World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:5, No:2, 2011

# A Multi-layer Artificial Neural Network Architecture Design for Load Forecasting in Power Systems

Axay J Mehta[1], Hema A Mehta[2], T.C.Manjunath[3], C.Ardil[4]



***Abstract*—**In this paper, the modelling and design of artificial neural network architecture for load forecasting purposes is investigated. The primary pre-requisite for power system planning is to arrive at realistic estimates of future demand of power, which is known as Load Forecasting. Short Term Load Forecasting (STLF) helps in determining the economic, reliable and secure operating strategies for power system. The dependence of load on several factors makes the load forecasting a very challenging job. An over estimation of the load may cause premature investment and unnecessary blocking of the capital where as under estimation of load may result in shortage of equipment and circuits. It is always better to plan the system for the load slightly higher than expected one so that no exigency may arise. In this paper, a load-forecasting model is proposed using a multilayer neural network with an appropriately modified back propagation learning algorithm. Once the neural network model is designed and trained, it can forecast the load of the power system 24 hours ahead on daily basis and can also forecast the cumulative load on daily basis. The real load data that is used for the Artificial Neural Network training was taken from LDC, Gujarat Electricity Board, Jambuva, Gujarat, India. The results show that the load forecasting of the ANN model follows the actual load pattern more accurately throughout the forecasted period.

***Keywords*—**Power system, Load forecasting, Neural Network, Neuron, Stabilization, Network structure, Load.

## I. Introduction

THE basic building block of all biological brains is a nerve cell, or a neuron. Each neuron acts as a simplified numerical processing unit. In essence, the brain is a bundle of many billions of these biological processing units, all heavily interconnected and operating in parallel. In the brain, each neuron takes input values from other neurons, applies a transfer function and sends its output on to the next layer of neurons. These neurons, in turns, send their output to the other layers of neurons in a cascading fashion. In a similar manner, Artificial Neural Networks (ANN) are usually formed from many hundreds or thousands of simple processing units, connected in parallel and feeding forward in several layers. Because of the fast and inexpensive personal computers availability, the interest in ANN's has blossomed in the current digital world. The basic motive of the development of the ANN was to make the computers to do the things, what a human being cannot do. A development of an ANN is an attempt to simulate a human brain [1].

The primary pre-requisite for system planning is to arrive at realistic estimates for future demand of power, which is known as Load Forecast. Power system planning is done to ensure adequate and reliable power supply to meet the estimated load demand both nearer and in the distant future. This must be done at minimum possible cost, keeping the quality of supply satisfactory. Basically, the load forecasting is not more than an intelligent projection of past and present patterns to determine future load with sufficient reliability.

A detailed study of past system performance with regards to its load pattern, constitution and region wise distribution is necessary to establish a general behavior pattern. Future assessment has to be drawn off after careful screening of the reference data pattern. An over estimation of the load may cause premature investment and unnecessary blocking of the capital, whereas under estimation of load may result in shortage of equipment and circuits. It is better to plan the system for the load slightly higher than expected so that no exigency may arise. From the past load growth, one can analyze the factors which contribute the growth. Unfortunately an accurate forecast depends on the judgment of the forecaster, and it is impossible to rely strictly on the analytical methods to obtain an accurate load forecast. In the literature, many techniques have been proposed, used for load forecasting [2].

Time series approach assumes that the load of a time depends mainly on previous load patterns, such as the auto regressive moving average models & the spectral expansion

[1] Axay J. Mehta is currently Professor & Head, Department of Electrical Engineering, G.H. Patel College of Engineering & Technology, Vallabh Vidyanagar, Gujarat, India and also a Research Scholar doing his Ph.D. in the Department of Systems & Control Engineering, Indian Institute of Technology Bombay, India, E-mail: axay@sc.iitb.ac.in, mehta_a_j@rediffmail.com
[2] Hema A. Mehta is currently Assistant Professor in the Department of Electrical Engineering, A.D. Patel Institute of Technology, Vallabh Vidyanagar, Gujarat, India
E-mail: engineer_resi@yahoo.com
[3] T.C. Manjunath is currently, Professor & Head in Electronics and Communications Engineering Dept. of East West Institute of Technology, Bangalore, Karnataka, India.
E-mail: tcmanjunath@gmail.com.
[4] C. Ardil is with the National Academy of Aviation, AZ 1056 Baku, Azerbaijan.

technique. The regression method utilizes the tendency that the load pattern has a strong correlation to the whether pattern. The sensitive portion of the load is arbitrarily extracted & modeled by a predetermined functional relationship with weather variable. In this paper, a load-forecasting model is developed using a multilayer neural network with an appropriately modified back propagation-learning algorithm.

The model is created to produce a short term load forecasting of the load in the 24 hours of the forecast day concerned or average daily load according to requirement. The real load data that used for the ANN training was taken from Load Dispatch Centre, Gujarat Electricity Board (G.E.B.), Jambuva, Gujarat, India. Data for the month of January for three calendar years (1997, 1998, 1999) was taken for training the proposed ANN model and the data for calendar year 2000 was use to compare the forecasted data.

The training is done in a supervised manner that is for every input vector, the desired output is given and weights are adapted to minimize an error function that measures the discrepancy between the desired output and the actual output computed by the network. For training the network, the back propagation-training algorithm is used where data of previous two years are used as inputs. In a series of experiments, the time series of daily load is presented to the network and the network is than made to predict the next value in the time sequence.

The comparison of the predicted value and the actual known value for the processed time slot resulted in a value of the calculated mean square error (MSE), which is then propagated back through the feed forward links. The back-propagation algorithm globally modifies the weights of all internal links to minimize the error. If the error gradient still exceeded a pre-defined-small value, a new epoch is initiated. Finally, after the training phase has processed all epochs, its performance could be judge in terms of the globally calculated MSE. After all epochs of the training phase were processed the network was ready for prediction and generalization.

The paper is organized in the following way. A brief introduction about the related literature about the neural network and load forecasting was presented in the previous section. In Section 2, the review of the Artificial Neural Network and various training algorithms are presented. In Section 3, a brief review of the network structures and the learning process is presented. The back-propagation algorithm is presented in section 4. Section 5 depicts the training process of the ANN. The load forecasting issue is dealt with in the section 6. various methods of the load forecasting are referred. Section 5 deals with the proposed algorithm for load forecasting. In the section 6, the load forecasting issues using the ANN is presented. Overall architecture is presented in section 7 with the training of the designed neural network in the section 8. The simulation results are presented in the section 9, followed by the conclusions and the references.

## II. Artificial Neural Network (ANN)

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANN's, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANN's as well [3].

The basic building block of all biological brains is a nerve cell, or a neuron as shown in the Fig. 1. Each neuron acts as a simplified numerical processing unit. In essence, the brain is a bundle of many billions of these biological processing units, all heavily interconnected and operating in parallel. In the brain, each neuron takes several input values from other neurons, applies a transfer function and sends its output on to the next layer of these neurons. These neurons in turn send their output to the other layers of neurons in a cascading fashion.

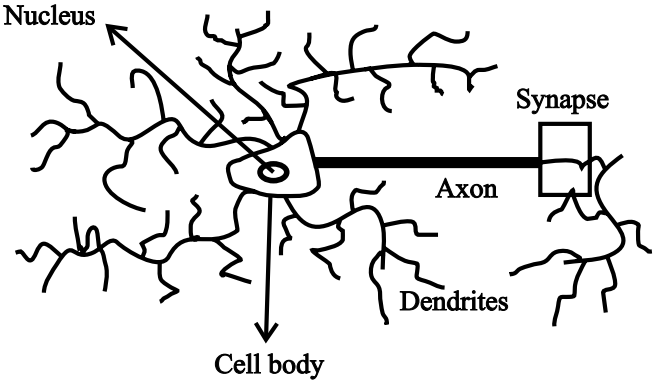


Fig. 1 Basic components of a neuron

In a similar manner, ANN's are usually formed from many hundreds or thousands of simple processing units, connected in parallel and feeding forward in several layers. In a biological neural network, the memory is believed to be stored in the strength of interconnections between the layers of neurons. Using neural network terminology, the strength or influence of an interconnection is known as its weight. ANN borrows from this theory and utilizes variable inter-connections weights between layers of simulated neurons.

