GENETIC OPTIMIZATION OF A SELF ORGANIZING FUZZY – NEURAL NETWORK FOR LOAD FORECASTING

P.K. Dash* S. Mishra* S. Dash+, A.C. Liew**

*Department of Electrical Engineering, Regional Engineering College, Rourkela, India.
+ Product Development Manager, Standard Charted Bank, Financial Engineering, Mumbai.
**National University of Singapore, Singapore

ABSTRACT

In this paper a self-organizing fuzzy-neural network with a new learning mechanism and rule optimization using genetic algorithm (GA) is proposed for load forecasting. The number of rules in the inferencing layer is optimized using genetic algorithm and an appropriate fitness function. We devise a learning algorithm for updating the connecting weights as well as the structure of the membership functions of the network. The proposed algorithm exploits the notion of error back propagation. The network weights are initialized with random weights instead of any preselected ones. The performance of the network is validated by extensive simulation results using practical data ranging over a period of two years. The optimized fuzzy neural network provides an accurate prediction of electrical load in a time frame varying from 24 to 168 hours ahead. The algorithm is adaptive and performs much better than the existing ANN techniques used for load forecasting.

I.INTRODUCTION

A number of algorithms and techniques has been suggested for the load prediction problems during the last two decades. The time-series and regression techniques are two major classes of conventional statistical algorithms, which are inefficient for providing acceptable accuracy limits. Artificial Neural Networks (ANN) have been proposed as a powerful tool for short-term load forecasting problems. It is known that, Artificial neural networks do not require any explicitly defined relationship between input and output variables. The corresponding mapping between the input and output is obtained using a training algorithm. The ANN modeling needs only the selection of input variables, thus avoiding the difficulties associated with conventional modeling processes.

A partially connected network [1,2] consisting of main and supporting blocks is also proposed, which makes use of model reference and functional relationships between input (e.g. past load, weather, day type and hour of the day) and output (next time step load). The various training schemes proposed include a minimum-distance based strategy [3,4] to identify the appropriate historical patterns of load and temperature. Both the above mentioned approaches use back-propagation training algorithm to train and update the model parameters.

Expert Systems provide a symbolic approach and exploit human expertise by capturing the knowledge of domain-experts and operators in the form of rules and symbols. The heuristic feature of an expert system provides an excellent method for load forecasting. Fuzzy logic based expert system for load forecasting requires development of a fuzzy rule base, which relies upon detailed knowledge of the parametric variation of the load pattern. Recently, Fuzzy-Neural-Nets (FNN) have been applied for load forecasting [5,6]. Usually, two approaches are adopted for implementing these FNNs. One of the approaches handles the fuzzified input data; in the second approach the weights of the network are computed based upon a fuzzy rule base without fuzzifying the input data. Neural Network based models for forecasting problems are less complex than fuzzy logic based systems. However, the simplicity is at the cost of an explicitly defined relationship between the individual inputs and overall model parameters. Fuzzy logic based systems allow some insight into the model parameters with the help of membership functions and rules.

The objective of the present approach is to study a self-organizing Fuzzy-Neural-Network (FNN) which combines self-organizing capability of neural networks and fuzzy logic reasoning attributes. The network modeling starts with random set of weights and hence an arbitrary set of fuzzy sets. The network is initialized with sufficiently large number of rule- nodes which subsequently get optimized using the genetic algorithm. We devise an adaptive mechanism for weight updation together with updation of the associated parameters of fuzzy membership function. The training algorithm exploits the notion of error back propagation.

0-7803-5935-6/00/$10.00 (c) 2000 IEEE

Further the genetic algorithm [7,8] is used to optimize the number of rule nodes in the fuzzification layer by manipulating a population of strings that represent different potential solutions, each corresponding to a sample point from the search space.

The following section (Section II) describes the structure of the Fuzzy Inference System (FIS) and the choice of input variables. Section III describes the training and structure optimization procedures using a genetic algorithm and the simulation results are given in section IV. The network is trained and tested using a load-data of Virginia Utility, U.S.A. over a period of 2 years.

II. OVERVIEW OF THE PROPOSED APPROACH

One of the salient aspects of Fuzzy Inference System (FIS) is the determination of the knowledge base (KB) which consists of the following subsystems:

- Mechanism for developing membership functions
- Fuzzy reasoning mechanism
- Number of rules and the rule base.

