A real-time short-term peak and average load forecasting system using a self-organising fuzzy neural network

P.K. Dash[a], H.P. Satpathy[b], A.C. Liew[a]

[a]*Department of Electrical Engineering, National University of Singapore, Singapore*
[b]*Department of Electrical Engineering, Regional Engineering College, Rourkela, India*

Abstract

This paper presents a self organising fuzzy-neural-network-based short-term electric load forecasting system for real-time implementation. A learning algorithm is devised for updating the connecting weights as well as the structure of the membership function of the network. The number of rules in the inferencing layer is optimised; this in turn optimises the network structure. The proposed algorithm exploits the notion of error back-propagation. The network is initialised with random weights. Experimental results of the system are discussed from a practical standpoint. The system accounts for seasonal and daily characteristics, as well as abnormal conditions, holidays and other conditions. It is capable of forecasting load with a lead time of one day to one week. The adaptive mechanism is used to train the network on-line. The results indicate that the proposed load forecasting system is robust and yields more accurate forecasts. Furthermore, it allows greater adaptability to sudden changes, compared with simple neural-network or statistical approaches. Extensive studies have been performed for all seasons, and some of them are presented in this paper. The new algorithm is tested with a typical load date of the Virginia Utility, and produces a very robust and accurate forecast, with a Mean Absolute of Percentage of Error (MAPE) mostly less than 1.8% for 24-hours-ahead peak load forecast, and 1.6% for a 168-hours-ahead forecast.

Keywords: Real-time implementation; Short-term electric load; Forecasting system; Self organising fuzzy neural network

1. Introduction

Accurate short-term (one day to one week ahead) load forecasts represent a great saving potential for the economic and secure operation of power systems [1]. Forecasts of peak and average loads are necessary for scheduling functions such as hydro-thermal co-ordination, Interchange evaluation, fuel allocation, maintenance and security assessments. So the development of an accurate, fast, simple and robust short-term load forecasting methodology is of importance to both electric utilities and their customers, and in turn introduces higher reliability and better management.

For nearly two and half decades researchers have studied this problem of improving load forecast accuracy, and the various models proposed have varied in the complexity of their functional form and estimation procedures.

The most commonly used techniques include statistically based techniques, expert-system-based methods and artificial neural-network algorithms (ANN). Time series [2] and regression [3] are two major classes of conventional statistical algorithms, and have been applied successfully in this field for many years. However, these techniques do not produce a sufficiently accurate forecast, and the accuracy deteriorates for longer variations of non-stationary load and weather variables.

Over the past few years, artificial neural networks (ANN) have been proposed as a powerful tool for short-term load forecasting [4–6] problems. It is known that artificial neural networks do not require any explicitly defined relationship between input and output variables. The corresponding mapping between input and output is obtained using a training algorithm. The ANN modelling needs only the selection of input variables, thus avoiding the difficulties associated with conventional modelling processes.

Peng *et al.* [5] proposed a minimum-distance-based strategy to identify the appropriate historical patterns

of load and temperature for training the network. A partially connected network [4] 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 variables. While all these prior works focus on the application of back-propagation training algorithms to train and update the model parameters, Peng *et al.* [2, 7] explored the effect of a (model development) ARMA Box-Jenkins model, and make use of the Widrow-Hoff delta rule [8] to update the model parameters.

Fuzzy systems have become another research area that is receiving increased attention. The pioneering work of Zadeh [9] in fuzzy set theory has inspired work in many research areas, including load forecasting [10, 11]. Many fuzzy expert systems have been developed for short-term load forecasting (STLF), and the heuristic features of an expert system provide an excellent approach to this. Fuzzy-logic-based expert systems for load forecasting require the 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 model building [12, 13]. Usually, one of two approaches is adopted for implementing these FNNs. One of the approaches handles fuzzified input data. In the second case the weights of the network are computed, based upon a fuzzy rule base, but without fuzzifying the input data [14, 15].

Neural-network-based models for forecasting problems are less complex than fuzzy-logic-based systems. However, the simplicity is achieved at the cost of an explicitly defined relationship between the individual inputs and the overall model parameters. Fuzzy-logic-based systems allow some insight into the model parameters with the help of membership functions and rules.

The objectives of the present approach is to study a self-organising fuzzy-neural-network (FNN) that combines the self-organising capability of neural networks with fuzzy-logic reasoning attributes. The network modelling starts with a random set of weights, and hence an arbitrary set of fuzzy sets. The network is initialised with a sufficiently large number of rule-nodes, which subsequently get optimised. An adaptive mechanism for weight updating has been devised, together with the updating of the associated parameters of the fuzzy membership function. The training algorithm exploits the notion of error back-propagation. The approach presented in this paper is highly flexible, and Sundays and holidays can easily be included. This is done by treating the Sundays and Saturdays separately from weekdays (Monday to Friday). The load curve on Sundays and public holidays is similar in nature, with small deviations due to load use pattern on holidays. So, important holidays like Christmas and New Year are treated as Sundays. Other holidays, including Good Friday, Memorial Day, Independence Day, Labour Day, the days preceding Christmas and New Year, and the day after, are treated as Saturdays.

The approach presented in this paper is amenable to real-time implementation, as hourly or daily adaptation of model parameters can be undertaken. The network is trained and tested using typical load-data from the Virginia Utility, USA. A comparison has also been made with the ANN and FNN [22] models for short-term load prediction, using the above-mentioned load data.

2. Overview of the proposed approach

One of the salient aspects of a fuzzy inference system (FIS) is the determination of the knowledge base (KB), which consists of the following subsystems:

- A mechanism for developing the membership functions.
- A fuzzy reasoning mechanism.
- The number of rules and the rule-base.

This is supported by an adaptive mechanism that allows learning to take place. This section presents the fuzzy inference system (FIS), along with the governing equations. The network, consisting of *input*, *fuzzification*, *inferencing* and *defuzzification* layers, is shown in Fig. 1. The network consists of N input variables, with N neurons in the input layer and R number of rules, with R neurons for inferencing; thus the number of neurons in the fuzzification layer is $N \times R$. The inputs to the model are chosen as follows:

$$X = \begin{bmatrix} Y_a(k-\Delta+1) \\ Y_a(k-2\Delta+1) \\ Y_a(k-n\Delta+1) \\ T_a(k+1) \\ T_a(k-\Delta+1) \\ T_a(k-2\Delta+1) \\ T_a(k-n\Delta+1) \end{bmatrix} \quad (1)$$

and the output $Y = Y_a(k)$, where $Y_a(k)$ represents the load $T_a(k)$ is the temperature at the k^{th} hour and Δ is the time step ahead for which forecasting is desired.