ANN's were proposed early in 1960's, but they received little attention until mid 80's. Prior to that time, it was not generally possible to train networks having more than two layers. These early two layers networks were usually limited to expressing linear relationships between binary input and output characters. Unfortunately, the real world is analog and doesn't lend itself to a simple binary model. The real break through in ANN research came with the discovery of the back propagation method.

Because of fast and inexpensive personal computers availability, the interest in ANN's has blossomed. The basic motive of the development of the neural network was to make the computers to do the things, which a human being cannot do. Therefore, ANN is an attempt to simulate a human brain. Hence, the ANN architecture can be easily compared with the human brain. The following sections follows a brief review about the literature of the neural networks and its structures.

A. *Perceptron*

The perceptron, a basic neuron is invented by Rosenblatt (1962). It is a single layer neuron that contains the adjustable weight & some threshold values. The inputs are more than one & their weight should be more. The inputs are $(x_1, x_2, ..., x_n)$ corresponding weights are $(w_0, w_1, ..., w_n)$ as shown in the Fig. 2. The processor itself consists of weights, the summession processor & the adjustable threshold. This weight can be thought of propensity of the perceptron to fire irrespective of its inputs. This weight can be known as the bias terms [4].

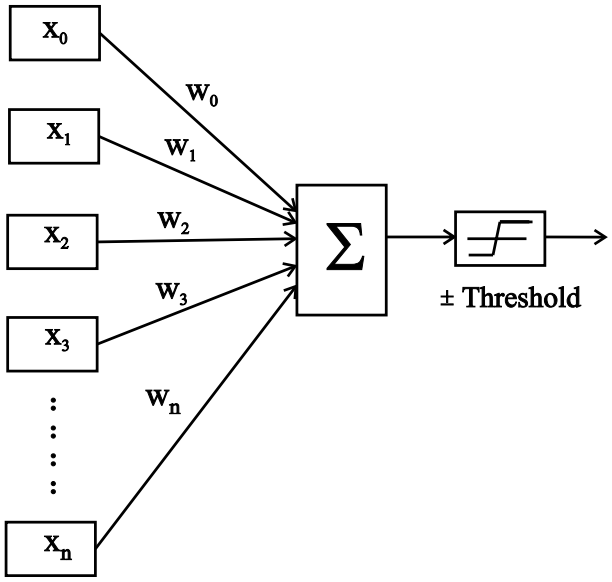


Fig. 2 Perceptron

B. *Basic Model of ANN*

Serious efforts to create a model of a neuron have been underway during the last 100 years, and remarkable progress has been made. Neural connections are significantly fewer and similar than the connections in the brains. The basic model of ANN is as shown in Fig. 3. This is the combination of perceptron as discussed earlier which forms the Artificial Neural Network which is actually practiced.

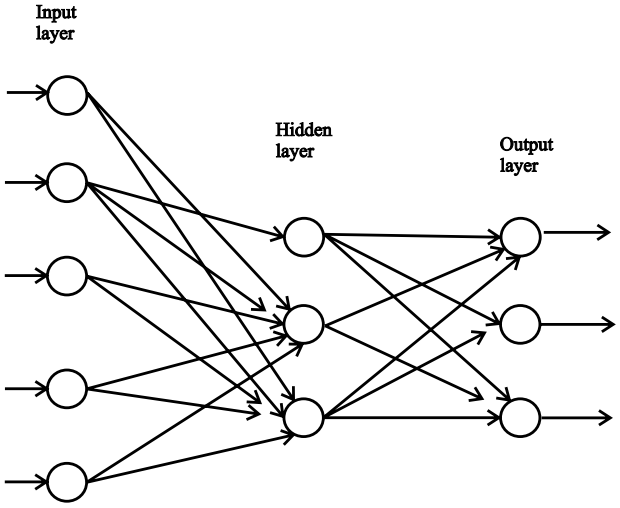


Fig. 3 Basic Model of ANN

C. *Neuron*

A neuron (or cell or a unit) is an autonomous processing element. Neuron can be thought of a very simple computer. The purpose of each neuron is to receive information from other neurons, perform relatively simple processing of the combined information, and send the results to one or more other neurons. In many of the illustrations shown in various literatures, the neurons are generally indicated by circles or squares. Sometime these neurons are denoted by $n_1, n_2, ..., n_N$, where $N$ is the total number of neurons in the networks.

D. *Layers*

A layer is a collection of neurons that can be thought of as performing some type of common functions. These neurons are usually numbered by placing the numbers or letters by each neuron and are generally assumed that no neuron is connected to another neuron in the same layer. All neurons nets have an input layer and have outputs to interface with the external environment. Each input layer and each output layer has at least one neuron. Any neuron that is not in an input layer or in an output layer is said to be in a hidden layer (shown in Fig. 2) because neither its input nor its output can be observed from outside. Sometimes the neurons in a hidden layer are called feature detectors because they respond to particular feature detectors because they respond to particular features in the previous layer.

E. *Synapses (arcs / links)*

An arc or a synapse can be a one-way or a two-way communication link between two cells as shown in the Fig. 1. A feed-forward network is one in which the information flows from the input cells through hidden layers to the output neurons without any paths whereby a neuron in a lower-numbered layer receives input from the neurons in a high-numbered layer. A recurrent network, by contrast, also permits communication in the backward direction [5].

F. *Weights*

A weight $w_{ii}$ is a real number that indicates the influence that neuron $n_i$ has on neuron $n_i$. For example, positive weights indicate reinforcement, negative weights indicate inhibition and a weight zero (or the absence of weight) indicates that no direct influence or connection exists. The weights are often combined into a matrix $w$. These weights may be initialized as given and predefined values, to initialized as random numbers, but they can be altered by experience. It is in this way that the system learns. Weights may be used to modify the input from any neutron. However, the neurons in the input layer have no weights, that is the external inputs are not modified before going into the input layer.

G. *Propagation Rule*

A propagation rule is a network rule that applies to all the neurons and specifies how outputs from cells are combined into an overall net input to neuron $n$. The term 'net', indicates this combination. The most common rule is the weighed sum rule wherein, adding the products of the inputs and their corresponding weights forms the sum,

$$net\ i = b_i + \sum w_{ij}\ n_{ij} \quad (1)$$

where, $j$ takes on the appropriate indices corresponding to the numbers of the neurons that send information to the neurons $n_i$. The term in $p_j$ represents the inputs to neurons $n_i$, from neuron $n_j$. If the weights are both positive and negative, then this sum can be computed in two parts: excitory and inhibitory. The term $b_i$ represents a bias associated with neurons $n_i$. Adding one or more special neuron(s) having a constant input of unity often simulates these neuron biases.

H. *Activation*

An activation of neuron $n_j$ is denoted as $a_i$ numerical value that represents that state of the neurons. Activation values may be discrete, such as {0,1}, {–1, 0, 1} or other values. Alternatively activation values may be continuous such as values in the intervals [0, 1] or $[aL_i, aH_i]$, where these limits are defined for each appropriate neuron. The activation $a_i$ is determined by the neuron's activation rule [6].

İ. *Activation Rule*

This network activation rule is often denoted as the '*activation function'*. Most frequently, the same function is used for all the cells and specifies how the input $x$ is combined with the current activation value $a$ in order to produce the new activation value. Several different activation functions have been used in neural net simulations.

*1) Identity Function :*

$g(x) = x$ ; The activation is just the value of the combined input.

*2) Linear Function :*

$g(x) = mx + c$ ; The activation depends only linearly on the combined input.

*3) Threshold Function : Here are three common types.*

$$g(x) = 0 \quad \text{for } x < p \quad (2)$$
$$= 1 \quad \text{for } x \geq p$$
$$g(x) = 0 \quad \text{for } x < p \quad (3)$$
$$= A_i \quad \text{for } x \geq p$$
$$g(x) = 0 \quad \text{for } x < p \quad (4)$$
$$= x \quad \text{for } x \geq p$$

The output is zero until the activation reaches a threshold $p$, then it jumps by the amount shown. When a threshold function is used, it is often illustrated by placing the threshold constant $p$ inside the appropriate neuron.

## III. Network Structures

In this section, we discuss in brief about the structure of the neural networks.

A. *Feed-forward and Recurrent Networks*

There are varieties of kinds of network structures. In a feed forward network, links are unidirectional and there are no cycles. Technically speaking, a feed forward network is a directed cyclic graph. In a layered feed forward network, each unit is linked only to units in the next layer; there are no links between units in the same layer, no links backward to a previous layer, and no links that skip a layer.