We present in this section the FIS alongwith the governing equations.

Fig.1 shows the architecture of the fuzzy neural network, comprising input, fuzzification, inference and defuzzification layers. Further the network can be visualized as consisting of N inputs, with N neurons in the input layer and R rules, with R neurons in the inference layer. There are NxR neurons in the fuzzification layer and K neurons for output layer. The signal propagation and basic function in each layer of the FNN is introduced in the following.

The input layer consists of x_i , i = 1,2, ...N, along with unity. Each neuron in the fuzzification layer represents a fuzzy membership function for one of the input variables. The activation function used in this layer is $f(\text{net}_{ij}) = \exp(-|net_{ij}|^{l_u})$ and the input to these neurons $\text{net}_{ij} = w_{ij1} x_i + w_{ij0}$, with w_{ij1} and w_{ij0} being the connecting weights between input layer and fuzzification layer.

Thus, the output of the fuzzification layer becomes

$$\mu_{ij}(x_i) = \exp(-|w_{ij1}x_i + w_{ij0}|^{l_u}) \qquad (1)$$

Where μ_{ij} is the value of fuzzy membership function of the i^{th} input variable corresponding to the j^{th} rule.

Each node j in the inference layer is denoted by Π, which multiplies the input signals and the output of the node becomes the result of product. Therefore, the output of the layer becomes

$$\rho_j(x_1, x_2, ...x_N) = \prod_i^N \mu_{ij}(x_i) \quad (2)$$

With v_{jk} being the output action strength of the kth output associated with the jth rule and utilizing weighted sum defuzzification, the network output becomes

$$o_k(x_1, x_2, ...x_N) = \sum_j^R v_{jk}\rho_j(x_1, x_2, ...x_N)$$
$$= \sum_j^R v_{jk} \prod_i^N \exp(-|w_{ij1}x_i + w_{ij0}|^{l_u}) \qquad (3)$$

It is found that the fuzzy neural network is easier to train with $y=\Sigma\rho_j\ vj$ instead of $y = \frac{\Sigma\rho_j v_j}{\Sigma\rho_j}$ in the defuzzification layer. This has been clearly mentioned in reference (9)of this paper. Equation(3) is similar to the output obtained in a radial basis functions neural network. The initial weights in V are the centers of output variable fuzzy membership functions. The range of the desired load output data is divided into R intervals as there are R number of rules. The initial v_j (j=1,2,,...,R) values are set to be the central value of these R intervals. The central values will, however, depend on the number of load and weather patterns used for training the network. Once the initial weights v_j are chosen as mentioned above, the weights are updated at every iteration during the training as shown in equation (9) using an LMS algorithm. For application of GAs to optimize the number of rules in the inferencing layer, this number is chosen as a parameter (rule node) that needs to be coded. In this study this variable is represented as a 10-bit binary number. Thus the chromosome in this case would contain a gene consisting of 10 binary digits.

III. TRAINING

Back propagation (BP) algorithm, which is the most popular method for neural network design, is being exploited to update parameters of the Fuzzy Neural network.

The error function E of the network be

$$E = \frac{1}{2}\sum_k (t_k - o_k)^2 \qquad (4)$$

Where t_k is the desired output in the k^{th} output node.

The parameter updation equations for the weights between the inference and output layers are:

$$v_{jk}(n) = v_{jk}(n-1) + \Delta v_{jk}(n) \qquad (5)$$

Where $\Delta v_{jk}(n) = \eta\delta_k y_v + \alpha\,\Delta v_{jk}(n-1)$ (6)

and n is the iteration count , $\delta_k = t_k - o_k$, $y_v = \rho_j$, η is the learning rate and α is momentum term.

