SOFT COMPUTING BASED TECHNIQUES FOR SHORT-TERM LOAD FORECASTING

V.S. KODOGIANNIS, E.M. ANAGNOSTAKIS

9-99

Preprint no. 9-99/1999

Department of Computer Science
University of Ioannina
451 10 Ioannina, Greece

Soft computing based techniques for short-term load forecasting

V.S. Kodogiannis*+, E.M. Anagnostakis
+ Department of Computer Science, University of Ioannina, Ioannina, GR-45110, Greece, Tel: +30-651-97308, Fax: +30-651-48131, Email: vassilis@cs.uoi.gr

* To whom correspondence should be addressed.

ABSTRACT: Neural Networks are currently finding practical applications, ranging from 'soft' regulatory control in consumer products to accurate modelling of non-linear systems. This paper presents the development of improved neural networks based short-term electric load forecasting models for the Power System of the Greek Island of Crete. Several approaches including radial basis function networks, dynamic neural networks and fuzzy-neural-type networks have been proposed and discussed in this paper. Their performances are evaluated through a simulation study, using metered data provided by the Greek Public Power Corporation. The results indicate that the load forecasting models developed provide more accurate forecasts compared to the conventional backpropagation network forecasting models. Finally, the embedding of the new model capability in a modular forecasting system is presented.

Key words: Short-term load forecasting, Neural Networks, Fuzzy-Neural-type Networks, Radial Basis Functions, Dynamic Neural Networks.

1. INTRODUCTION

Load forecasting has been a central and an integral process in the planning and the operation of electric utilities. Load forecasting with lead-times, from a few minutes to several days, helps the system operator to efficiently schedule spinning reserve allocation. In addition, load forecasting can provide information that is able to be used for possible energy interchange with other utilities. On the other hand, load forecasting is also useful for power system security. If applied to the system security assessment problem, valuable information to detect vulnerable situations in advance can be obtained.

The load prediction period may be a month or year for the long- and the medium-term forecasts, and day or hour for the short-term forecast. The long- and the medium-term forecasts are used to determine the capacity of generation, transmission, or distribution system additions, and the type of facilities required in transmission expansion planning, annual hydro and thermical maintenance scheduling, etc. The short-term forecast is needed for control and scheduling of power system, and also as inputs to load flow study or contingency analysis.

The short-term load forecast (one to twenty four hours) is of importance in the daily operations of a power utility. It is required for unit commitment, energy transfer scheduling and load dispatch. With the emergence of load management strategies, the short-term forecast has played a broader role in utility operations. The development of an accurate, fast and robust short-term load forecasting methodology is of importance to both the electric utility and its customers and thus introducing higher accuracy requirements.

Many techniques have been proposed in the last few decades for short-term load forecasting (STLF) [7]. The traditional techniques that have been used for the STLF include the Kalman filtering, the Box and Jenkins method, regression models, the autoregressive model and the spectral expansion technique [21].

The Kalman filter approach requires estimation of a covariance matrix. The possible high non stationarity of the load pattern, however, typically may not allow an accurate estimate to be made [20]. The Box-Jenkins method requires the autocorrelation function for identifying proper ARMA models. This can be accomplished by using pattern recognition techniques. A major obstacle here is its slow performance [33]. An interesting application to STLF, using regression procedures have been proposed by Irisarri [8]. However their Generalised Linear Square Algorithm was faced with numerical instabilities when applied to a large database. Additionally, enormous computational efforts are required to produce reasonable results.

The AR model is used to describe the stochastic behaviour of hourly load pattern on a power system. This model assumes that the load at a specific hour can be estimated by a linear combination of the previous few hours. Generally, the larger the data set, the better is the result in terms of accuracy. A longer computational time for the parameter identification, however, is required.

The spectral expansion technique utilises the Fourier series. Since load pattern can be approximately considered as a periodic signal, load pattern can be decomposed into a number of sinusoids with different frequencies. However, load patterns are not perfectly periodic. This technique usually employs only a small fraction of possible orthogonal basis set, and therefore is limited to slowly varying signals. Abrupt changes of weather cause fast variations of load patterns that result in high frequency components in frequency domain. Hence, the spectral expansion technique can not provide any accurate forecasting for the case of fast weather changes unless sufficiently large number of base elements is used [5].

Recently, with the developments of artificial intelligence, alternative solutions to the STLF problem have been proposed. Expert systems have been successfully applied to STLF [30]. This approach, however, presumes the existence of an expert capable of making accurate forecasts that will train the system. However, it is extremely difficult to transform the knowledge of an expert to mathematical rules [9]. Neural networks, on the other hand, are a more promising area of artificial intelligence since they do not rely on human experience but attempt to learn by themselves the functional relationship between system inputs and outputs.