The significance of the lack of cycles is that computation can proceed uniformly from input units to output units. The activation from the previous time step plays no part in the computation, because it is not fed back to earlier units. Hence, the feed forward network simply computes a function of the input values that depends on the weight setting-it has no internal state other than the weight themselves.

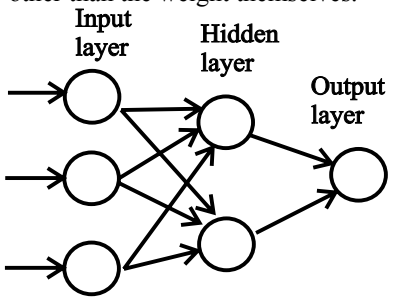


Fig. 4(a) Basic structure of the neural network-1

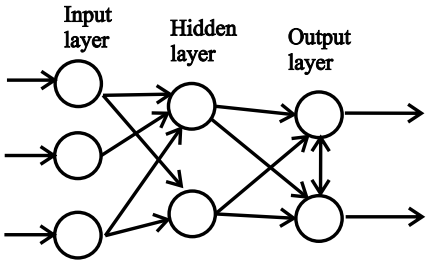


Fig. 4(b) Basic structure of the neural network-2

The brain cannot be a feed forward network, it have no short-term memory. Brain is recurrent network. Because activation is fed back to the units that cause it, recurrent have internal states stores in the activation level of the units. This

also means that the computation is much less orderly than in feed forward networks. Recurrent can becomes unstable, or oscillate, or exhibit chaotic behavior. Given some input value, it can't take long time to compute a stable output, and learning is made more difficult. Recurrent networks can implement more complex agent designs and can model system with state. The basic structure of the neural network consisting of the input, hidden and the output layers is shown in the Figs. 4(a) and 4(b) respectively [7].

### B. *Learning in ANN*

Learning is the process by which the neural network adapts itself to a stimulus, and eventually, it produces a desired response. Learning is also a continuous classification process of input stimuli when a stimulus appears at the network; the network either recognizes it or develops a new classification. In actuality, during the process of learning, the network adjusts its parameters the synaptic weights, in response to an input stimulus so that its actual output response is the same as the desired one, the network has completed the learning phase in other words, it has 'acquired knowledge'. Mathematical expressions, learning equations describe the learning process for the paradigm, which in actuality is the process for self-adjusting its synaptic weights.

### C. *Supervised Learning*

During the training session of a neural network, an input stimulus is applied that results in an output response. The response is compared with a prior desired output signal, the target response, if the actual response differs from the target response; the neural network generates an error signal, which is then used to calculate the adjustment that should be made to the network's synaptic weights so that the actual output matches the target output. In other words, the error is minimized, possibly to zero. The error minimization process requires a special circuit known as supervisor; hence, the name, 'supervised learning'.

With ANN the amount of calculation required to minimize the error depends on the algorithm used; Some parameters to watch, are the time required per iteration, the number of iterations per input pattern for the error to reach minimum during the training session, whether the neural network has reached a global minimum or a local one, and, if a local one, whether the network can escape from it or it remains trapped.

### D. *Unsupervised Learning*

Unsupervised learning does not require a teacher, that is, there is no target output. During the training session, the neural net receives at its input many different excitations, or input patterns and it arbitrarily organizes the patterns in categories. When a stimulus is later applied, the neural net provides an output response indicating the class to which the stimulus belongs. If a class cannot be found for the input stimulus, a new class is generated. Example, show a person a set of different objects. Then ask him/her to separate object out into groups or classifications such that objects in a group have one or more common features that distinguishes them form another group.

When this is done, show the same person another object and ask him/her to place the object is one of the groups. Grouping may be based on shape, color, or material consistency or on some other property of the object. If no guidelines have been given as to what type of features should be used for grouping the objects, the grouping may or may not be successful.

### E. *Reinforced Learning*

Here, we requires one or more neurons at the output layer and a teacher that, unlike supervised learning, does not indicate how close the actual output is to the desired output but whether the actual output is the same with the target output response is obtained.

The teacher does not present the target output to the network, but presents only a pass/fail indication. Thus, the error signal generated during the training session is binary: pass or fail.

If the teacher's indication is 'bad' the network readjusts its parameters and tries again and again until it get output response right. Process of correcting synaptic weight follows a different strategy than the supervised learning process [8].

Some parameters to watch are the following: the time per iteration and the number of iteration per pattern to reach the desired output during the training session, whether the neural network reaches a global minimum or a local minimum. Certain boundaries should be established so that the trainee should not keep trying to get the correct response at infinitum.

### F. *Competitive Learning*

Competitive learning is another form of supervised learning that is distinctive because of its characteristic operation and architecture and several neurons are at the output layer.

When an input stimulus is applied, each output neuron competes with the others to produce the closest output signal to the target.

This output then becomes the dominant one, and the other outputs cease producing an output signal for that stimulus. For another stimulus, another output neuron becomes the dominant one, and so on.

When an ANN with competitive learning is part of greater ANN system then, because of connectivity issues, these random specializations may not always be desirable.

Competitive learning is frequently encountered in groups of people where each member of the group was selected and trained to perform specific tasks based on the principle of the right person at the right time at the right place.

## IV. Back-propagation Learning Algorithm

Back-propagation algorithm is a supervised learning algorithm, which is dealt with in the following paragraphs.

### A. *Learning Factors*

#### *1) LEARNING RATE :*

The learning rate governs the rate at which the weights are allowed to change at any given presentation. Higher learning

rates speed the convergence process, but can result in overshooting or non-convergence. Slower learning rates produce more reliable results at the expense of increased training time.

*2) MOMETUM FACTOR :*

During the learning process of neural network, the speed of convergence can be increased by modifying the weight modification steps by including the momentum term, 'a'. The momentum term technique can be recommended for a problem with convergence that occurs too slowly or for cases when learning is difficult to achieve. By varying 'a', it is possible to shape the error surface. Values of 'a' that are close to unity ensuring circular error contours in contrast to small values, which produce a narrow valley [9].

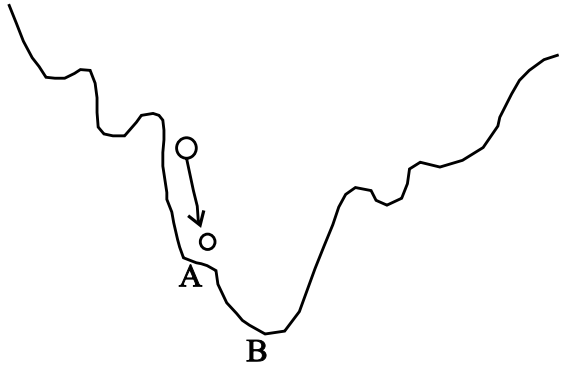


Fig. 5 Position of local minima (A) and global minima (B)

*3) LOCAL MINIMA:*

This can be well understood from Fig. 5 shown. Assume that the stable state for a given neural network is "B" but due to certain reasons; may be higher learning rate or higher momentum factor or due to insufficient available historical data, the network will settle itself ultimately into one of the nearest stable state; which is labeled as "A" here instead of setting in "B". This stable state "A" in this case is termed as "local minima".

*4) GENERALIZATION AND MEMORIZATION:*

If all possible inputs and outputs are shown to a back-propagation network, the network will probably find a set of weights that maps the inputs onto the outputs. For many Al problems, however it is impossible to give all possible inputs. e.g., Face recognition problem. ANN should promise a generalization mechanism. If it works in a domain where similar inputs get mapped onto similar outputs, network will interpolate when given inputs it has never seen before that means that network's input-output pattern is no more generalized, but it has memorized it. Memorization can adversely affect the testing of the ANN. This can be very well understood from the Fig. 6 shown.

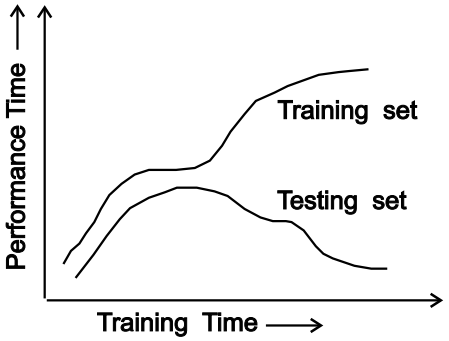


Fig. 6 A common generalization and memorization effect in neural network learning

The Fig. 6 shows the common generalization effect during a long training period. During the first part of the training, performance on the training set improves as the network adjusts its weights through the back propagation. Performance on the test set also improves, although it is never quite as good as the training set. After a while, network performance reaches a plateau as the weights shift around, looking for a path to further improvement. Ultimately, it is found and the performance on the training set improves again. But the performance on the test set gets worse. Because the network has begun to memorize the individual input-output pairs rather than setting for weights that generally describe the mapping for all cases [10].