For the weights between the input and fuzzification layer

$$w_{ij1}(n) = w_{ij1}(n-1) + \Delta w_{ij1}(n) \qquad (7)$$

Where $\Delta w_{ij1}(n) = \eta\delta_{ij}y_1 + \alpha\Delta w_{ij1}(n-1)$ (8)

$$\delta_{ij} = \rho_j \sum_k \delta_k v_{jk} \Big/ \mu_{ij} \text{ and } y_1 = -l_{ij}x_i\mu_{ij}|w_{ij1}x_i + w_{ij0}|^{l_u - 1} \quad (9)$$

Similarly,

$$w_{ij0}(n) = w_{ij0}(n-1) + \Delta w_{ij0}(n)$$

where $\Delta w_{ij0}(n) = \eta \delta_{ij} y_0 + \alpha \, \Delta w_{ij0}(n-1)$ (10)

$$y_0 = -l_{ij}\mu_{ij}\left|w_{ij1}x_i + w_{ij0}\right|^{l_{ij}-1}$$

The fuzzy membership function parameter is updated as

$$l_{ij}(n) = l_{ij}(n-1) + \Delta l_{ij}(n) \quad (11)$$

Where $\Delta l_{ij}(n) = \eta \delta_{ij} y_l + \alpha \Delta l_{ij}(n-1)$

and $y_l = -\mu_{ij} \log_e(\left|w_{ij1}x_i + w_{ij0}\right|) \left|w_{ij1}x_i + w_{ij0}\right|^{l_{ij}}$ (12)

We also tune the value of the learning parameter η as

$\eta(k) = \eta(k-1) + 0.2 \in (k) + 0.1 \in (k-1)$

The learning parameters are updated till some stopping criterion is reached:

$E \le \in, or \; K \ge I_{max}$

where $\in > 0$ and I_{max} is the maximum number of iterations allowed. In this implementation $I_{max} = 2400$

During training, the number of rules is increased from 1 till a satisfactory performance of the network is found using a Genetic algorithm. The learning rate 'η' which controls the rate of convergence initially set to 0.2 and is reduced gradually to 0.01 and the momentum constant 'α', added to speed up the training and avoid local minima, is kept at 0.6 throughout. The initial weights are randomly selected in the interval [-1, +1] and l_{ij} is initialized to 2. The number of iteration is set to 5000 in all cases. The training is continued till E(n)<ε at all points for a window length of 100 or the number of iteration reaches its maximum. The value of ε is taken to be $1*e^{-14}$ during training.

A. *Model Optimization*

Genetic algorithms (GA) can be viewed as a general-purpose search method, an optimization method, or a learning mechanism, based loosely on Darwinian principles of biological evolution, reproduction and "the survival of the fittest" along with genetic recombination.

GA's maintain a set of candidate solution called a population. Candidate solutions are usually represented as strings of fixed length, called chromosomes, coded with binary character set. Given a random initial population GAs operate in cycles called generations.

The basic optimization of rule nodes is started as follows:

A population of P candidate solutions encoded in the binary strings are generated at random. The fitness $f_{fitness}(P)$ corresponding to each candidate solution is computed. In this problem fitness function is defined as

$$f_{fitness}(P) = \frac{k}{k + error}, \; k = \text{iteration count} \quad (13)$$

Where, error represents the error at the output node of the fuzzy neural net resulting from a particular candidate solution. Offsprings are generated by applying two point crossover operator and mutation operators. We used the following selection mechanism for selection of the offspring for the next generation:

If the average fitness of the offspring is more than that of the parents, then they are considered as parents for the next generation, otherwise parents are retained for the new generation. Thus out of $2P$ population P candidate solutions are considered for the next generation by using the selection mechanism. The crossover and mutation probabilities are taken as P_c =0.6 and P_m = 0,0.1, respectively.

The above process is repeated until the stopping criterion is reached. The algorithm is stopped when more than half the population has equal and high fitness. Here, each string of the entire population represents a rule. The optimization process of the best number of rules in a population over each generation is illustrated in Fig. 2(a). After 98 generations, the value of the best number of rules in each generation is found to be R = 14; where R is the number of Rule nodes and is initially chosen as 5.

In the above procedure, the rule nodes which result in the output error close to zero over the entire training set are searched and the node that does not contribute significantly to the model output are omitted keeping an eye over the model performance. In a similar way, the inputs for which the fuzzy membership grade is unity or close to unity over the entire training set are traced down and are omitted from the network. The fitness value in each generation is shown in Fig. 2(b). Here we see the fitness value settles around 0.985 after 100 generations. However, we find the variation in the fitness values are very small. While operating in real-time environment, it is imperative that the load forecasting system should be able to adapt to changing conditions. We achieve this objective by the following methodology:

Before appreciable change in the load pattern is encountered the network is operated in prediction mode. Suppose at k^{th} time-step it is observed that the error in prediction is showing a gradual-increasing trend, this can be assumed that a substantial change in the load pattern is occurring. In order to reduce the error within the desired performance level, we retain the network using available data up to $(k-1)^{th}$ time-step. After training we again use the network for forecasting.