The past loads are taken into account to improve upon the prediction capabilities; the notion being similar to that of auto-regression [3, 16]. The temperatures are included to reflect the weather-sensitivity of the load. It should be noted that the $(k+1)^{th}$ element of the input vector is the *a priori* information on the temperature at the hour at which load forecasting is to be done.

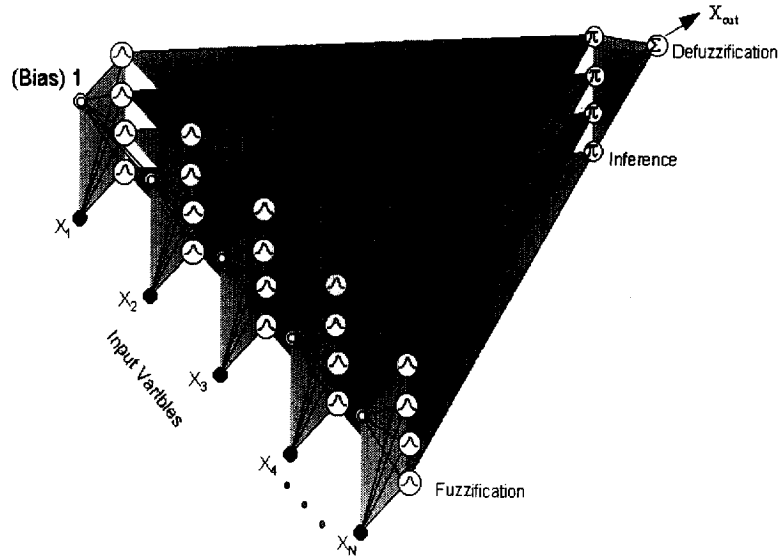

Fig. 1. Load forecasting model using a self-organising fuzzy neural network (SFNN).

The input to the fuzzification layer is a weighted version of the input variables, which can be represented as

$$\bar{\eta} = W_1^T X + W_0. \tag{2}$$

The set of weights between the input and fuzzification layers is given by:

$$W = \{W_0, W_1\} = \left\{\{w_{ij0}, w_{ij1}\} : i = 1, \ldots, N; j = 1, \ldots, R\right\} \tag{3}$$

alternatively, the individual elements of the fuzzification (2^{nd}) layer are given by

$$\eta_{ij} = w_{ij}x_i + w_{ij0}. \tag{4}$$

The output of each neuron in the *fuzzification layer* is a fuzzy membership, corresponding to a particular choice of variables. The activation (membership) function used for this fuzzification layer is given by

$$f(\eta_{ij}) = \exp\left(-|\eta_{ij}|^{\gamma_{ij}}\right), \tag{5}$$

where γ_{ij} is in the range of $1.0 \leq \gamma_{ij} \leq 5.0$. During training, γ_{ij} is updated in this range, which is initially set at 2. The output of the fuzzification layer can be expressed as

$$\mu_{ij} = f(\eta_{ij}) = \exp\left(-|w_{ij}x_i + w_{ij0}|^{\gamma_{ij}}\right), \tag{6}$$

where μ_{ij} is the value of the fuzzy membership function of the i^{th} input variable corresponding to the j^{th} rule. The activation function in the *inference layer* (3^{rd}) uses multiplicative inference. The output of this layer is given by

$$\mu_j(x_1, x_2, \ldots, x_N) = \prod_i^N \mu_{ij}(x_i). \tag{7}$$

The *defuzzification layer* (4^{th}) has the connecting weights (v_j) to the output from the inferencing layer, and these weights signify the strength of each rule in the output of the model. The output X_{out}, is given as

$$X_{out}(x_1, x_2, \ldots, x_N) = \sum_{j=1}^{R} v_j \mu_j(x_1, x_2, \ldots, x_N) = \sum_{j=1}^{R} v_j \prod_i^N \exp\left(-|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\right). \tag{8}$$

3. Training and model adjustment

3.1. Training procedures

The network is trained to minimise an objective function. The performance index (PI) to be minimised is the *Mean-Square-Error* (MSE), given by $PI = (y_d(k) - y(k))^2$, where $y_d(k)$ is the desired output and $y(k)$ is the model output. The model parameters of the proposed FIS are updated using the notion of error back-propagation.

The aim is to optimise the performance index $PI = (y_d(k) - y(k))^2$. The W_{ij1}, W_{ij0} and γ_{ij} are updated till some stopping criterion is reached. The changes in W_{ij1}, W_{ij0} and γ_{ij} are computed by differentiating the PI with respect to the required parameters. Define

$$\varepsilon_k^2 = \left(y_d(k) - y(k)\right)^2, \tag{9}$$

$$\hat{\nabla}_{k1} = \frac{\partial \varepsilon_k^2}{\partial W_1} \text{ and } \hat{\nabla}_{k0} = \frac{\partial \varepsilon_k^2}{\partial W_0},$$

$$\hat{\nabla}_{k1} = \frac{\partial \varepsilon_k^2}{\partial W_1} = 2\varepsilon(k)\frac{\partial \varepsilon(k)}{\partial W_1},$$

$$\frac{\partial \varepsilon(k)}{\partial w_{ij1}} \hat{\equiv} \nabla W_{ij1} = \frac{\partial (y_d(k) - y(k))}{\partial w_{ij1}} = \frac{\partial y(k)}{\partial w_{ij1}}.$$

From equation (7)

$$\frac{\partial y(k)}{\partial w_{ij1}} = \frac{\partial}{\partial w_{ij1}}\left[-\sum_{j=1}^{R} v_j \prod_{i=1}^{N} \exp\left(-|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\right)\right]$$

$$= v_j\left(\prod_{\substack{i=1\\ i\neq j}}^{N} \exp\left(-|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\right)\right)\cdot\frac{\partial}{\partial w_{ij1}}$$

$$\left[-\sum_{j=1}^{R}\exp\left(-|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\right)\right]$$

$$= v_j\mu_j\gamma_{ij}\cdot|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}-1}x_i. \tag{10}$$