To avoid the memorization following steps can be taken:

- Stop training when a plateau has been reached on the assumption that any other improvement will come through "cheating".
- Add deliberately small amount of noise to the training inputs. The noise should be enough in order to prevent memorization, but it should not be so much that it confuses the classifier.
- Reduce the number of hidden units in the network, creating a bottleneck between the input and output layers. Confronted with a bottleneck, the network will be forced to come up with compact internal representations of its inputs.
- Initial weights: The weights of the network to be trained are typically initialized at small random values. The initialization strongly affects the ultimate solution. If all weights start out with equal weights values, and if the solution requires that unequal weights be developed, the network may not train properly. Unless the network is disturbed by random factors or the random character of input patterns during training, the internal representation may continuously result in symmetric weights.
- Also, if the weights are initialized to high values, may lead to situation of saturation and the error, will not decrease after a certain training period. Because of this, the results obtained will be erroneous because of insufficient training of the network.
- Network failure: Also, the network may fail to learn the set of training examples with the stabilizing or even increasing as the learning continues. In fact, many

empirical studies of the algorithm point out that continuing training beyond a certain low-error plateau results in the undesirable drift of weights, causes error to increase and the quality of mapping implemented by the network decreases.

- Steepness of the activation function: As discussed in back propagation algorithm, the neuron's activation function is characterized by its steepness '$\lambda$'. Both the choice and shape of the activation function will strongly affect the speed of the network learning.
- The Fig. 7 shows the slope function of the activation function and it illustrates how the steepness '$\lambda$' affects the learning process. Since weights are adjusted in proportion to the value $f'(\lambda)$, the weights that are connected to units responding in their midrange are changed the most.
- The weights of uncommitted neurons with uncertain responses are thus affected more strongly than of those neurons that are already heavily turned on or turned off.

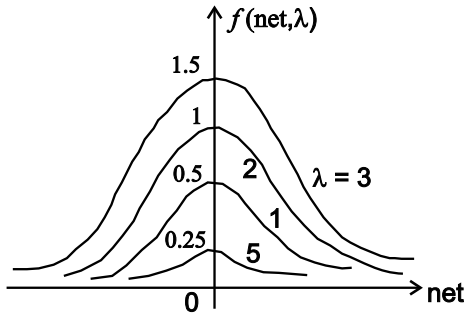


Fig. 7 The slope of activation function for various '$\lambda$' values

## V. Training of ANN

The training of the artificial neural network is done as follows.

### GATHER DATA

The first step in training an ANN is obtaining accurate historical data on the phenomena of interest. Noisy, production grade data is acceptable. Because of their highly parallel nature, ANN's are very fault-tolerant and can learn to ignore noise with little impact on the network output in fact, some times noise is purposely added to the historical data during the network's training process. This is done to prevent memorization and improve network ability to generalize.

### NORMALIZE DATA

Once the historical data is gathered, the next step in training is to normalize all the data so that each value falls within the range from –1 to +1. This is done to prevent the simulated neurons from being driven too far into saturation. Once saturation is reached, change in the input value results in little or no change in the output. This means that it's the intelligent that can't be captured. Fortunately, most commercial ANN software will perform the normalization automatically.

### SELECT NEURAL NETWORK ARCHITECTURE.

Neural networks are multiplayer devices. Because, two-layer networks are very limited in their abilities, most neural networks are made up three or more layers. The first layer is always the input layer. It acts as simple buffer feeding information from the input vector through the interconnection weights to the second layer, which is known as the hidden layer.

The final output layer is where the output data is recovered and renormalized. By verifying the number of hidden layers and the number of simulated neurons within each layer, the performance of a neural network can be improved or degraded. There are currently no general guidelines for determining a performance with combinations of neurons and hidden layers. The researcher must experiment with different architectures to find a suitable and a amicable solution.

### TRAIN NETWORK.

To train a network, the historical input and output training patterns are shown to the network repeatedly. With each presentation, an input pattern is passed forward through each layer of the network. As shown in Fig. 8, the output value from each neuron is multiplied by its respective interconnection weight to arrive at the input value for the next neuron.

At the input layer, each neuron's output is simply equal to its input. In the hidden and output layers, each neuron's output is determined by the sum of its inputs and its sigmoid transfer function.

Each input vector is passed forward through the network and an output vector is calculated. During training, the network's output is compared to the actual observations, and an error term is created.

This error is fed backwards through the network, from the output layer, through the hidden layer and back to the input layer, the interconnection weights between each layer are adjusted based on the computed error and a learning rate parameter [11].

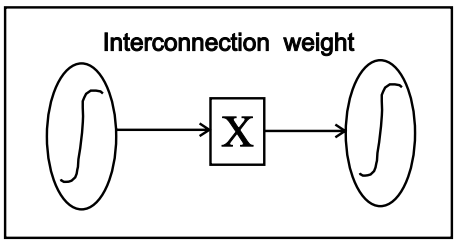


Fig. 8 Output of each neuron is multiplied by an interconnection weight to calculate the input for the following neuron

This process of presenting input data, passing it forward through the network and back propagating the error is repeated for each observation in the historical training set. The training set is shown to the network many times, until the output values converge to a solution.

## VI. Load Forecasting

Power system planning is done to ensure adequate and reliable power supply to meet the estimated load demand both in the nearer and distant future. This must be done at

minimum possible cost keeping the quality of supply satisfactory. The primary pre-requisite for system planning is to arrive at realistic estimates for future demand of power, which is known as "LOAD FORECAST". Basically load forecast is no more than an intelligent projection of past and present patterns to determine future once with sufficient reliability. There is no easy and short cut method for this exercise.

A detailed study of the past system performance with regards to its load pattern and its constitution and region wise distribution is necessary to establish a general behavior pattern. Future assessment has to be drawn off after careful screening of the reference data. An over estimation of the load will cause premature investment and unnecessary blocking of the capital; where as under estimation of the load will result in a shortage of the equipment and circuits. It is always better to plan the system for the load slightly higher than expected one, so that no exigency may arise.

From the past load growth, one can analyze the factors which contribute that growth. For future, the contribution of these factors, if any, should be carefully analyzed. Unfortunately, an accurate forecast depends on the judgment of the forecaster, and it is impossible to rely strictly on analytical methods to obtain an accurate forecast. Power system expansion and planning starts with a forecast of anticipated future load requirements. Classification and characteristics of various types of loads are as described below.

### A. *Classification of different types of loads*

Load may be classified broadly as:

- Residential Load
- Commercial Load
- Industrial Load
- Other Loads

Residential customer uses load for domestic purposes, where as commercial and industrial customers use energy for commercial and industrial purpose. Other customers include municipalities or division of state and federal government using energy for street and highway lighting. In addition sales to public authority and to rail road and railways come under the other classification. Residential customer is subdivided into rural, urban and other subdivisions. Classification of customer for forecasting purpose would be made by types of use, level of use, rate schedule, or as per the geographic area.

### B. *Characteristics of loads*

Of the three broad classes of loads, residential loads have the most constant annual growth rate and the most seasonal fluctuations. The seasonal variations of the residential component in many cases are responsible for the seasonal variations in system peak, the extent of the residential influence depending on the percentage of total system load that is residential. This characteristic is due to the use of weather-sensitive devices such as space heaters and air conditioners, water heaters, refrigerators and dryers. Residential loads will play an even greater role in determining system load patterns. Industrial loads are considered base loads that contain little weather dependent variations.

However, these loads may have unique characteristics because of shift options. Most of the system peak demands occur as direct results of seasonal weather extremes. "Heat storms" and "Cold wave" are terms describing inclement weather and that adversely affects the ability of a utility to meet load requirements. In addition to the tremendous impact of weather on the residential and commercial loads, economic and demographic effects cause other seasonal variations. Industrial customers are characterized by cyclical variations in load requirement due to normal variation in production level.

