "Introduction to Artificial Neural Systems", Jaico publications, 3rd edition, (1999).*

[3] *Maureen Caudill and Charles Butler, "*Naturally Intelligent Systems*" MIT Press, Cambridge MA, (1990).*

[4] *Daved E. Rumelhart and James L. McClelland, "Parallel Distributed Processing" vol.: 1, MIT Press, Cambridge, MA, (1986).*

[5] *Omer Mahmoud, Farhat Anwar and Momoh Jimoh E. Salami, "Learning Algorithm Effect on Multilayer Feed Forward Artificial Neural Network Performance in Image Coding", Journal of Engineering Science and Technology Vol. 2, No. 2 (2007), 188 – 199.*

[6] *M. Stevenson, R. Winter and B. Widrow, "Sensitivity of feedforward neural networks to weight errors," in* IEEE Transactions on Neural Networks, *vol. 1, no. 1, pp. 71-80, March (1990).*

[7] *Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H., "State-of-the-art in artificial neural network applications: A survey", Heliyon, 4(11), (2018)*

[8] *Anil K Jain and R. Duin, "Introduction to Pattern Recognition". The Oxford Companion to the Mind, Second Edition, Oxford University Press, Oxford, UK, (2004).*

[9] *G. Bugmann, M. Plumbley & J.G. Taylor, "Direct Approaches to Improving the Robustness of Multilayer Neural Networks", Proc. of Int.Conf. on Artificial Neural Networks (ICANN 92), 4-7 September, Brighton, UK, (1992), pp. 1063-1066.*

[10] *M.J. Carter, "The illusion of Fault Tolerance in Neural Networks for Pattern Recognition and Signal Processing", Proc. Technical Session on Fault-Tolerant Integrated Systems, Durham NH: University of New Hampshire, (1988).*

[11] *R. A. Maxion, "Fault Tolerance in Intelligent User Interfaces", IFAC Proceedings Volumes, Vol. 19, Issue 11, Oct (1986), 117-122.*

[12] *D. S. Phatak and I. Koren, "Complete and Partial Fault Tolerance of Feed forward Neural Nets", IEEE Transactions on Neural Nets, March (1995).*

[13] *George Bolt, "Investigating Fault Tolerance in Artificial Neural Networks", Advanced Computer Architecture Group. Department of Computer Science, University of York, Heslington, U. K (1991).*

[14] *Xin Yao, "Evolving artificial neural networks," in Proceedings of the IEEE, vol. 87, no. 9, September (1999), pp. 1423-1447.*