Interest in applying artificial neural networks (ANNs) to STLF began recently. Park *et al.* proposed the use of a multi-layer network with three layers, *i.e* one input, one hidden, one output. The training of the network is performed through a simple back-propagation algorithm. Using load and weather information the system produces three different forecast variables, *i.e.* peak load, total daily load and hourly load [25]. Peng *et al.* presented a search procedure for selecting the training cases for ANNs to recognise the relationship between the weather changes and load shape. It utilises a neural network algorithm that includes a combination of linear and non-linear terms which map past load and temperature inputs to the load forecast output. The single forecast output is the total daily load [27]. Lu *et al.* used a similar feedforward neural network incorporating the previous load demands, day of week, hour of day and temperature information for load forecasting [18]. Papalexopoulos *et al.* proposed the inclusion of additional input variables such as a seasonal factor and a cooling/heating degree into a single neural network [24]. Rather questionable results are reported to Bakirtzis *et al.*, [1] where a 24-hour prediction can be obtained by an enormous, in size, three layer neural network.

Although the number of hidden nodes is much smaller the input dimension, the authors claim that this number does not significantly affect the forecasting accuracy. Lee *et al*. treat electric load demands as a non-stationary time-series and they modelled the load profile by a recurrent neural network. The forecast was made separately for weekends and weekdays using load data only [16]. Finally, an alternative technique for load forecasting using Recurrent High Order Neural Networks was considered in [10]. This type of neural network is supposed, in theory, to approximate very accurately any non-linear function, with exponential error convergence to zero [15]. Based on an interpolation procedure, the method although gives superb results in training, in testing the results were rather disappointed. The lack of generalisation can be explained by the fact that the specific method needs an enormous amount of training patterns in order to create a satisfactory input-output mapping. In this work, the training data set created from the whole year of 1994 and respectively the testing one for the first four months of 1995. The insufficiently relative small training data set, had as a consequence the poorly performance of the Recurrent High Order Neural Network model, despite its perfect training phase.

Amongst the above neural based forecasting techniques most of them generally can be classified into two categories in accordance with techniques they employ. One approach treats the load pattern as a time series signal and predicts the future load by using the already mentioned techniques. In the second approach the load pattern is considered to be heavily dependent both on weather variables (temperature, humidity, etc.) and previous load patterns. Such models that include weather variables are limited in use by problems such as inaccuracy of weather forecasts and difficulties in modelling the weather-load relationship [26].

In this paper we present algorithms which follow the time series approach. In our case ANNs trace previous load patterns and predict a load pattern using recent load data without using weather information for modelling. The developed systems identify the load model, which reflects the stochastic behaviour of the hourly load demand for the island of Crete in Greece. The balance of this paper contains a comparative study of various prediction techniques used to develop the STLF for the power system of Crete.

2. THE SHORT-TERM LOAD FORECASTING PROBLEM

This section is dealing with the application of the developed STLF models on the power system of the island of Crete. The objective of STLF is to predict the 24 hourly loads of the next day. Here, a modular-constructed forecasting system is proposed, where 24 neural blocks with a single output have to be developed and trained separately to represent the 24 hourly loads respectively. The outline of the proposed architecture is illustrated in Fig. 1.

Each neural block is fed by its previous one. Hence, step by step, 24 hour load prediction can be obtained. In this work, the training data set created from the whole year of 1994 and respectively the testing one for the first four months of 1995. Both training and testing sets were classified into 24 time series, each one corresponds to a different daily hour. Therefore, for each neural block, the following non-linear load model is proposed for one-hour ahead forecasting:

$$x(t) = F\{\, x(t-24i),\ x(t-j),\ x(t-24i-j) \,\} \qquad i = 1,\dots,p \qquad j = 1,\dots,q \tag{1}$$

where p and q indicate the number of previous days and hours respectively. The main objective of the proposed system, is the development of sufficiently accurate blocks represented the individual hourly loads.

3. MODELLING USING NEURAL NETWORKS

The investigation of Neural Networks (NNs) for system modelling has been carried out for the prediction of all 24 hours, each one trained separately, one for each hour. An assumption has been made in the case of blackouts that occurred during the whole year. All the zero load values have been removed from both training and testing set, and replaced by the mean value of the preceding and next load value.

Eq. (1) is used to provide the input variables to the individual blocks. According to Eq.(1), the requested load is forecasted not only with the load data of previous days but also with the forecasted load data for the same day at previous time steps. In the following sections only results which correspond to hours with the maximum (14:00h) and minimum (02:00h) load consumption are illustrated.