### C. *Approaches to load forecasting*

In preparing a forecast, the system planner is immediately confronted with the following basic questions:

- Should peak demand be forecast using forecasted energy and load factors, or should it be forecast separately ?
- Should the total forecast be determined by combining forecasts of appropriate load components, or should the total forecast be directly obtained from historical total load data ?
- Should average or extreme whether conditions be used ?
- Should simple forecasting methods be used, or should more formal mathematical be investigated ?

The answers to these fundamental questions set the direction to be taken by the forecaster to determine future requirements. Probably, every utility answers these questions differently indicating that no one approaches satisfactorily in all these cases. It is now clear that both energy & demand forecasts are usually needed. Thus, two options are open to the forecaster. He can forecast energy-considered to be less difficult and obtain the demand forecast from it by using forecasted load factors [12].

But, forecasting load factors can itself be difficult for the systems having load factors that are varying erratically. The second option is to forecast pick demand directly, although peak demand can vary erratically. The advantages of using energy as a primary forecast and obtaining pick demand forecast from it, is that energy tends to be much less erratic and is considered a better train growth indicator and it is readily related to demographic and economics factors, using energy data broken down into a classes of services, areas, etc.

To assemble the total load forecast is by appropriately combining the forecast of certain components. An advantage of the component is that abnormal condition in growth trends a certain component can be detected, thus preventing misleading forecast conclusions. Again no one approaches consistently used in the industry, and both are used successfully by a different utility. Future weather sensitive load components can be obtained by extrapolation. Other utilities used historical weather data to develop normal probability distribution for weather variable, which are than

used in conjunction with a weather load model to determine the probability distribution of the weather sensitive components of future loads [13].

D. *Load forecasting methodology*

Forecasting, as stated in the introduction to this part, is simply a systematic procedure for quantitatively defining future loads; Depending on the time period of interest, a specific forecasting procedure may be classified as a short-term intermediate, or a long-term technique. Because, system planning is our basic concern and because planning for the addition of new generation, transmission, and distribution facilities must begin 4 - 10 years in advance of the actual in-service date, the methodology of intermediate-range forecasts is discussed here. Forecasting techniques may be divided into three broad classes. Techniques may be based on extrapolation or on correlation or on a combination of both. Techniques may be further classified as deterministic, probabilistic, or stochastic. Some of the methods are listed below.

- METHOD OF LEAST SQUARE
- EXTRAPOLATION
- SCHEER'S METHOD.
- END-USE METHOD.
- MATHEMATICAL EQUATION.
- LOAD FACTOR METHOD.
- CO-RELATION METHOD.

## VII. LOAD FORECASTING USING ANN

A load-forecasting model is developed using a multilayer neural network with an appropriately modified back propagation learning algorithm. The model can be created to produce a short term forecast of the load in the 24 hours of the forecast day concerned or average daily load, i.e. according to the requirement. However, for some poor initial starting points, because the problem is very complex, it takes a much longer time to obtain a much comparable solution. The points in the direction of evolutionary computing being integrated with parallel processing techniques to solve practical problems.

A. *Development of load forecasting model using ANN*

Power load demand is sensitive to whether variable such as temperature, wind speed, relative humidity & cloud cover (sky cover/light intensity). Here, the example of hourly forecast for a day is considered. There are 58 inputs to the multi-layered ANN. The required output of the next day is denoted as day (i). There will be 24 outputs since the hourly load of all the 24 hours of the day is required all together. The features that are taken into account as input factors in the load forecast system are as follows:

Two 24 hr load records of day (i-1) & day (i-2). Six more inputs are the max. & min. temperatures of day (i), day (i-1) & day (i-2). One input is also reserved to indicate where day (i) is a holiday or not. These inputs are provided to the network in the form of binary coding. Three inputs as a binary code to show 7 days of a week. On binary code is dedicated to the holidays or any yearly special occasions that may affect the forecast. In short the designed ANN is of the multi-layered type & is used to learn about the relationship the 58 inputs & 24 outputs.

B. *THE INPUTS ARE:*

- 24 Hourly loads for two days prior to the forecast day.
- 24 Hourly loads for the day prior to the forecast day.
- Max. & min. temp. for 2 days prior to the forecast day. - Max &min temp. for the forecast day.
- Day of the week 3 bits
- Holiday 1 bit

C. *THE OUTPUTS ARE:*

- Load forecast for all 24 hours of the day
- But one main thing to be considered while feeding the inputs to the network is that, the values of the inputs have to be normalized. They are normalized as indicated by the below given equation;

$$\text{Normalized value} = \frac{\text{Actual Values-Min. Values}}{\text{Max. Values-Min. Values}} \quad (5)$$

where, min. & max. are the minimum & maximum values of the attribute, respectively. Generally, mean square error (MSE) is used to measure the accuracy of the network & the sigmoid activation function is adopted. Other error functions are also available which can be used according to the requirement. The number of neurons in the hidden layer is determined by the trial & error method using the validation set. This guarantees the proper selection of the number of neurons in hidden layer. Various neuron numbers are tested & the MSE error on training validation & testing is measured.

A Simple architecture feed-forward Artificial Neural Network (ANN) is proposed for the Short - Term - Load - Forecasting (STLF) in this paper. The network is trained using the back propagation-learning algorithm with a learning rate '$\eta$' and with a momentum factor '$\alpha$'. The performance of the network for one-day ahead average load forecast and one day ahead hourly forecast is compared with the historical data available. It is well established that STLF is pattern recognition problem; hence, ANN is well suited for the solution as ANN mainly deals with pattern recognition problems [14].

There are 2 different neural network architectures introduced, one of which is for average daily load forecast and the other is for one day ahead forecast for hourly basis which is taken solely for verification of the correctness of the methods employed for designing the network. It is expected that the architectures are utility dependent, i.e. a single architecture may not suit to other utilities expect one for which it is designed. Hence, different architectures are required for forecasting different utility loads [18].

Feed-forward Neural Network is used here because of its easier implementation. The power-starved nature of the Indian utilities makes a major difference from its counterparts in the

developed countries as far as load behavior is concerned. Hence, the load data generated may not be able to capture the human behavior, social and environmental factors appropriately due to unbalance in supply and demand. This makes the accurate forecast of these utilities more difficult.

D. *INPUTS*

The network is required to be supplied with proper inputs to recognize the day type, hour of the day and other factors such as weather variable. The following inputs were arrived at after exhaustive experimentation and analysis [13].

E. *DAY TYPE*

TABLE I : SEVEN INPUTS IN BINARY FORM TO REPRESENT THE DAYS

| DAY of the week | BINARY CODE |
|---|---|
| MONDAY | 1000000 |
| TUESDAY | 0100000 |
| WEDNESDAY | 0010000 |
| THURSDAY | 0001000 |
| FRIDAY | 0000100 |
| SATURDAY | 0000010 |
| SUNDAY | 0000001 |

F. *SEASONAL INPUT VARIABLE*

TABLE II : 4 INPUTS IN BINARY FORM TO REPRESENT 12 MONTHS OF A YEAR

| MONTH of the year | BINARY CODE |
|---|---|
| JANUARY | 0001 |
| FEBRUARY | 0010 |
| MARCH | 0011 |
| APRIL | 0100 |
| MAY | 0101 |
| JUNE | 0110 |
| JULY | 0111 |
| AUGUST | 1000 |
| SEPTEMBER | 1001 |
| OCTOBER | 1010 |
| NOVEMBER | 1011 |
| DECEMBER | 1100 |

Seven inputs as shown in the Table I are used in binary form to represent these days. Inputs corresponding to the day of interest are one and the rest of six are zero. Seasonal load trend is the load component that is changing from month to month. This component reflects seasonal load variation caused by heating and cooling load over one year period. Four inputs are used in binary form to represent twelve months of a year as shown in Table II.

G. *DATE OF THE RESPECTIVE MONTH*

In the absence of the actual temperature prevalent for the available load data, the weather variables cannot be accommodated in the work in the network. In order to overcome this limitation, date of the respective month was selected as an input variable. Five input neurons in binary form were used to represent thirty-one days of a month as shown in Table III.