IV. LOAD FORECASTING RESULTS

For forecasting load time series in real-time it is imperative that the forecasting system should be able to handle non-stationary data. To achieve this adaptation, the optimized weights are used from the training set to forecast the first day load. For our simulation studies 5-day input nodes(N) and 14 rule nodes (R) are ultimately retained in the final forecasting model. After the end of the day the forecast model parameters are updated till the error becomes significantly small. The number of iterations required for this purpose is very small and once the new weight vector is established, this set will be used for forecasting of load over the next 24 hours. The fuzzy-neural network based load model is chosen to be different for the weekdays (Monday through Friday) and weekends (Saturday and Sunday). Holidays and special event days will have separate load model.

As the weekends are excluded from the 1st set of database for the weekdays the weight vector obtained after the forecast of Friday load is used to predict the load on Monday. For one-week ahead forecasts, the adaptation is done once a week that is at the end of the week when the entire load profile for the whole week will be available. However, if the load pattern changes appreciably, which can be identified by observing the growing prediction error, the fuzzy-neural network model is retrained using the available data and the prediction is restarted with the new set of weight vector.

The mean absolute percentage (MAPE) is used to test the performance of the model, which is defined as follows:

$$\text{MAPE} = (1/N)\sum_{i=1}^{N}\frac{\left|\text{forecasted load} - \text{actual load}\right|}{\text{actual load}} \times 100 \quad (14)$$

Where, N is the number of patterns in the data set used to evaluate the error and the accuracy of the load forecasting model.

The abnormal data obtained during thunderstorms, and faults in transmission systems are not considered for the evaluation of the new load prediction model. Holidays are considered separately along with the weekends for building the forecasting model to predict the loads during these days. The special holiday data in the past and the weekend data are used to train the forecasting model before the forecasting of load during holidays are taken up.

The proposed scheme is validated using practical data collected from the Virginia Utility, USA. We have carried out extensive simulations to validate our new approach. To evaluate the performance of the proposed network architecture, it is used to forecast both one-day and one-week ahead peak and average load.

The 24-hours ahead peak load prediction over 4-weeks in winter is shown in Fig.3(a) and Fig. 3(c). The corresponding percentage of error is shown in Fig.3 (b) and Fig. 3(d) respectively. It is observed from these figures that the predicted output closely follows the actual load pattern and the percentage of error is mostly .within ±2.0% (except in one occasion). From Fig.3(a)-(d) it is clear that the short-term load prediction in winter proposed is better than that of the summer. The degradation in performance of summer season can be attributed to abrupt variation of weather pattern in summer season.

The 24-hours and 168-hours ahead average load forecast over 4-weeks in winter are shown in Fig.4(a) and Fig.4(b), respectively. It is observed that the average load forecast is better than that of peak load forecast. This phenomenon is due to the averaging effect of the load pattern.

The results in Fig.5(a) shows the percentage of the number of days different week-days lie within certain percentage of error (PE) over the whole year. Here, in the results for 24-hours ahead peak load forecast, we see that – out of all days of individual week-day types, 70% of the days are within a PE value of 1.0, 80% within a PE of 2.0. Similarly, the percentage of number of days each week-day having different megawatt (MW) error (difference between actual and predicted load) is shown in Fig. 6(b). The results in Fig.5(b) show that almost 50% of the days of any week-day type is within 20MW error range, 70% is within 40MW and 90% within 60 MW range for 24-hours ahead peak load forecast.

The performance of the new time series forecasting model is compared to the existing ANN and FNN (Fuzzy Neural Network) based models. To evaluate the accuracy of the load prediction models, the forecasts of both 24-hours and 168-hours ahead peak and average loads are considered. Fig.6(a) and Fig.6(b) show the percentage of errors for the ANN, FNN, and self organizing FNN with genetic rule optimization (SFNN) for the month of January over a 24-hour and 168-hour time frame, respectively. The optimized SFNN model shows significant improvement in forecasting accuracy in comparison to ANN and FNN load prediction models.