Now, the update rule for W_1 can be written as

$$w_{ij1}(k+1) = w_{ij1}(k) - 2\alpha(k)\varepsilon(k)\cdot\nabla W_{ij1}. \tag{11}$$

Similarly, it can be shown that

$$\frac{\partial\varepsilon(k)}{\partial w_{ij0}} \hat{=} \nabla W_{ij0} = v_j\mu_j\gamma_{ij}\cdot|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}-1} \tag{12}$$

and the update rule for W_0 can be written as

$$w_{ij0}(k+1) = w_{ij0}(k) - 2\alpha(k)\varepsilon(k)\cdot\nabla W_{ij0}. \tag{13}$$

The update rule for γ_{ij} is derived as follows:

$$\frac{\partial\varepsilon(k)}{\partial\gamma_{ij}} = \frac{\partial\left(y_d(k) - y(k)\right)}{\partial\gamma_{ij}} = \frac{\partial y(k)}{\partial\gamma_{ij}}$$

$$= \frac{\partial}{\partial\gamma_{ij}}\left[-\sum_{j=1}^{R} v_j\prod_{i=1}^{N}\exp\left(-|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\right)\right]$$

$$= v_j\left(\prod_{\substack{i=1\\ i\neq j}}^{N}\exp\left(-|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\right)\right)\cdot\frac{\partial}{\partial\gamma_{ij1}}$$

$$\left[-\sum_{j=1}^{R}\exp\left(-|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\right)\right]$$

$$= v_j\mu_j\cdot|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\cdot\log_e\left(|w_{ij1}x_i + w_{ij0}|\right).$$

Hence,

$$\nabla\gamma_{ij} = v_j\mu_j\cdot|w_{ij1}x_i + w_{ij0}|^{\gamma_{ij}}\cdot\log_e\left(|w_{ij1}x_i + w_{ij0}|\right) \tag{14}$$

and

$$\gamma_{ij}(k+1) = \gamma_{ij}(k) - 2\alpha(k)\varepsilon(k)\cdot\nabla\gamma_{ij}. \tag{15}$$

Hence, the learning algorithm can be summarised as

$$\begin{aligned} w_{ij0}(k+1) &= w_{ij0}(k) - 2\alpha(k).\varepsilon(k).\nabla w_{ij}(k),\\ w_{ij1}(k+1) &= w_{ij1}(k) - 2\alpha(k).\varepsilon(k).x_i(k).la w_{ij}(k),\\ \gamma_{ij}(k+1) &= \gamma_{ij}(k) - 2\alpha(k).\varepsilon(k).bla\gamma_{ij}(k),\\ v_j(k+1) &= v_j(k) - 2\alpha(k).\varepsilon(k).\mu_j(k), \end{aligned} \tag{16}$$

where

$$\varepsilon(k) = y_d(k) - y(k),$$

$$\nabla w_{ij}(k) = \gamma_{ij}(k).\mu_n(k).v_n(k).Q(k)^{\gamma_{ij}(k)-1} \text{ and}$$

$$\nabla\gamma_{ij}(k) = \mu_n(k).v_n(k).Q(k)^{\gamma_{ij}(k)}\log_e(Q(k)).$$

$Q(k)$ is defined as

$$Q(k) = |w_{ij1}(k).x_i(k) + w_{ij0}(k)|.$$

The value of α is also tuned according to the following update equation:

$$\alpha(k) = \alpha(k-1) + 0.2\varepsilon(k) + 0.1\dot{\varepsilon}(k), \tag{17}$$

where the different error $\dot{\varepsilon}(k) = \varepsilon(k) - \varepsilon(k-1)$.

The training is continued until the following stopping criterion is reached

$$\sum_{k=i}^{i-50} PI_k \leq \varepsilon, \quad \text{or} \quad k \geq I_{\max},$$

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

If, after completion of training, the performance of the model is not found to be satisfactory, the weights are reinitialised, the number of rules is increased and the above-mentioned procedure is repeated.

3.2. *Model optimisation*

After achieving a suitable level of performance over the entire training range, the following optimisation procedure is followed to optimise the size of the network:

i. The *rule-nodes* that produce an output close to *zero* over the entire training set are sought. Obviously, these nodes do not contribute significantly to the model output. Hence, these rule-nodes can be safely omitted without affecting the model performance.

ii. Again, inputs for which the fuzzy membership is unity or close to unity over the entire training set, do not play any role in the actual model. These input variables can be traced down, and are omitted from the network.

The model is tested without any retraining and, if the performance is still satisfactory without the above-mentioned *rule-nodes* and *inputs*, then a model with reduced size has been obtained. This makes the network computationally more efficient. However, if the performance of the network is not found to be suitable, then the network has to be retrained with the reduced set of rules and inputs. In the simulations described here 5 input-nodes (M) and 16 rule-nodes (R) were ultimately retained for the final model.

3.3. Simulation results and evaluation

While operating in a real-time environment, it is imperative that the load forecasting system should be able to adapt to changing conditions. This objective is achieved by the following methodology.

For daily adaptation, the optimised weights are used from the training set to forecast the first day's load. After the end of the day of the forecast model, parameters are updated until the error becomes insignificantly small. The number of iterations required for this purpose is extremely small. Once the new weights have been established, the forecast for the next day is attempted, using this weight vector and the new set of input data for the day.

For identifying the parameters of the load model, data related to weekdays (Monday through Friday) are treated separately from the weekends (Sundays or Saturdays). For this purpose the data related to weekends are separated from the load database for the purpose of identification of the parameters of the weekday load model during network training. In a similar way, the parameters of the weekend load model are identified by using the appropriate load database. Similar load patterns are used during the training of the network to obtain a faster convergence.

As the weekends are excluded from the first set of the database, the weight vector obtained after the forecast of a Friday's load is used to predict the load on Monday. For one-week-ahead forecasts, the adaptation is done once a week, i.e. at the end of the week, when the entire load profile for the whole week is available.

However, until some appreciable change in the load pattern is encountered, the network is left to operate in prediction mode. Suppose that at the k^{th} time-step it is observed that the error in prediction is showing a gradual increasing trend; this indicates a substantial change in the load pattern. In order to reduce the error within the desired performance level, the network is retrained using available data up to the $(k-1)^{th}$ time-step. After training, the network is again used for forecasting.