3.1 Autoregressive Linear Modelling

Although this work is primarily oriented to neural network approaches to STLF, it was assumed that it was necessary to test the well known stochastic autoregressive model to this type of data. An autoregressive (AR) model of a time series assumes that the current value of the time series is equal to a weighted average of a finite number of previous values, with the addition of a constant offset and a random noise term. The original AR model is expressed in equation

$$x_t = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \alpha_t \tag{2}$$

where $x_t, x_{t-1}, \ldots, x_{t-p}$ are the terms of time series, $\phi_1, \phi_2, \ldots, \phi_p$ are the unknown coefficients of the model , ϕ_o is a constant term , and α_t is the random noise term.

The main objective here, is the computation of the unknown parameters of the AR model $\phi_1, \phi_2, \ldots, \phi_p$. The recursive least squares method was used, in conjunction with the well known U-D algorithm [29] for optimisation of the numerical calculations. The order *p* of the AR model was approximated using the well known Akaike's information criterion. Hence, for the majority of the test cases it has been found that *p*=4. Attempts to increase the model's order for the sake of improved accuracy, resulted a severe deterioration of its performance.

3.2 Backpropagation Algorithms for STLF prediction.

Some artificial neural network architectures exhibit the capability of forming complex mappings between input and output which enable the network to approximate general non-linear mathematical functions. The multi-layer perceptron (MLP) neural network, trained by the standard back-propagation (BP) algorithm, is probably the most widely used network and its mathematical properties for non-linear function approximation are well documented [32]. The generalised delta rule is applied for adjusting the weights of the feedforward networks in order to minimise a predetermined cost error function. The rule of adjusting weights is given by the following equation:

$$w_{ij}^{P}(t+1)= w_{ij}^{P}(t)+ \eta\delta_j^{P} y_j^{P} + \alpha \Delta w_{ij}^{P}(t) \tag{3}$$

where, η is the learning rate parameter, α the momentum term and δ is the negative derivative of the total square error in respect to the neuron's output.

It is well known that the training of MLPs using that standard backpropagation algorithms plagued by a slow convergence. In this work a simple heuristic strategy, the Adaptive learning rate Backpropagation Neural Network (ABP), which relates the learning rate with the total error function E have been adopted in order to accelerate the convergence speed [34]. The algorithm uses the batch update mode, and the update rule for the weights is

$$w_{ij}^{P}(t+1)= w_{ij}^{P}(t)- \frac{\rho(E)\dfrac{\vartheta E}{\vartheta w_{ij}^{P}}}{\left\| \dfrac{\vartheta E}{\vartheta w_{ij}^{P}} \right\|^{2}} \tag{4}$$

with $\rho(E)$ is some function of the error E, which is given by

$$\rho(E)= \begin{cases} \eta \\ \eta E \\ \eta \tanh(\frac{E}{E_o}) \end{cases} \tag{5}$$

where η and E_0 are constant, non-negative numbers, representing the learning rate and the error normalisation factor respectively. The main advantage using these formulas is the dependence of the learning rate on the instantaneous value of the total squared error E. Therefore a faster convergence of the algorithm is achieved.

In order to provide sufficient information to model each neural block using an MLP, a structure with two hidden layers and 8 inputs was used. The inputs were selected from the model of Eq. (1) by setting the parameters *p* and *q* equal to 2.

A common approach taken to enable a neural network to capture the dynamics of a non-linear system, as the load forecasting problem, is to configure and train a network to represent a non-linear, autoregressive (NAR) model structure. It is natural to reflect the dynamic nature of the problem by sequential information processing. There are two approaches to processing inputs in the time domain: one is to window the inputs and then treat the time domain like another spatial domain; alternatively, to use some internal storage to maintain a current state.

In the first approach, the network input consists of a moving window of time-delayed system outputs and the network is trained to predict the system output at the next time step. It has been found that a major factor affecting the neural model prediction accuracy is the method by which data are coded in the network. This is especially in the case when the network is acting as a recurrent model. The reason for this is because any errors in the predicted output will tend to accumulate. The conventional method of conditioning the data is to re-scale and represent using a single node at the input or output layers of the network. An alternative representation, called spread encoding (SE), has been shown to enable a network to maintain a high degree of accuracy [11].

In the SE technique, each data value is represented as the mean value of a sliding Gaussian pattern of excitation over several nodes at the network input and output. A similar and reverse procedure is applied at the network output to decode the output back into the original variable range. This approach has similarities with data fuzzification techniques where the scalar dimensional space of each variable is fuzzified to a space of higher dimensions. Also, decoding of the network output using the SE method consists of computing a weighted summation of the node excitations which is analogous to the conventional centre of gravity defuzzification technique. Thus a network utilising spread encoding can be considered as a fuzzy-neural-type network. The SE method is also keeping with the heritage of neural networks from biological systems where information is often represented by the combined activity of a population of receptors, as in the retina of the eye [19]. Fig. 2 illustrates the internal architecture of this technique.