V. CONCLUSIONS

In this paper we address the problem of load forecasting using a fuzzy neural network structure. The weights in different layers of the network are optimized using a novel update algorithm. The network performs satisfactorily starting from an initial set of random weights. Thus the problem of proper choice of initial weights is avoided. Again, the number of rules is determined using Basic Genetic Algorithm (BGA). The efficiency of the network is validated by selecting a practical load pattern in summer and winter. As summer load pattern is considerably temperature sensitive; the peak load forecast demonstrates the network's efficiency. The weekly MAPE value is mostly within 0.5-1.5% for 24-hours ahead peak load forecast and within 0.05-0.15% for average load forecast.

VI. REFERENCES

1. Chen S.T., Yu D.C. and Moghaddamjo A.R., "Weather sensitive short-term load forecasting using nonfully connected artificial neural network', IEEE Trans. on Power system, Vol.7, No.3, Aug.1992, pp.1098-1105.
2. Chow T.W.S. and Leung C.T., "Neural Network based short-term load forecasting using weather compensation", IEEE Trans. on Power Systems, Vol.1, No.4, 1996, pp.1730-1736.
3. Dash P.K , Satpathy H.P., Liew A.C. and Rahman S., "A real-time short-term load forecasting system using functional link neural network", IEEE Trans. on Power Systems, Vol.12, No.2, 1997, pp.675-681
4. Dash P.K., Liew A.C., Rahman S., "A fuzzy neural network and fuzzy expert system for short-term electric load forecasting", IEE Proceedings, Gen Trans. & Dist., Vol.143, No.1, 1996, pp.371-379.
5. Mori H. and H. Kobayashi H., "Optimal fuzzy inference for short-term load forecasting', IEEE Trans. on Power Systems, Vol. 11, No.1, 1996, pp.390-396.
6. Papadakis S.E., Theocharis J.B., Kiartzis S.J. and Bakirtzis A.G , "A novel approach to short-term load forecasting using fuzzy-neural networks", IEEE Trans. on Power Systems, Vol.13, 1998, No 2, pp 480-492.
7. Huang S.J., Huang C.L., "Application of Genetic-Based neural networks to thermal unit commitment", IEEE Trans. on Power Systems, Vol 12, No.2,1997, pp.654-660.
8. Homaifar A. and Cormic E. Mc, "Simulations design of membership functions and rule sets for fuzzy controllers using enetic algorithms", IEEE Trans Fuzzy Systems, Vol 3, No 3, 1995, pp 129-139

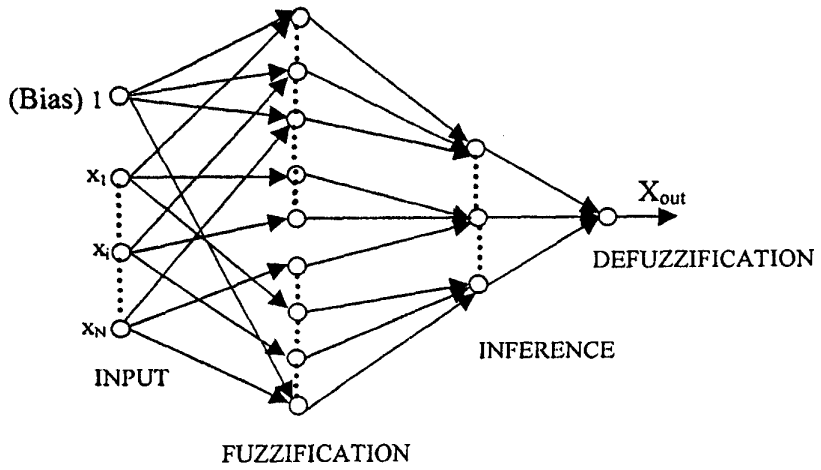


Fig.1 Load forecasting Model

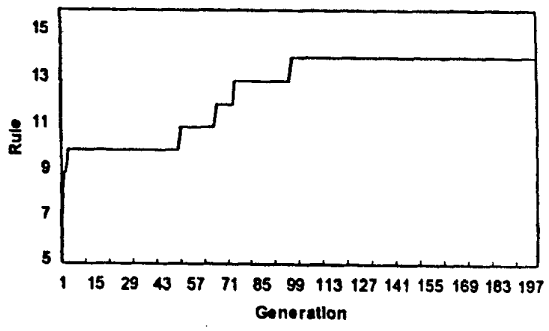


Fig. 2(a) Optimisation of the best rule value in A Population over 200 generations.