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

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

where N is the number of patterns in the date set used to evaluate the forecasting capability of the model.

The standard deviation (SD) of the absolute relative error is as follows:

$$SD = \left[\frac{1}{T}\sum_{t=t_0}^{T+t_0}(\%err_t - MAPE)^2\right]^{1/2}$$

where T is the number of time samples, and the MAPE is computed over the same range of t. The err_t is the absolute error relative to the peak load for the day.

Abnormal weather and system conditions, such as thunderstorms or transmission outages, are treated as abnormal events with bad real-time readings, and are not considered in the forecasting models. The influence of standard holidays is also considered in the real-time forecast, and is treated separately, along with weekends.

The special holiday data occurring in the past and the weekend data are used to train the model before a prediction for a holiday is attempted. The collection of similar days from the past ensures that the characteristics of only that type of holiday are reflected in the data set.

The performance of the proposed model has been compared with those of some existing neural tools and time-series approaches [1, 18–22]. It has been found that this method performs better than the other models, especially those that use back-propagation as the training algorithm. To evaluate the performance of the proposed network architecture, it was used to forecast both one-day and one-week-ahead peak and average loads.

This section also describes an example of the authors' early work, which uses a fuzzy neural network (FNN) modelled as a five-layer feed-forward neural network [22]. The input vector to the FNN consists of differences in weather parameters between the present and forecast instant. The output of the FNN gives the load correction which, when added to the past load, gives the forecast load. Fig. 2 shows the proposed model of the FNN using the ANN architecture. The FNN has a total of five layers. Nodes at layer one are the input linguistic nodes. Layer 5 is the output layer and consists of two nodes, one for the actual load correction e_{LC} and the other the desired load correction $\hat{e}_{LC}$. Nodes in layers two and four are term nodes which act as membership functions to represent the term sets of the respective linguistic variables. Each node in layer three represents the preconditions of the rule nodes, and layer-four links define the consequences of the rules.

In the simulations, data from a utility in the state of Virginia, USA are used. A mathematical software package, MATLAB, is used for obtaining load forecasts.

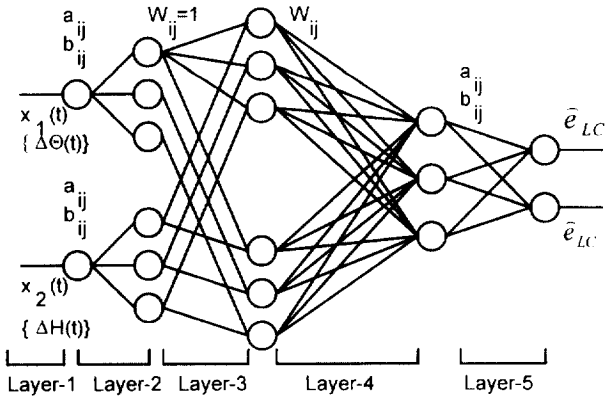

Fig. 2. Load forecasting model using a fuzzy neural network (FNN).

4. Load forecasting results

The 24-hours-ahead peak load prediction over four weeks in winter is shown in Fig. 3(a). The corresponding percentage of error (PE) is shown in Fig. 3(c). It is observed from these figures that the predicted output closely follows the actual load pattern, and the error is mostly within $\pm 3\%$. Similarly, the 24-hours-ahead peak load forecast and its corresponding error values for summer season are shown in Fig. 3(b) and (d). From Fig. 3(a)–(b) it is clear that the prediction in winter is better than in summer. This can be attributed to abrupt variations in the weather pattern in the summer season.

In some utilities a week-ahead forecast (168-hours) is also desired. The 168-hours-ahead predictions over four weeks in winter and summer are therefore presented in Fig. 4(a) and (b), respectively. These results are marginally better than those from the 24-hours-ahead forecasts. This reinforces the fact that the load patterns of similar day types bear a greater resemblance. For example, a Tuesday's load pattern is more similar to that of the previous Tuesday than to that of the immediately preceding Monday.

The 24-hour average load forecasts over four weeks in winter and summer are shown in Fig. 5(a) and (b) respectively, and the 168-hours-ahead average load forecasts are given in Fig. 6(a) and (b). It is observed that the average load forecast is better than the peak load forecast. This phenomenon is due to the averaging effect of the load pattern.

The comparison of the percentage of errors (PEs) between the ANN, FNN and SFNN models for average load forecasting is shown in Fig. 7(a) and (b). Fig. 7(a) gives the PE for 24-hours-ahead forecasts in the month of January (winter) and Fig. 7(b) gives the comparison for 168-hours-ahead forecasts in the month of June (summer). Similarly, Fig. 8 shows the comparison for 168-hours-ahead peak load forecasts in