Analytically, this data conditioning method of SE consists of mapping each network variable, $x \in [x_{\min}, x_{\max}]$, onto a sliding Gaussian activation pattern of N network nodes, which includes additional nodes either side of the variable range to contain overspill resulting from the use of a mapping function with wide support. The level of activation of each node is confined to be in the range [0.1, 0.9] similar to the conventional normalisation technique. Each node is assigned a value, α_i, linearly spaced by a distance, δ, to span the range of x, and the centre of the Gaussian excitation pattern corresponds to the value coded [12].

The SE algorithm is derived by creating a discrete map which represents the mean value of a continuous probability distribution, $\varphi(\alpha)$, within each class interval. This then provides a simple mechanism for retrieving the original coded value as a sum of the activity of the node excitations, each weighted by the values at the centres of the class intervals α_i. For a particular value of x, the excitation of each node is defined by

$$\psi_i(x) = \frac{\int_{\alpha_i-\delta/2}^{\alpha_i+\delta/2} \alpha\, \varphi(\alpha - x)\,d\alpha}{\alpha_i} \tag{6}$$

which satisfies the requirement that

$$\sum_{i=1}^{N} \alpha_i\, \psi_i(x) = \int \alpha\, \varphi(\alpha - x)\,d\alpha = \overline{\alpha} = x \tag{7}$$

It is assumed that the distribution $\varphi(\alpha)$ has unit area.

The activation of a particular node can be evaluated from Eq. (6) by integration by parts:

$$\alpha_i \psi_i(x) = [\alpha\, \Phi(\alpha - x)]_{\alpha_i - \delta/2}^{\alpha_i + \delta/2} - \int_{\alpha_i - \delta/2}^{\alpha_i + \delta/2} \Phi(\alpha - x)d\alpha \tag{8}$$

where $\Phi(\alpha)$ is a parent cumulative distribution with $\varphi(\alpha) = \Phi'(\alpha)$. In the investigations reported in this paper, the integral term in Eq. (8) was approximated using the first two terms in the trapezium rule resulting in

$$\psi_i(x) \approx \Phi(\alpha_i + \delta/2 - x) - \Phi(\alpha_i - \delta/2 - x) \tag{9}$$

which was found to provide sufficient accuracy in these studies. The relationship between this coding technique and conventional fuzzification techniques is illustrated by considering a first approximation of the integral term in Eq. (8) resulting from a Taylor series expansion of the cumulative function about the interval centre, α_i, and keeping the linear term in the expansion. This leads to

$$\psi_i(x) \approx \varphi(\alpha_i - x) \tag{10}$$

which is analogous to the use of membership functions in fuzzy logic [35].

The practical advantage of spread encoding is that it leads to more accurate models using static feed-forward neural networks than representing normalised physical variables using single nodes. The main reason for this is that signal noise is reduced in the spread encoding representations by suitable matching of the coding function with the interval width spanned by each node, exploiting hence the network's fault tolerance.

The spread encoding algorithm was implemented by initially scaling the data to a normalised range where the original data range $r \in [\, r_{min}, r_{max}]$ was represented by $x \in [0, N\text{-}2N_0]$ with N the total number of nodes, N_0 the number of nodes on either side of the variable range and δ=1. The data coding is performed using the following relation

$$\psi_i(x) = \Phi(\alpha_i + 1/2 - x) - \Phi(\alpha_i - 1/2 - x), \quad i = 1,\ldots,N \tag{11}$$

where

$$\alpha_i = i - N_0 - c \tag{12}$$

and the cumulative Gaussian distribution function was approximated by the sigmoidal function centred at *x*:

$$\Phi(\alpha - x) = \frac{1}{1+e^{-\beta(\alpha - x)}} \tag{13}$$

In Eq. (12), *c* is an offset term which shifts the position of the range limits on the nodes. The width of the node excitations is inversely controlled by the parameter β in Eq. (13). Errors arise in decoding using a straight-forward applications of Eq. (7) because the node excitations, $\psi_i(x)$, are calculated by an approximation, Eq. (9). The accuracy of decoding is improved by dividing the weighted sum by the sum of the node excitations. Thus, the network output is decoded back to the normalised range using

$$x = \frac{\sum_{i=1}^{N} \alpha_i \psi_i(x)}{\sum_{i=1}^{N} \psi_i(x)} \tag{14}$$

which is analogous to the conventional centre of gravity defuzzification technique. In this work, the parameters used in the spread encoding algorithm were N=6, N_0=2, c =0.5 and β=2.3 which were found to provide sufficiently accurate coding and decoding in the application reported in this paper.