TABLE III : 4 INPUT NEURONS IN BINARY FORM TO REPRESENT 31 DAYS OF A MONTH

| DATE of the Month | BINARY CODE |
|---|---|
| Date 1 | 00001 |
| Date 2 | 00010 |
| Date 3 | 00011 |
| Date 4 | 00100 |
| Date 5 | 00101 |
| Date 6 | 00110 |
| Date 7 | 00111 |
| Date 8 | 01000 |
| Date 9 | 01001 |
| Date 10 | 01010 |
| Date 11 | 01011 |
| Date 12 | 01100 |
| Date 13 | 01101 |
| Date 14 | 01110 |
| Date 15 | 01111 |
| Date 16 | 10000 |
| Date 17 | 10001 |
| Date 18 | 10010 |
| Date 19 | 10011 |
| Date 20 | 10100 |
| Date 21 | 10101 |
| Date 22 | 10110 |
| Date 23 | 10111 |
| Date 24 | 11000 |
| Date 25 | 11001 |
| Date 26 | 11010 |
| Date 27 | 11011 |
| Date 28 | 11100 |
| Date 29 | 11101 |
| Date 30 | 11110 |
| Date 31 | 11111 |

H. *LOAD*

It is necessary to have the information of the loads of the previous days because the future load depends on these loads. These loads also discriminate the load set of the particular day to the other. The catered load of the previous day [Day (i-1)], load of the same day of the previous week, Day (i-7) and the load of the same day but of previous year, Day ($i_1$) and the inputs in this category. This forms a set of three inputs for daily loads prediction. The daily loads that are in megawatt-hours have to be scaled down to 1000 : 1 units ratio.

İ. *HIDDEN LAYER AND ITS NEURONS.*

The number of hidden layer neurons depends on the non-linearity of the problem. A single hidden layer of 10 Neurons was found to be sufficient for the present problem. This figure was arrived at by enough trials and errors. Initially, a hidden layer consisting of 19 neurons was selected for the design. The results obtained with such a network configuration was having a larger deviation from the actual load curve, hence, the hidden layer neurons was reduced to ten neurons to create

a bottle-neck. This was found to reduce the error in the forecasted load [15].

J. *BIAS*

Bias term has been applied to the input layer and the hidden layer. The value of the bias introduced can range from 0 to 1. But, by trial and error method, it is preferred to keep low the values of the bias term, such as 0.1 0.05, etc [18].

K. *OUTPUT LAYERS.*

Daily load is the only output for the present problem. Thus, the output layer has one neuron for the daily load. In case of the hourly periods, hourly load forecast has to be done as discussed in the previous sections and in the previous paragraphs, then 24 neurons in the output layer are to be required [17].

L. *LEARNING RATE.*

Learning rate controls the amount of change imposed on a connection weight during training. An appropriate choice of the learning rate is desirable for proper generalization of the network in a reasonable training. Initially learning rate of $\eta$ = 0.35 was used for faster convergence. Later on, to improve the accuracy of the load forecasting by the designed ANN network, the learning factor was changed to $\eta$ = 0.1.

M. *MOMENTUM FACTOR 'α'.*

Momentum factor is introduced to facilitate faster weight adjustment and hence faster convergence. But too high momentum factor leads to problem of 'local minima'. Hence, initially in the network $\alpha$ = 0.5 was chosen, but due to erroneous results, $\alpha$ was reduced to 0.1. With $\alpha$ = 0.1, load forecast error within permissible error limits was available.

## VIII. OVERALL ARCHITECTURE

The sum of the total number of inputs to represent day type, month type, date, relevant load comes to 19. Neurons in the hidden layer were chosen by a trial and error method. Too many numbers of neurons increase the error and too less make the network inefficient to train itself to the given variable inputs. The optimum number of neurons in the hidden layer was found to be 10. Finally, the architecture of 19-10-1 feed-forward three-layered network was found suitable, i.e., 19 neurons in the input layer; 10 neurons in the hidden layer; 1 neurons in the output layer. Algorithm for network training is as shown in Fig. 9.

## IX. NETOWRK TRAINING

The training was done in a supervised manner that is for every input vector, the desired output is given and weights are adapted to minimize an error function that measures the discrepancy between the desired output and the output computed by the network. The mean squared error (MSE) is employed as the error function. Input of previous two years was used for training the network. The back propagation training algorithm using momentum factor = 0.1 and learning rate = 0.1 was used. In a series of experiments, the time series of daily load was presented to the network and the network was than made to "predict" the next value in the time sequence [16].

The comparison of the just "predicted" value and the known value for the processed time-slot resulted in a value of the calculated mean square error (MSE) which was than propagated back through the feed forward links. The back-propagation algorithm globally modified the weights of all internal links to minimize the error. If the error gradient still exceeded a pre-defined-small value, a new epoch was started. Finally, after the training phase has processed all epochs, its performance could be judge in terms of the globally calculated MSE. After all epochs of the training phase were processed the network was ready for prediction and generalization.

The back-propagation learning algorithm is characterized by two parameters, the learning rate and the momentum factor. These were used by the process of progressive refining weights based on the RMS error minimization. One of the objectives was to find the optimal number of the training cycles performing on the training data. To do this, a network was subjected to a number of training sessions with varying number of training cycles. After each experiment the network was tested for its ability to correctly classify the test data. The experiments included training sessions of 1000 to 5000 training cycles [17].

With the increased number of the applied cycles, the results gradually showed improvement in the network ability to "predict" the input parameter used in training. However, such better performance also indicated the network's symptoms of over-fitting cycles and has been elected as not to compromise the network ability and to correctly generalize on the new (previously unseen) input parameters.

The neural network was trained in the neural network toolbox available in MatlabR12. The function "*trainlm*" was used with learning rate $\eta$ = 0.1 and momentum factor $\alpha$ = 0.1 is implemented. Initially the synaptic weights were randomly allotted to train the network. The network was trained first for 5000 iterations. Once, trained the network returned the synaptic weights, which were then fed to the network. Therefore, instead of random weights, now predefined values of synaptic weights were allotted [19].

Also, the momentum factor and learning rate were too high initially, which led the network to the condition of LOCAL MINIMA and erroneous results were obtained. Later on, the learning rate and momentum factor were reduced to a value of 0.1.

TABLE IV : HISTORICAL DATA FOR YEAR 1997, 1998, 1999 AND 2000 THAT WAS USED FOR THE ANN TRAINING WAS TAKEN FROM LOAD DISPATCH CENTRE (LDC), JAMBUVA, GUJARAT ELECTRICITY BOARD (G.E.B.), GUJARAT, INDIA.

| Date | JANUARY 1997 | JANUARY 1998 | JANUARY 1999 | JANUARY 2000 |
|---|---|---|---|---|
| 1 | 111.42 | 123.354 | 133.347 | 139.679 |
| 2 | 114.589 | 126.893 | 133.518 | 139.382 |
| 3 | 115.651 | 130.846 | 131.377 | 138.938 |

| | | | | |
|---|---|---|---|---|
| 4 | 115.056 | 131.7 | 131.921 | 140.374 |
| 5 | 115.674 | 129.95 | 135.228 | 138.674 |
| 6 | 113.957 | 130.169 | 130.188 | 141.792 |
| 7 | 110.332 | 127.624 | 131.506 | 139.749 |
| 8 | 104.719 | 123.485 | 128.198 | 137.602 |
| 9 | 105.951 | 129.282 | 129.122 | 136.67 |
| 10 | 113.957 | 127.679 | 131.5 | 138.12 |
| 11 | 111.788 | 128.433 | 135.842 | 127.214 |
| 12 | 115.17 | 125.035 | 136.068 | 132.223 |
| 12 | 117.282 | 125.946 | 130.131 | 139.85 |
| 14 | 110.548 | 123.55 | 116.567 | 127.19 |
| 15 | 108.229 | 120.224 | 123.865 | 124.095 |
| 16 | 110.797 | 122.425 | 131.886 | 128.582 |
| 17 | 113.265 | 123.372 | 133.178 | 133.903 |
| 18 | 112.17 | 122.959 | 131.471 | 135.808 |
| 19 | 112.9 | 126.248 | 131.648 | 137.67 |
| 20 | 105.688 | 124.842 | 129.823 | 141.744 |
| 21 | 115.134 | 122.02 | 132.709 | 141.68 |
| 22 | 109.114 | 125.408 | 133.441 | 139.256 |
| 23 | 112.058 | 124.444 | 134.641 | 143.879 |
| 24 | 117.26 | 137.235 | 131.382 | 144.213 |
| 25 | 116.254 | 127.235 | 130.554 | 140.216 |
| 26 | 110.912 | 129.1 | 127.379 | 135.443 |
| 27 | 111.244 | 124.782 | 127.116 | 135.444 |
| 28 | 113.7 | 127.051 | 130.556 | 135.956 |
| 29 | 111.406 | 131.865 | 126.87 | 138.281 |
| 30 | 115.558 | 132.421 | 130.602 | 139.055 |

The real load data that was used for the ANN training was taken from Load Dispatch Centre (LDC), Jambuva, Gujarat Electricity Board (G.E.B.), Gujarat, India. Data for the month of January for three calendar year (1997, 1998, 1999) was taken for training the proposed ANN and the data for calendar year 2000 was use to compare the forecasted data. (Historical Data for year 1997, 1998, 1999 and 2000 is shown in Table IV A). The network took approximately 84 epochs to train and no further improvement in training error was found, hence the further training was stopped as over training may lead to simply memorizing the input-output spaces and the network loses its ability to generalize.