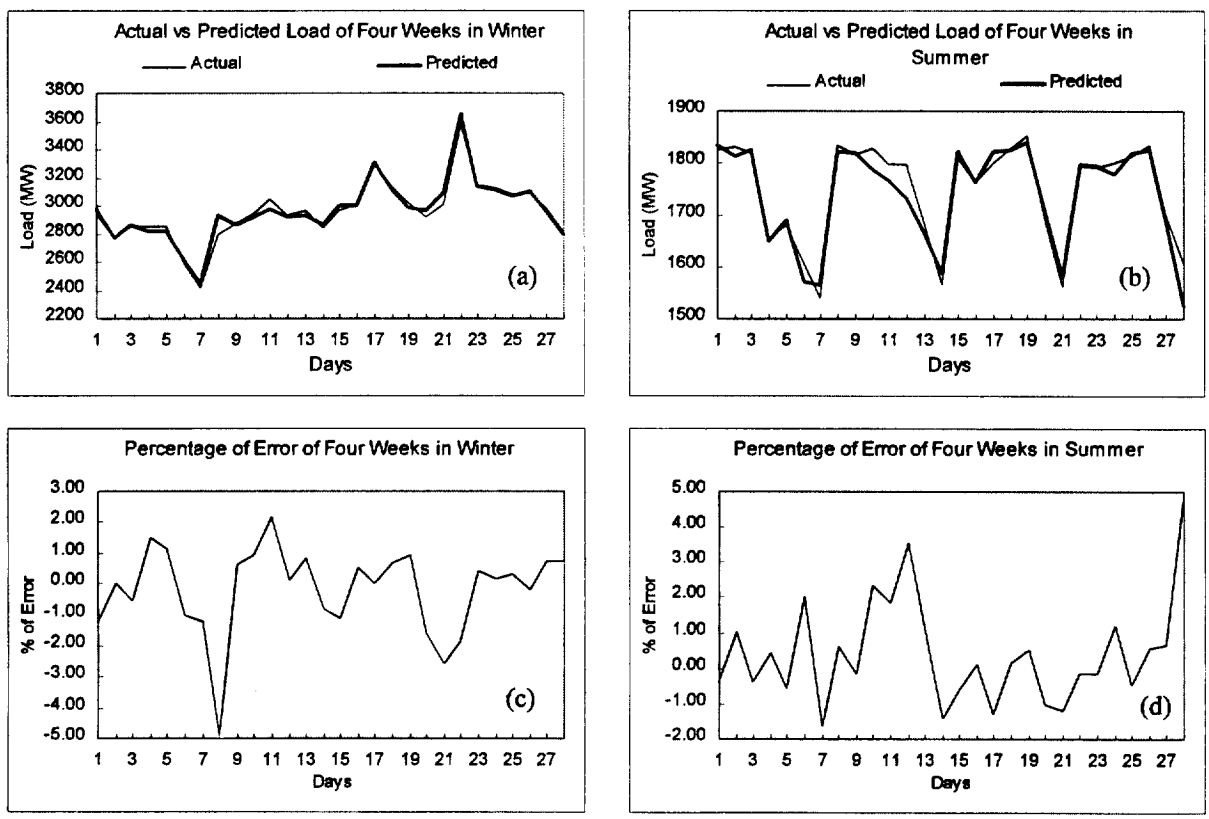

Fig. 3. 24-hours-ahead actual vs predicted peak load. (a) Winter (b) Summer; corresponding percentage of error (c) Winter (d) Summer.

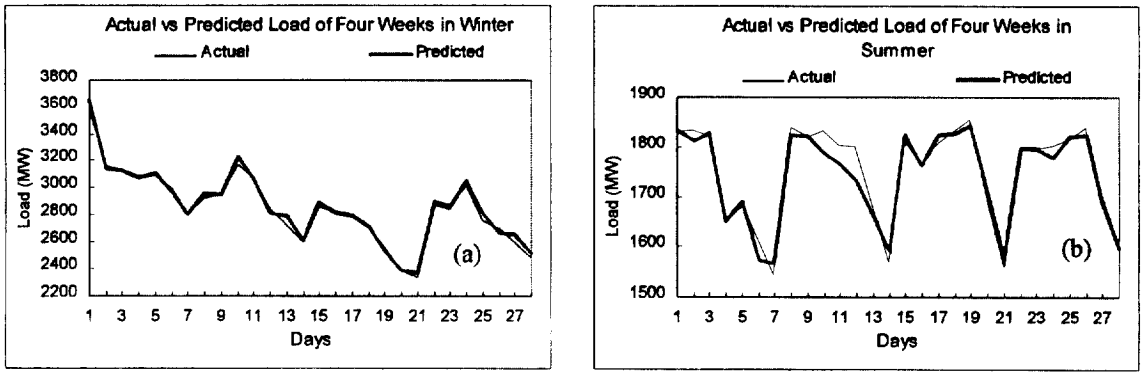

Fig. 4. 168-hours-ahead actual vs predicted peak load, (a) Winter (b) Summer.

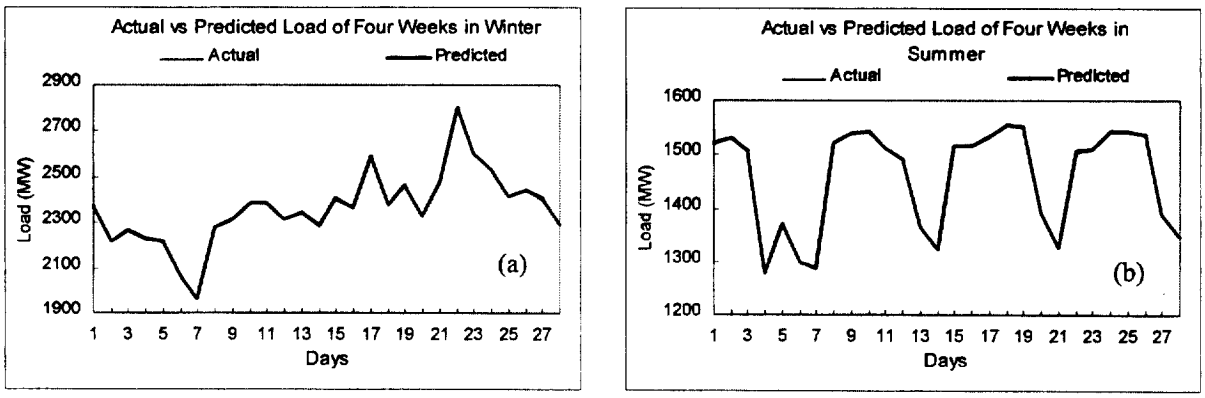

Fig. 5. 24-hours-ahead actual vs predicted average load. (a) Winter (b) Summer.

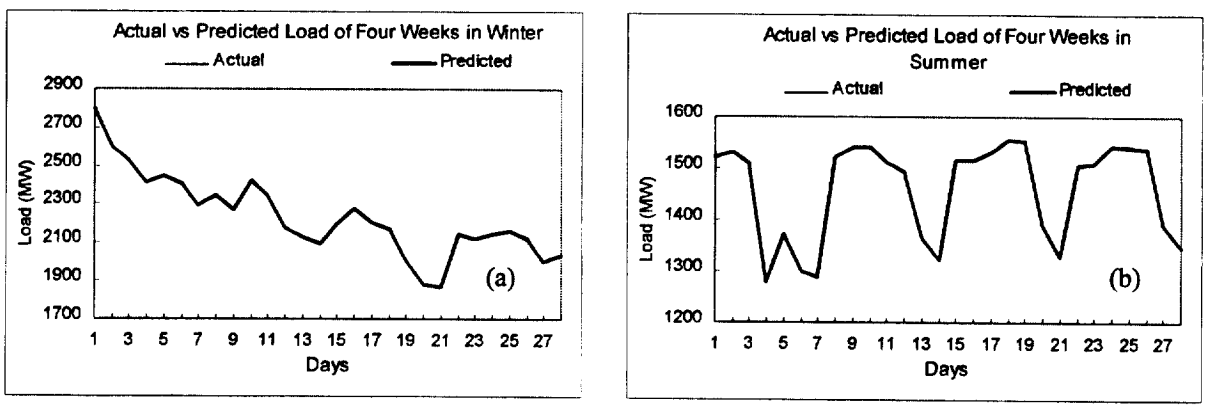

Fig. 6. 168-hours-ahead actual vs predicted average load. (a) Winter (b) Summer.

the month of January. The figure shows the improved performance achieved by the SFNN model in terms of faster convergence and improved overall accuracy, in comparison with the ANN and FNN approaches.