Formal techniques for determining an optimum number of nodes in the hidden layers are still an area of current research and presently, this task is often achieved by experimentation. The resulting MLP network topology with the spread encoding applied to the network data, was 48 input nodes, 34 and 16 nodes for the two hidden layers and 6 output nodes. The results for the 14.00h and 2.00h which correspond to hours with maximum and minimum load consumption respectively, are illustrated in Tables 1 and 2.

3.3 Window Random Activation Weight Neural Network (WRAWN)

General function approximation can be obtained by feed-forward neural networks consisting of just one hidden layer of non linear neurons [4]. The innovation here is that training of the weights between the input and hidden layer is not required. By taking these activation weights as random numbers, the problem, concerning the parameters, could be considered as linear, thus can easily be solved using a sliding window least-squares estimator [28].

Let consider a feed-forward neural network consisting of a hidden layer with a non linear sigmoidal activation function, and one output layer with a linear activation function. The related equations are written in matrix form as:

$$\begin{aligned} Z &= XW^h \\ U &= f(Z) = \alpha \tanh(bZ) \\ Y_n &= UW^o \end{aligned} \tag{15}$$

where **X** is the input vector, $\mathbf{W^h}$ is the weight matrix between the input and hidden layer, **U** is the output matrix of hidden layer, $\mathbf{W^o}$ is the weight matrix between the hidden and output layer, and finally $\mathbf{Y_n}$ is the output vector of neural network. If **Y** assumed to be the target vector, then it could be easily generated by the matrix equation

$$\mathbf{Y} = \mathbf{U}\mathbf{W}^o + \mathbf{e} \tag{16}$$

where e is the error term. Suppose that the weights $\mathbf{W^h}$ are fixed. Then the training of the network is equivalent to find the weights $\mathbf{W^o}$ that minimise a specific cost function, which in our case is the difference between the network's output $\mathbf{Y_n}$ and the target output **Y**. Basically, this is a classic least squares problem and provided that the matrix **U** is of full rank, the least-squares solution is given by

$$\mathbf{W}^o = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{Y} \tag{17}$$

Let define

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^T(1) \\ \vdots \\ \mathbf{u}^T(t) \end{bmatrix} \tag{18}$$

where $\mathbf{u}^T(1)=[u_1(1)\cdots u_m(t)],\ldots,\mathbf{u}^T(1)=[u_1(t)\cdots u_m(t)]$ and $\mathbf{y}=[y(1)\ldots y(t)]^T$.

For a moving window of length n_w, define

$$\mathbf{U}_t = \begin{bmatrix} \mathbf{u}^T(t-n_w+1) \\ \vdots \\ \mathbf{u}^T(t) \end{bmatrix} = \begin{bmatrix} \mathbf{u}^T(t-n_w+1) \\ \text{-------} \\ \mathbf{U}(t,t-n_w+2) \end{bmatrix}$$

$$\mathbf{U}_{t+1} = \begin{bmatrix} \mathbf{u}^T(t-n_w+2) \\ \text{--------} \\ \vdots \\ \text{--------} \\ \mathbf{u}^T(t+1) \end{bmatrix} = \begin{bmatrix} \mathbf{U}(t,t-n_w+2) \\ \vdots \\ \text{-------} \\ \mathbf{u}^T(t+1) \end{bmatrix} \tag{19}$$

The unknown parameter vector is calculated as a least squares estimator

$$\mathbf{y}_{t+1} = \mathbf{W}^o(t+1)\mathbf{U}_{t+1} \Rightarrow \mathbf{W}^o(t+1)=(\mathbf{U}_{t+1}^T\mathbf{U}_{t+1})^{-1}\mathbf{U}_{t+1}^T\mathbf{y}_{t+1} \tag{20}$$

Therefore, matrices $(\mathbf{U}_{t+1}^T\mathbf{U}_{t+1})^{-1}$ and $\mathbf{U}_{t+1}^T\mathbf{y}_{t+1}$ should be computed . Substituting the matrices $\mathbf{U}_t^T, \mathbf{U}_{t+1}^T, \mathbf{y}_t, \mathbf{y}_{t+1}$ by their above forms, Eq. (19), it yields

$$\mathbf{U}_t{}^T\mathbf{U}_t = \mathbf{u}(t-n_w+1)\,\mathbf{u}^T(t-n_w+1)+\mathbf{U}^T(t,t-n_w+2)\mathbf{U}(t,t-n_w+2) \tag{21}$$

and after some calculations

$$\mathbf{U}_{t+1}{}^T\mathbf{U}_{t+1} = \mathbf{U}_t{}^T\mathbf{U}_t + \Gamma(t+1) \tag{22}$$

where $\Gamma(t+1)=\mathbf{u}(t+1)\mathbf{u}^T(t+1)-\mathbf{u}(t-n_w+1)\mathbf{u}^T(t-n_w+1)$. (23)