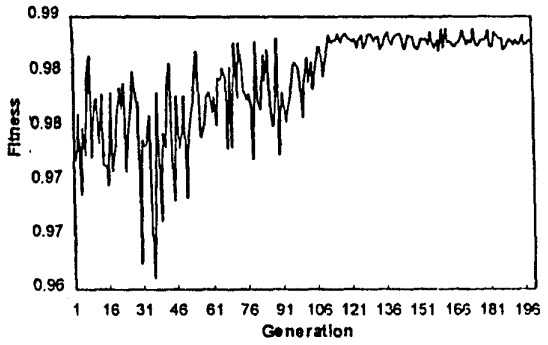
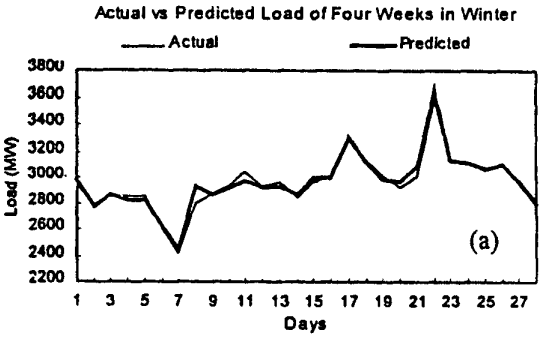


Fig. 2(b) Fitness value in a Population over 200 generations



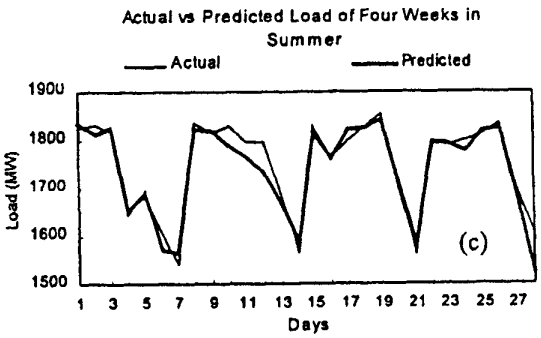


Fig. 3. 24-hours ahead actual vs Predicted Peak load , a) Winter (c) Summer

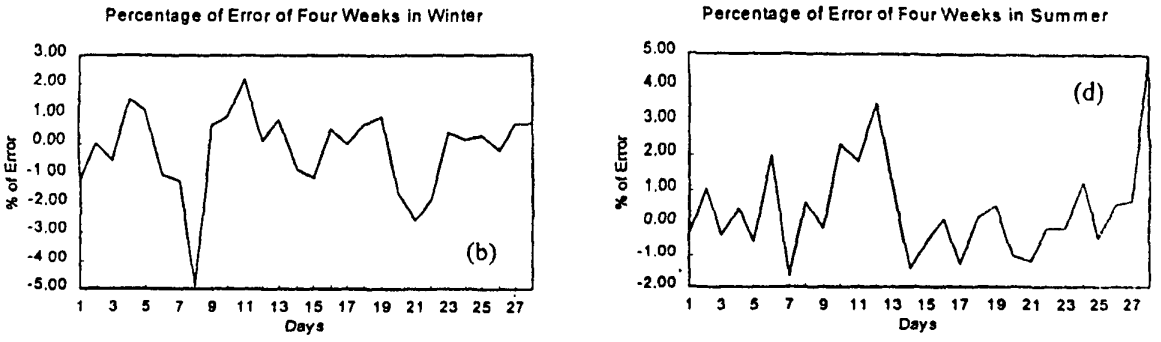


Fig. 3. Corresponding % of Error of (b) Winter (d) Summer.