The results in Fig. 9 show the percentage of the number of days on which different week-days lie within a certain percentage of error over the whole year. Here, in the results for 24-hours-ahead peak load forecasts, it is seen that—out of all the days, categorised into individual week-day types—70% of the days are within a PE value of 1.0, 80% within a PE of 1.5 and 90% within a PE of 2.0, which is much better than the results obtained by Mohammed *et al.* [8], Khotanzad *et al.* [19] and Bakirtzis *et al.* [21].

Similarly, the percentage of the days of each week-day type, having different levels of megawatt (MW) error (difference between actual and predicted load), is shown in Fig. 10. The results in Fig. 10(a) show that almost 50% of the days of any week-day type lie within a 20 MW error range, 70% are within 40 MW

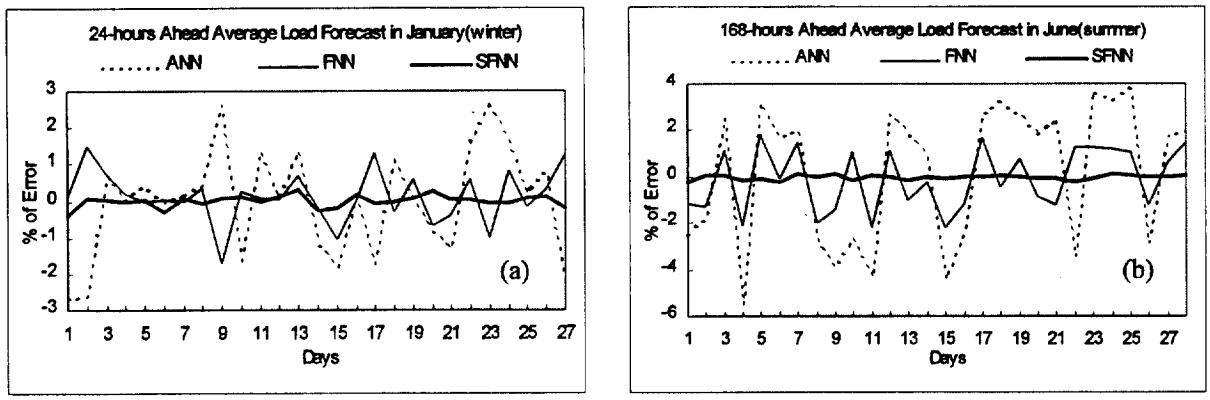

Fig. 7. Comparison of PEs between ANN, FNN and SFNN models in average load forecasts: (a) 24 hours ahead (b) 168 hours ahead.

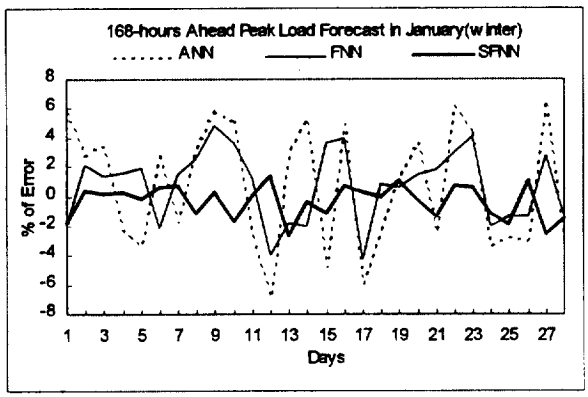

Fig. 8. Comparison of PEs between ANN, FNN and SFNN models in peak load forecasts: 168 hours ahead.

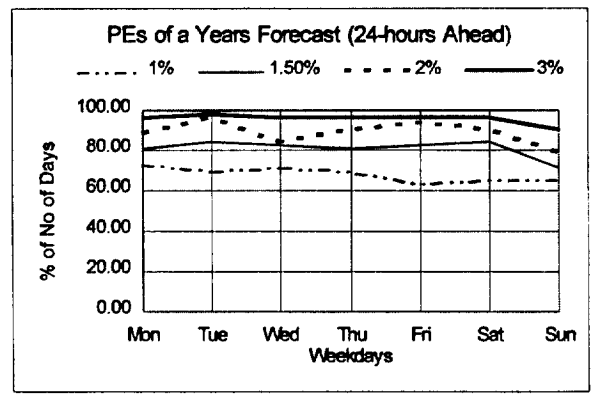

Fig. 9. Percentage of days on which a particular weekday had different PE values over a year.

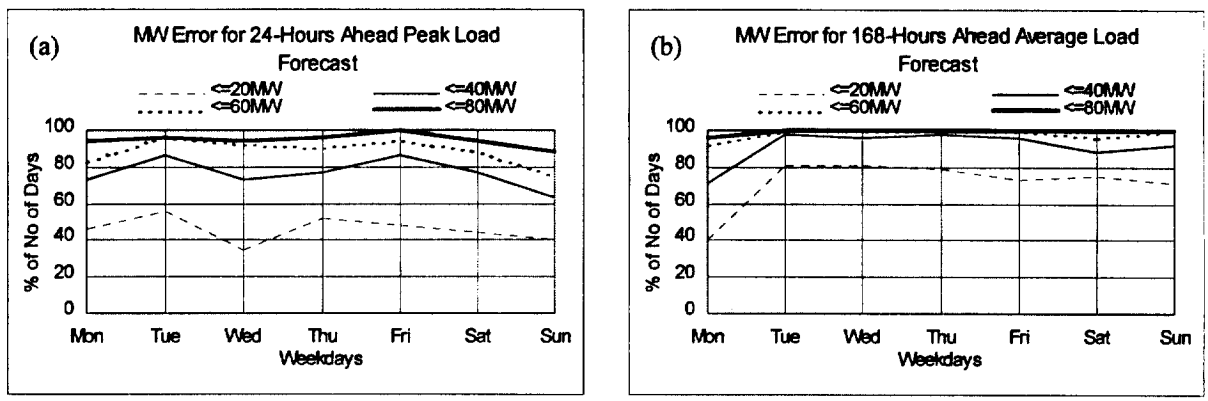

Fig. 10. Percentage of number of days on which a particular weekday had different Mega Watt errors over a year. (a) 24-hours-ahead peak load forecasts, (b) 168-hours-ahead average load forecasts.

and 90% within a 60 MW range for 24-hours-ahead peak load forecasting. For 168-hours-ahead average load forecasting almost 80% of the days for any given weekday type fall within a 20 MW range. This is shown in Fig. 10(b).