## X. SIMULATION RESULTS

The input data of January for calendar year 2000 was provided on daily basis and forecasted value is compared with that of the actual load. As can be seen, the ANN model's forecasts followed the actual load pattern more accurately and, on the average throughout the forecasted period (only 2 times it exceeds 5000 MWHr). This high level of performance is mainly due to the more accurate weather modeling, and the ability of the ANN model to respond rapidly to sudden changes. Rapid model response was one of the main objectives of this development from its conception and it is certainly one of the main contributions of this work [20].

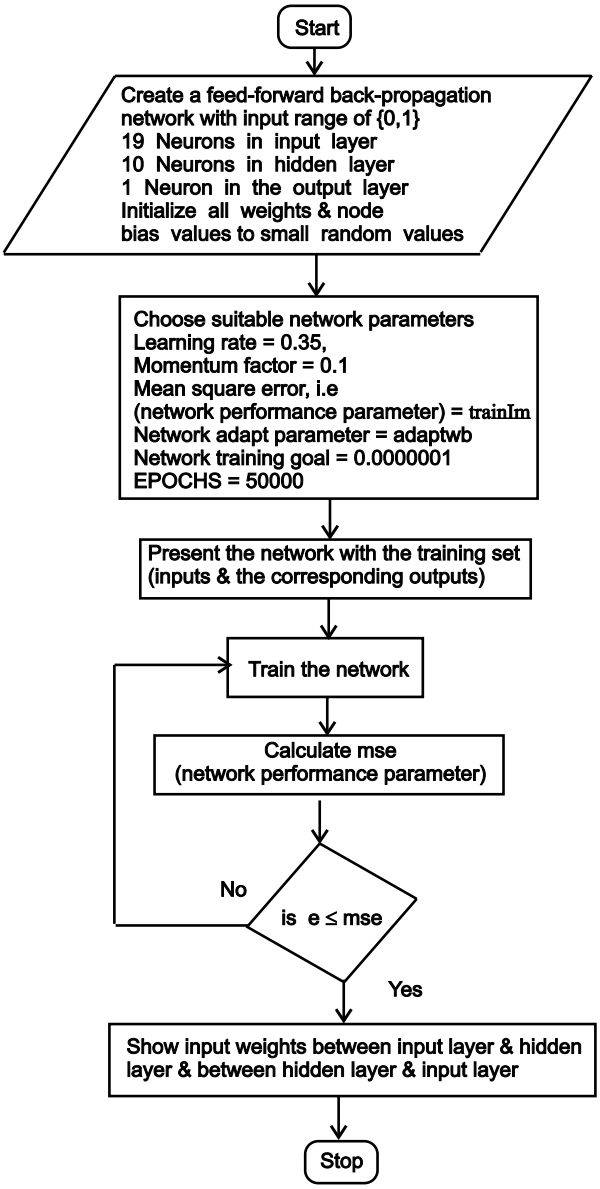


Fig. 9 Network architecture & the algorithm used

The results are presented in this section to demonstrate the performance of the developed neural network. The average error is taken as the performance measure. The forecast was based as said earlier from the historical data collected from LDC, Jambuva. The errors at certain points are more due to abnormal load behavior during those particular days. This means that the network was not trained for such a situation and the results are unexpected. It is also observed that the error is generally more on those hours of steep fall or rise in the load compared to smooth transition in load from one day to another.

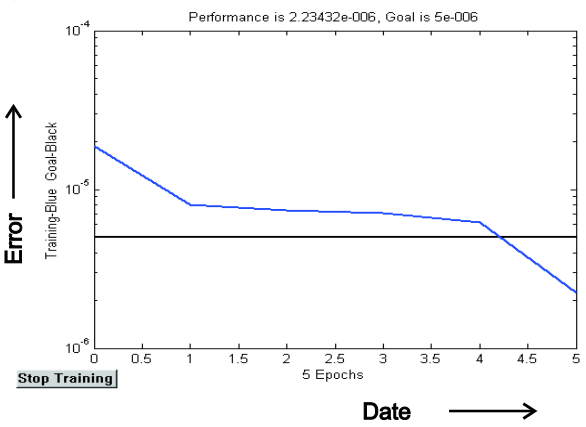


Fig. 10 Plot of error vs. date

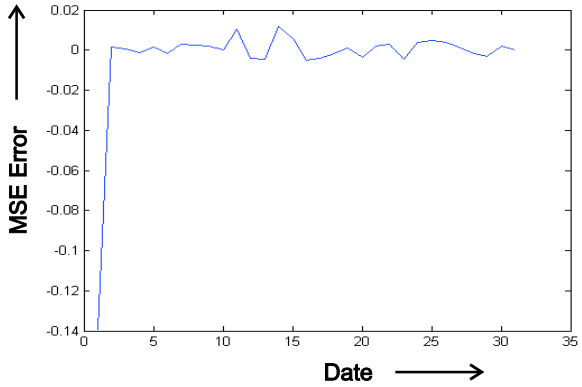


Fig. 11 Plot of MSE error vs. date

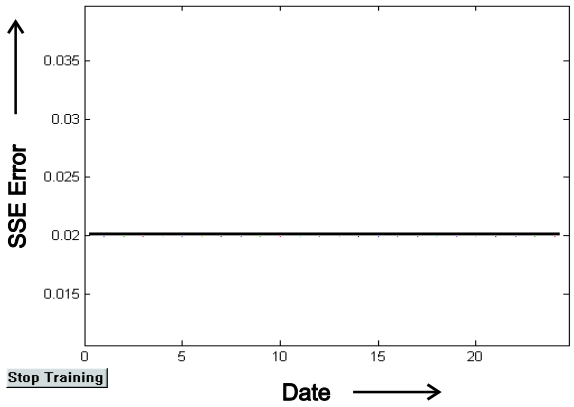


Fig. 12 Plot of SSE error vs. date

## XI. Conclusions

The modelling & design of neural network architecture for load forecasting purposes was investigated in this research paper and was successfully implemented. The results shown in the simulation section (Figs. 10-13) shows the effectiveness of the developed method. The network was subjected to a number of training sessions with varying number of training cycles. After each experiment the network was tested for its ability to correctly classify the test data. With the increased number of the applied cycles, the results gradually showed improvement in the network ability to "predict" the input parameter used in training. Rapid model response was one of the main objectives of this development from its conception and it is one of the main contributions of this work.

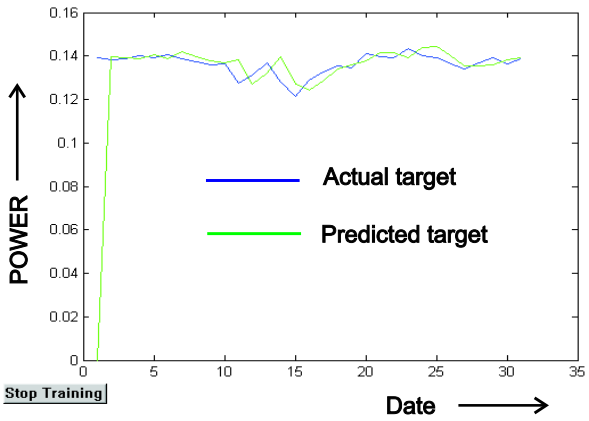


Fig. 13 Graph of Power vs Date

## References

[1]. Hadi Saadat, "Power System Analysis", *Tata McGraw-Hill Publishing Company Limited*, New Delhi, India.