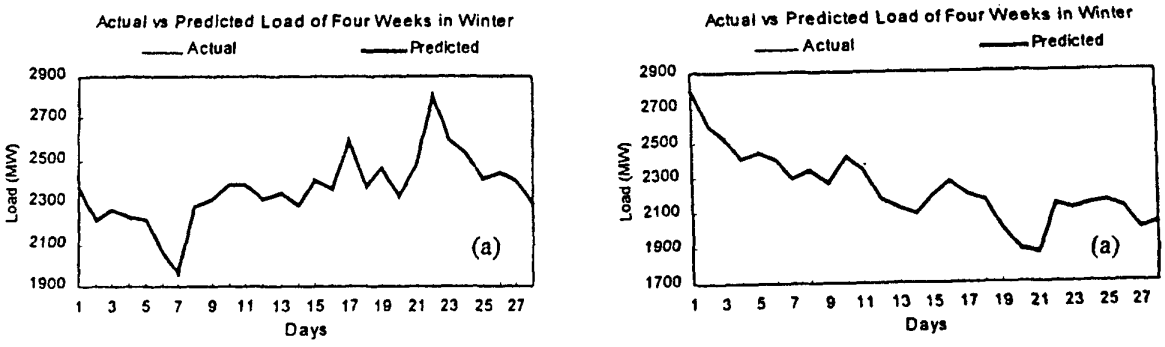


Fig. 4. Actual vs Predicted Average load in Winter 24-hours (b) 168-hours ahead.

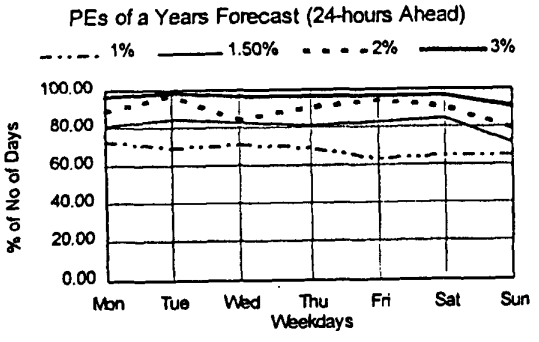


Fig.5(a) Percentage of days a particular weekday Having different PE values over a year.

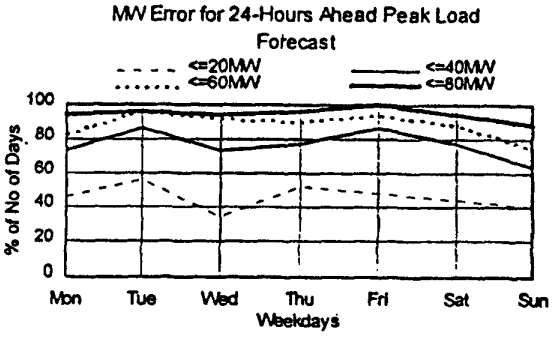
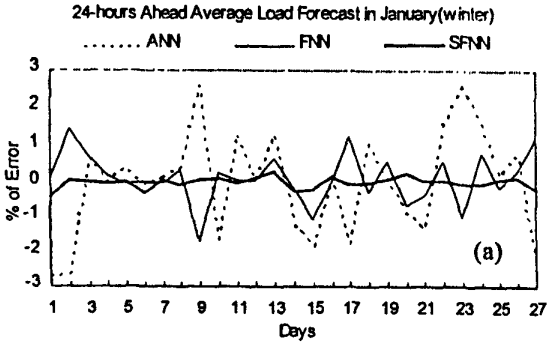


Fig. 5(b) Percentage of number of days a particular Weekday having different Mega Watt Error over a year for 24-hours ahead Peak load forecast



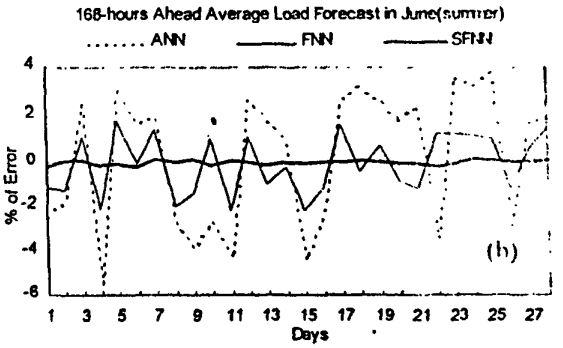


Fig. 6. Comparison of PEs between ANN, FNN, and SFNN Models in Average load forecast; (a) 24-hours Ahead (b) 168-hours Ahead