Fig. 11(a) and (b) show the mean absolute percentage of error (MAPE) calculated over each week for 24-hours-ahead as well as 168-hours-ahead forecasts for 50 weeks of a year. It can be observed from these figures that the weekly MAPE values for average load forecasts mostly occur within MAPE values of 0.05 to 0.15, and for peak load within 0.5 to 1.5, which can be compared with results obtained by Mohammed *et al.* [18] and Khotanzad *et al.* [19].

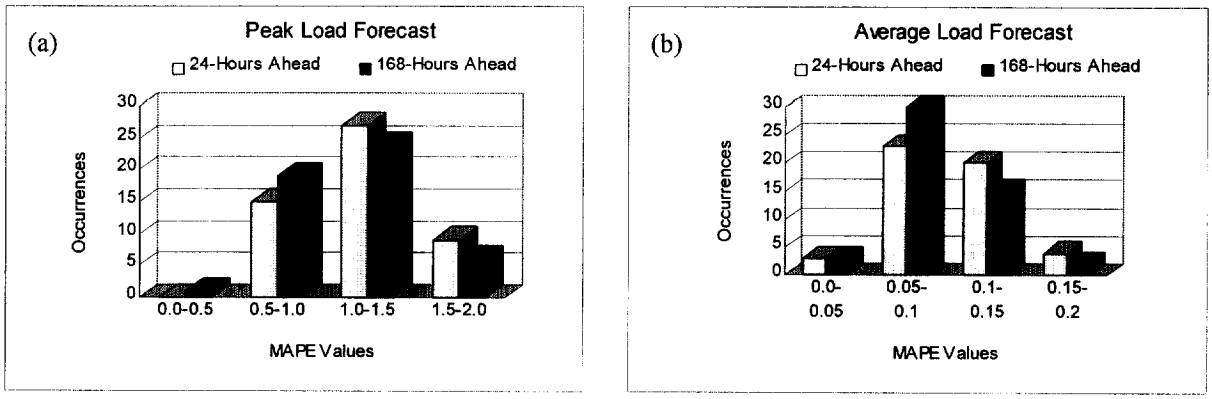

Fig. 11. Occurrences of weekly MAPE values; (a) Peak load (b) Average load.

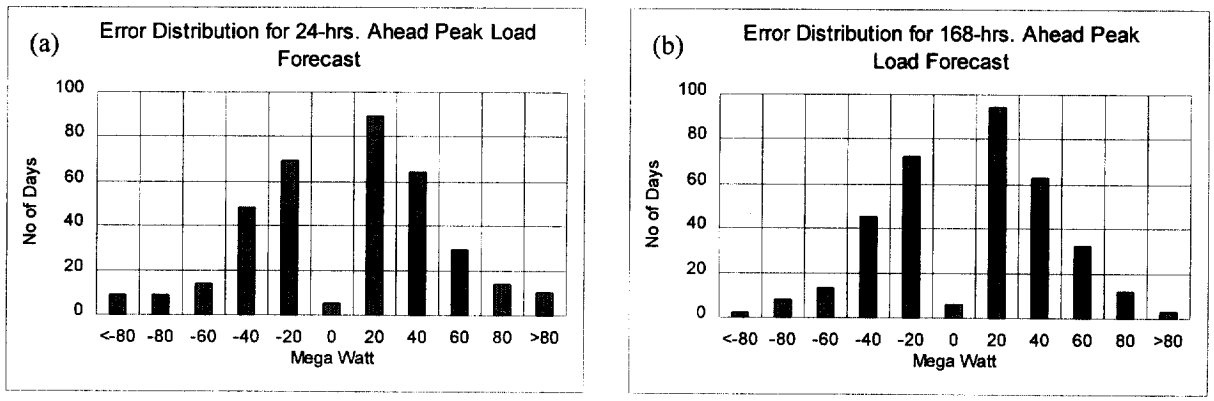

Fig. 12. Error distribution of peak load forecast over a year, (a) 24 hours ahead and (b) 168 hours ahead.

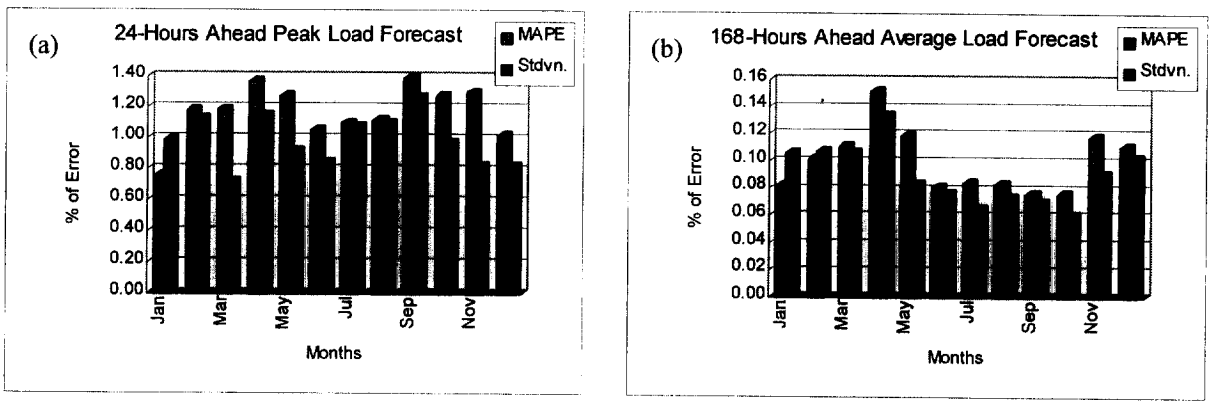

Fig. 13. MAPE and standard deviation for each month over a one-year period for (a) 24-hours-ahead peak load forecast, (b) 168-hours-ahead average load forecast.