[2]. Kothari D.P., and I.J. Nagrath, "Modern Power System Analysis", Tata McGraw-Hill Publishing Company Limited, Third Edition, New Delhi, India. 2003.

[3]. Robert Louis Stevenson, "Power System Analysis", *Mc. Graw Hill*, Singapore.

[4]. Simpson P.K., "Fuzzy min-max neural networks - Part I : Classification", *IEEE Trans. Neural Networks*, Vol. 3, pp. 776-786, Sep. 1992.

[5]. Guo Y, Hill D J, Wang Y, "Global transient stability and voltage regulation for power systems", *IEEE Transactions on Power Systems*, Vol.16, No. 4, pp. 678-688, Nov. 2001.

[6]. Yanjun Li, David J. Hill, Tiejun Wu, "Optimal Coordinated Voltage Control of Power Systems - An Immune Algorithm Solution", *Proc. Fifth Asian Control Conference 2004*, Paper P206, Melbourne, Australia, 21 Jul-24 Jul. 2004.

[7]. D.C. Park, M.A El-Sharkawi, R.J. Marks II, L.E. Atlas and M.J. Damborg, "Electric Load Forecasting Using An Artificial Neural Network", IEEE *Transactions on Power Systems,* Vol. 6, No. 2, May 1991, pp. 442-449.

[8]. K.Y. Lee, Y.T. Cha, and J.H. Park, "Short-term load forecasting using an artificial neural network", IEEE *Transactions on Power Systems,* Vol. 7, No. 1, February 1992, pp. 125-132.

[9]. S.-T. Chen, D.C. Yu, and AR. Moghaddamjo, "Weather sensitive short-term load forecasting using nonfully connected artificial neural network", IEEE *Transactions on Power Systems,* Vol. 7, No. 3, August 1992, pp. 1098-1105.

[10]. C.N. Lu, H.T. Wu, and S. Vemuri, "Neural Network Based Short-term Load Forecasting", IEEE *Transactions on Power Systems,* Vol. 8, No. 1, February 1993, pp. 336-342.

[11]. A.G. Bakirtzis, V. Petridis, S.J. Klartzis, M.C. Alexiadis and AH. Maissis, "A Neural Network Short Term Load Forecasting Model for the Greek Power System", IEEE *Transactions on Power Systems,* Vol. 11, No. 2, May 1996, pp. 858-863.

[12]. I. Drezga and S. Rahman, "Input Variable Selection for ANN-based Short term Load Forecasting", IEEE *Transactions* on *Power Systems,* Vol. 13, No. 4, November 1998, pp. 1238-1244.

[13]. Grene, S., I. Dobson, F.L. Alvardo, "Sensitivity of the loading margin of the voltage collapse w.r.t. arbitrary parameters," IEEE *Transactions* on *Power Systems,* Vol. 11, No. 1, Feb. 1997, pp. 262-272.

[14]. Wenxin Liu, Ganesh K., Venayagamoorthy and Donald C. Wunsch, II, "Design of an adaptive neural network based power system stabilizer", *Trans. Neural Networks*, Vol. 16, Issues 5-6, June-July 2003, Pages 891-898.

[15]. Srinivas Pillutla and Ali Keyhani , "Power system stabilization based on modular neural network architecture", *International Journal of Electrical Power & Energy Systems*, Vol. 19, Issue 6, August 1997, pp. 411-418.

[16]. José A.L. Barreiros, André M.D. Ferreira, Carlos Tavares-da-Costa, Jr., Walter Barra, Jr. and João A.P. Lopes, "A neural power system stabilizer trained using local linear controllers in a gain-scheduling scheme", *International Journal of Electrical Power & Energy Systems*, Volume 27, Issue 7, September 2005, pp. 473-479.

[17]. Srinivas Pillutla and Ali Keyhani , "Power system stabilization based on modular neural network architecture", *International Journal of Electrical Power & Energy Systems*, Vol. 19, Issue 6, August 1997, pp. 411-418.

[18]. Shu-Rong Li; Hai-Tao Shi, "Neural network based on excitation controller design of power systems via backstepping", Proc. 2003 International Conference on Machine Learning and Cybernetics, Vol. 2, Issue 6, 2-5 Nov. 2003, pp. 934 - 939.

[19]. Ruzic, S.; Vuckovic, A.; Nikolic, N, "Weather sensitive method for short term load forecasting in Electric Power Utility of Serbia", *IEEE Transactions on Power Systems*, Vol. 18, Issue: 4, pp. 1581-1586, Nov. 2003.

[20]. Loi Lei Lai, "Intelligent System Applications in Power Engineering: Evolutionary Programming and Neural Networks", *John Wiley*, New York.

**Axay J Mehta**, born in Bharuch, Gujarat, India on Oct. 20, 1975 received the B.E. Degree in Electrical Engineering from the Gujarat University in 1996 in First Class Distinction and M.Tech. in Electrical Engineering with specialization in Control Systems from the Indian Institute of Technology, Kharagpur, India in 2002 and currently a Research Scholar at the Interdisciplinary Programme in Systems and Control Engineering Department of Indian Institute of Technology Bombay, respectively. He has got a vide teaching experience in engineering colleges of Gujarat and is currently working as Professor and Head of the Department of Electrical Engineering in G. H. Patel College of Engineering and Technology, Gujarat, India. He has published a number of papers in the various National, International journals and Conferences. He is a student member of IEEE, life member of the ISTE (India), life member of SSI (India) and life member of the CSI (India). His current research interests are in the area of Variable structure control, Non-linear system dynamics and control, Power system operation and control, Artificial Intelligence.

**Hema A Mehta,** born in Godhra, Gujarat, India on Dec. 04, 1974 received the B.E. Electrical Engineering and M.E Degree in Electrical Engineering with specialization in Power System from the Gujarat University in 1996 and 1999 respectively and currently she is an Assistant Professor at A. D. Patel Institute of Technology, Gujarat, India. She has published a number of papers in the national and international journals & conferences. She is life member of ISTE and her research interest is Power system operation and control, Artificial Intelligence in power system.

**T.C.Manjunath**, born in Bangalore, Karnataka, India on Feb. 6, 1967 received the B.E. Degree in Electrical Engineering from the University of Bangalore in 1989 in First Class and M.E. in Electrical Engineering with specialization in Automation, Control and Robotics from the University of Gujarat in 1995 in First Class with Distinction and Ph.D. from the Interdisciplinary Programme in Systems and Control Engineering Department of Indian Institute of Technology Bombay in the year 2007, respectively. He has got a teaching experience of 18 long years in various engineering colleges all over the country (Karnataka, Tamil Nadu, Gujarat, Maharashtra) and is currently working as Professor and Head of the Dept. of Electronics and Communication Engg. in East West Institute of Technology in Bangalore, Karnataka, India. He also worked as a Research Engineer in the Systems and Control Engg. (IIT Bombay, India) for nearly a year and worked on control of space launch vehicles using FOS feedback technique. He has published a number of papers in the various National, International journals and Conferences and published two textbooks on Robotics, one of which has gone upto the fourth edition, titled, 'Fast Track to Robotics' and the other, which has gone upto the fifth edition, titled, 'Fundamentals of Robotics' in 2 volumes, Vol.-1 and Vol.-2 along with a CD which contains about 150 C / C++ programs for performing various simulations on robotics. He also published a research monograph in the International level from the Springer-Verlag publishers based on his Ph.D. thesis topic titled, "Modeling, Control and Implementation of Smart Structures", Vol. 350, LNCIS, costing 79.95 Euros. He was a student member of IEEE for 6 years, SPIE student member and IOP student member for 4 years, life member of ISSS (India), life member of the ISTE (India), life member of ISOI (India), life member of SSI (India) and life member of the CSI (India) and a fellow of the IETE (India). He has visited Singapore, Russia, United States of America and Australia for the presentation of his research papers in various international conferences. His biography was published in 23rd edition of Marquis's Who's Who in the World in the 2006 issue. He has also guided more than 2 dozen robotic projects. Many of his guided projects, interviews have appeared in various national newspapers and magazines. He has also presented a number of guest lectures and various seminars and participated in more than a dozen CEP / DEP courses, seminars, workshops, symposiums in the various parts of the country in different institutions and also conducted a few courses. His Ph.D. research work was based on the mathematical modeling, control and implementation of smart structures and its applications. His current research interests are in the area of Robotics, Smart Structures, and Control systems.