Similarly

$$\mathbf{U}_t^T\mathbf{y}_t = \mathbf{u}(t-n_w+1)\,\mathbf{y}(t-n_w+1)+\mathbf{U}^T(t,t-n_w+2)\,\mathbf{y}(t,t-n_w+2) \tag{24}$$

$$\mathbf{U}_{t+1}^T\mathbf{y}_{t+1} = \mathbf{U}_t^T\mathbf{y}_t + \delta(t+1) \tag{25}$$

Hence the parameter estimation update equation is

$$\hat{\mathbf{W}}^o(t+1)= \hat{\mathbf{W}}^o(t)-\mathbf{P}(t+1)[\Gamma(t+1)\hat{\mathbf{W}}^o(t)-\delta(t+1)] \tag{26}$$

with

$$\mathbf{P}(t+1)=(\mathbf{U}_{t+1}{}^T\mathbf{U}_{t+1})^{-1}=\mathbf{P}^{-1}(t)+\Gamma(t+1) \tag{27}$$

as the covariance matrix of the estimate $\mathbf{W}^o(t+1)$

and

$$\delta(t+1) = \mathbf{u}(t+1)\mathbf{y}(t+1) - \mathbf{u}(t-n_w+1)\,\mathbf{y}(t-n_w+1) \tag{28}$$

$$\mathbf{y}(t) = \begin{bmatrix} y(t-n_w+1) \\ \vdots \\ y(t) \end{bmatrix} = \begin{bmatrix} y(t-n_w+1) \\ ------- \\ y(t,t-n_w+2) \end{bmatrix}$$

$$\mathbf{y}(t+1) = \begin{bmatrix} y(t-n_w+2) \\ \vdots \\ y(t+1) \end{bmatrix} = \begin{bmatrix} y(t,t-n_w+2) \\ ------- \\ y(t+1) \end{bmatrix} \tag{29}$$

The major advantage of this variation is that $\hat{\mathbf{W}}^{o}(t)$ is estimated using information from the last n_w samples. As a result there is a reduction in speed which is proportional to the length of the window, since the dimensions of P, Γ, and δ are independent of the window size.

The main objective in this method is the right choice of activation weights $\mathbf{W}^h$. In order matrix $\mathbf{U}^T\mathbf{U}$ not be close to singularity, sigmoid function should always operates in the linear range. This task is achieved by normalizing input vector in the range $[-a,a]$, where a is a small number $0 < a < 1$, and scaling the randomly chosen weights $\mathbf{W}^h$ in an efficient manner. Hence, the following equation is satisfied:

$$\max_k \operatorname{var}_j \left\{ \sum_{l=1}^{N_i+1} \mathbf{x}_l(k) W_{lj}^h \right\} \le a \tag{30}$$

The weight matrix $\mathbf{W}^h$ is randomly generated by a normal distribution with zero mean and standard deviation of 1. Then the normalised weight matrix $\mathbf{W}^h$ is given by the following equation

$$\mathbf{W}^h = \sqrt{\frac{a}{\max_k \sum_{l=1}^{N_i+1} \mathbf{x}_l^2(k)}}\; N(0,1) \tag{31}$$

where *N*(0,1) denotes a generator of random numbers with zero mean normal distribution and standard deviation of 1. Best results were obtained using an 8/20/1 structure. The results for the 14.00h and

2.00h which correspond to hours with maximum and minimum load consumption respectively, are illustrated in Tables 1 and 2.

3.4 Radial Basis Functions

An alternative model to the multilayer networks for the time series identification, is the neural network employing radial basis functions (RBFs). An RBF is a function which has in-built distance criterion with respect to a centre. A typical RBF neural network consists of three layers (input, hidden, output). The activation of a hidden neuron is determined in two steps: The first is computing the distance (usually by using the Euclidean norm) between the input vector and a center c_i which represents the ith hidden neuron. Second, a function h which is usually bell-shaped is applied, using the obtained distance to get the final activation of the hidden neuron. In our case the well known Gaussian function G(x)

$$G(x)=\exp(-\frac{x^2}{\sigma^2}) \tag{32}$$

was used. The parameter σ is called unit width and is determined using the heuristic rule "*global first nearest-neighbor*" [22]. It uses the uniform average width for all units using the Euclidean distance in the input space between each unit m and its nearest neighbor n. All the widths in the network are fixed to the same value σ and this results in a simpler training strategy.

The activation of a neuron in the output layer is determined by a linear combination of the fixed nonlinear basis functions, *i.e.*

$$F^*(x)=\sum_{i=1}^{M} w_i \phi_i(x) \tag{33}$$

where $\phi_i(x)=G(\|x-c_i\|)$ and w_i are the adjustable weights that link the output nodes with the appropriate hidden neurons. These weights in the output layer can then be learnt using the least-squares method.