The FNN results in terms of megawatt error distribution for all the days over a year is shown in Fig. 12. Fig. 12(a) and (b) show the results of both 24-hours-ahead and 168-hours-ahead peak load forecasts, respectively. Taking the total number of days as 100, it was found that this result is much better than those obtained using BP (back-propagation) in an ANN by Mohammed *et al.* [18].

The MAPE and standard deviation (SD) of each month over a year are shown pictorially in Fig. 13. As shown in the figure, the maximum value of the MAPE is below 1.4% for 24-hours-ahead peak load forecasts

(Fig. 13(a)), and the MAPE is within 0.16% for 24-hours-ahead average load forecasts (Fig. 13(b)). This is significant in comparison to other approaches e.g., those of Dash *et al.* and Paraman *et al.* [22, 23].

5. Conclusion

This paper has addressed the problem of load forecasting using a self-organising fuzzy neural network (SFNN) structure, which gave a significant improvement in performance in comparison to ANN and FNN models. The weights in the different layers of the network are optimised using a novel updating algorithm. The network performs satisfactorily, starting from an initial set of random weights. Thus the problem of a proper choice of the initial weights is avoided. The efficacy of the network is validated by selecting a practical load pattern in summer and winter. As the summer load pattern is extremely 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 forecasting, and within 0.05–0.15% for average load forecasting, which is a very good result in comparison to those obtained by BP neural networks.

References

Hargan, M.T., Behr, S.M., 1987. The time series approach to short-term load forecasting. IEEE Trans. on PWRS vol. PWRS-2 (no. 3), 785–791.

Box, G.E., Jenkins, G.M., 1976. Time Series Analysis Forecasting and Control. Holden-Day, San Fransisco.

Papalexopoulos, A.D., Hesterberg, T.C., 1990. A regression based approach to short-term system load forecasting. IEEE Trans. on PWRS vol. PWRS-5 (no. 4), 1535–1544.

Chen, S.T., Yu, D.C., Moghaddamjo, A.R., 1992. Weather sensitive short-term load forecasting using nonfully connected artificial neural network. IEEE Trans. on Power System vol. 7 (no. 3), 1098–1105.

Peng, T.M., Hubele, N.F., Karady, G.G., 1992. Advancement in the application of neural networks for short-term load forecasting. IEEE Transactions on PWRS vol. 7 (no. 1), 250–257.

Park, D., El-Sharkawi, M., Marks, R., Atlas, L., Damborg, M., May. Electric load forecasting using an artificial neural network. IEEE Trans. on Power Systems vol. 6 (no. 21991), 442–449.

Peng, T., Hubele, N.F., Karady, G., 1993. An adaptive neural network approach to one-week ahead load forecasting. IEEE Trans. on PWRS vol. 8 (no. 3), 1195–1202.

Widrow, B., Lehr, M.A., 1990. 30 years of adaptive neural networks : perceptron, madaline, and backpropagation, in Proc. IEEE vol. 78 (no. 9), 1415–1492.

Zadeh, L., 1973. Outline of a new approach to the analysis of complex systems and decision process. IEEE Transaction on Systems, Man and Sybernatics vol. SMC-3 (no. 1), .

Hsu, Y., Ho, K., 1992. Fuzzy expert system: An application to short-term load forecasting. IEE Proceeding-C vol. 139, 471–477.

Rahman, S., Bhatnagar, R., 1988. An expert system based algorithm for short-term Load forecast. IEEE Trans. on PWRS vol. 3 (no. 2), 392–399.

Lin, Y., Cunningham, G.A., May. A new approach to fuzzy-neural system modelling. IEEE Transactions on Fuzzy Systems vol. 3 (no. 21995), 190–198.

Horikawa, S., Furuhashi, T., Uchikawa, Y., 1992. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. IEEE Trans. Neural Networks vol. 3 (no. 5), 801–814.

Hayashi, Y., Bukley, J., Czogala, E., 1993. Fuzzy neural network with fuzzy signals and weights. Int. J. Intell. Syst. vol. 8, 527–537.

Ishibuchi, H., Fujioka, R., Tanaka, H., 1993. Neural networks that learn from fuzzy if-then rules. IEEE Trans. fuzzy Systems vol. 1 (no. 2), 85–96.

Schneider, A.M., Takenawa, T., Schiffman, D.A. 1985. 24-hour electric utility load forecasting. Comparative models for electric load forecasting. chapter 7. John Wiley & Sons Ltd. pp. 87–108.

Fan, J.Y., McDonald, J.D., 1994. A real-time implementation of short-term load forecasting for distribution power system. IEEE PWRS vol. 9 (no. 2), 988–994.

Mohammed, O., Park, D., Marchant, R., Dinh, T., Tong, C. et al., 1995. Practical experiences with an adaptive neural network short-term load forecasting. IEEE Trans. PWRS vol. 10 (no. 1), 254–265.

Khotanzad, A., Hwang, R., Abaye, A., Maratukulam, D., 1995. An adaptive modular artificial neural network hourly load forecaster and its implementation at electric utilities. IEEE Trans. PWRS vol. 10 (no. 3), 1716–1722.

Dash, P.K., Liew, A.C., Rahman, S., 1995. Peak load forecasting using a fuzzy neural network. Electric Power System Research 32, 19–23.

Bakirtzis, A.G., Theocharis, J.B., 1995. Short term load forecasting using fuzzy neural networks. IEEE Trans. PWRS vol. 10 (no. 3), 1518–1524.

Dash, P.K., Satpathy, H.P., Rahman, S., 1995. Short term daily average and peak load predictions using a hybrid intelligent approach. IEEE, EMPD-95, Singapore, Catalogue No. 95TH8130.IEEE vol. 2, 565–570.

Paarmann, L.D., Najar, M.D. 1995. Adaptive online load forecasting via time series modeling. Electric Power System Research. Elsevier Science S.A. 32, pp. 219–225.