The present study adopts a systematic approach to the problem of centre selection. Because a fixed centre corresponds to a given regressor in a linear regression model, the selection of RBF centres can be regarded as a problem of subset selection. The orthogonal least squares (OLS) method can be employed as a forward selection procedure which constructs RBF networks in a rational way. The algorithm chooses appropriate RBF centres one by one from training data points until a satisfactory network is obtained. Each selected centre minimises the increment to the explained variance of the desired output, and so ill-conditioning problems occurring frequently in random selection of centres can automatically be avoided. In contrast to most learning algorithms, which can only work if a fixed network structure has first been specified, the OLS algorithm is a structural identification technique, where the centres and estimates of the corresponding weights can be simultaneously determined in a very efficient manner during learning. OLS learning procedure generally produces an RBF network smaller that a randomly selected RBF network [2]. Due to its linear computational procedure at the output layer, the RBF is faster in training time compared to its backpropagation counterpart.

A major drawback of this method is associated with the input space dimensionality. For large numbers of inputs units, the number of radial basis functions required, can become excessive. If too many centres are used, the large number of parameters available in the regression procedure will cause the network to be over sensitive to the details of the particular training set and result in poor generalisation performance (overfit). To avoid this problem, regularisation method has been proposed by Orr [23]. Both the forward selection and zero-order regularisation techniques were proposed to construct parsimonious RBF networks with improved generalisation properties. Using the regularized forward selection algorithm (RFS) the error criterion for minimisation is given by the equation

$$J = e^T e + \lambda w^T w \qquad (34)$$

instead of the standard $J = e^T e$, where λ is the regularisation parameter. However, the proposed algorithm does not utilise an orthogonalisation scheme; therefore computationally is more expensive from the standard OLS algorithm.

An efficient combination of the zero-order regularization and the OLS algorithm proposed by Chen *et al* [3]. Similarly, the new error criterion for minimization in the ROLS algorithm is

$$J = e^T e + \lambda\, g^T g \tag{35}$$

where g is the orthogonal weight vector which satisfy the triangular system

$$g = AW \tag{36}$$

and A is an upper triangular matrix which computed directly from the OLS regression procedure [2].

In our case, the ROLS algorithm was employed to model the hourly demand load. Best results were obtained using 10 inputs, selected from the general Eq. (1), by setting the parameters p, q equal to 3. The proposed "*global first nearest-neighbor*" method for width definition was found in practice to be inadequate for our problem. A rather heuristic method by taking the half the maximum Euclidean distance was finally adopted for our simulations. Although an elegant approach to the selection of the regularization parameter λ is to adopt Bayesian techniques, in this work this parameter was set by trial and error to small positive values, which satisfy the optimal problem's solution.

The λ parameter was set equal to 0.0002 and 0.0008 respectively for the two cases. As a result, the corresponded centres were found by the OLS procedure to be equal to 27 and 13 respectively. Just for comparison purposes, it should be noted that for the first case of 14h, the original OLS algorithm, without the use of λ parameter, gave us a network with a similar accuracy but with the computational cost of 69 centres. The results for the 14.00h and 2.00h which correspond to hours with maximum and minimum load consumption respectively, are illustrated in Tables 1 and 2.

3.5 Autoregressive recurrent neural network

In the previous sections, the so called *windowed input network* has been applied for modelling the electric load. Another approach has been that of explicitly including the time depedency into the network structure. The commonly used back-propagation algorithm contains no *memory*, hindering the learning of temporal patterns. Here, the alternative is to use a dynamic network that is given some kind of memory to encode past history.

In the standard multilayer perceptron, the input to a neuron are multiplied by feedforward weights and summed, along with a node bias term. The sum is then passed through a smooth sigmoidal transfer function, producing the neuron's output value. This neural model has no memory, because th output value is not expilicitly dependent upon previous outputs.

In this section a novel network architecture for the load forecasting problem, called the autoregressive recurrent network (ARNN) is proposed [12]. It was designed to contain internal memory of the previous states, while training rapidly using a generalised BP algorithm. The idea is that we still use a recurrent neural network model but the recurrent neurons are decoupled so that each neuron only feeds back to itself. With this modification the ARNN model is considered to converge easier and to need less training cycles than a fully recurrent network.

The ARNN is a hybrid feedforward / feedback neural network, with the feedback represented by recurrent connections appropriate for approximating the dynamic system. The structure of the ARNN is shown in Fig. 3. There are two hidden layers, with sigmoidal transfer functions, and a single linear output node. The ARNN topology allows recurrence only in the first hidden layer. For this layer, the memoryless backpropagation model has been extended to include an autoregressive memory, a form of self-feedback where the output depends also on the weighted sum of previous outputs.

A modified backpropagation algorithm was developed to train the ARNN which includes dynamic recursive equations in time. The mathematical definition of the ARNN is shown below:

$$y(t) = O(t) = \sum_{l} W_l^O Q_l(t), \quad Q_l = f(S_l), \quad S_l = \sum_{j} W_{jl}^H Z_j(t) \qquad (37)$$

and

$$Z_j(t) = f(H_j(t)), \quad H_j(t) = \sum_{k=1}^{k=n} W_{jk}^{D} Z_j(t-k) + \sum_i W_{ij}^{I} I_i \tag{38}$$

where $I_i(t)$ is the i^{th} input to ARNN, $H_j(t)$ is the sum of inputs to the j^{th} recurrent neuron in the first hidden layer, $Z_j(t)$ is the output of the j^{th} recurrent neuron, $S_l(t)$ is the sum of inputs to the l^{th} neuron in the second hidden layer, $Q_l(t)$ is the output of the l^{th} neuron in the second hidden layer and $O(t)$ is the output of the ARNN. Here, $f(\bullet)$ is the sigmoid function and W^I, W^D, W^H and W^O are input, recurrent, hidden and output weights, respectively. The index n indicates the number of internal memories in the hidden nodes at reccurent layer and for this application was set to five. Thus the network can be considered as a model capable to carry out five-steps-ahead accurate predictions. The five memories in each node at the first hidden layer allow the network to encode state information [14].

Eight inputs were selected form the model of Eq.(1) by setting the parameters p and q equal to 2. Hence a structure of 8/20/14/1 nodes has been used for the simulation of the STLF problem. The results for the 14.00h and 2.00h which correspond to hours with maximum and minimum load consumption respectively, are illustrated in Tables 1 and 2.

From the results, which are shown in a next section, it was proved that the approach of producing a dynamic memory is clearly simpler than the other proposed techniques, with the result that the computational burden to be substantially reduced [13]. This fact, in conjunction with the simpler network complexity, proves that the current approach is better suited to this kind of sequence processing.

3.6 Neuro-Fuzzy inference system

The various neural architectures presented in previous sections, illustrated their strength for modelling the individual '*blocks*' in the proposed modular architecture. It is well known that a number of complex systems are difficult to be modelled due to their non-linear behaviour and imprecise measurement information. Therefore, imprecise systems states and a set of imprecise heuristic rules are needed.

Zadeh introduced the linguistic approach to system design based on fuzzy logic. The main goal of a fuzzy inference system is to model human decision making within the conceptual framework of fuzzy logic and approximate reasoning [31]. Such a system consists of four important parts: the fuzzifications interface, knowledge base unit, decision making unit and output defuzzification interface.

Recently, the resurgence of interest in the field of artificial neural networks (ANNs) has injected a new driving force into the 'fuzzy' literature. The backpropagation learning rule, which drew little attention till its applications to ANNs was discovered, is actually an universal learning paradigm for any smooth parameterised models, including fuzzy inference systems. As a result, a fuzzy inference system can now not only take linguistic information (linguistic rules) from human experts, but also adapt itself using numerical data (input/output pairs) to achieve better performance. This gives fuzzy inference systems an edge over neural networks, which cannot take linguistic information directly.

In this section a simple fuzzy logic system implemented by using a multilayer feedforward neural network is presented for modelling the individual hourly load '*blocks*' in the proposed modular architecture. A schematic diagram of the proposed fuzzy neural network (N_Fuzzy) structure is shown in Fig. 4.

The system consists of four layers. Nodes in layer one are *input nodes* which represent input linguistic variables. Nodes in layer two are *membership nodes* which act like membership functions. Each membership node is responsible for mapping an input linguistic variable into a possibility distribution for that variable. The *rule nodes* reside in layer three. Taken, together, all the layer three nodes form a fuzzy rule base. Layer four, the last layer, contains the *output variable nodes*.

The links between the membership nodes and the rule nodes are the premise links and those between the rule nodes and the output nodes are the consequence links. For each rule node, there is at most one premise link from a membership node of a linguistic variable. Hence there are $\prod_i|T(x_i)|$ rule nodes in the proposed FNN structure. Here $|T(x_i)|$ denotes the number of fuzzy partitions of input linguistic variable x_i. Moreover, all consequence links are fully connected to the output nodes and interpreted directly as the strength of the output action. In this way, the consequence of a rule is simply the product of the rule node output, which is the firing strength of the fuzzy rule and the consequence link. Thus, the overall network output is treated as a linear combination of the consequences of all rules instead of the complex composition, a rule of inference and the defuzzification process. This fuzzy neural network is a slight modification of the network reported by Lin [17].

For an n-input one-output system, let x_i be the ith input linguistic variable and a^j the firing strength of rule j, which is obtained from the product of the grades of the membership functions $\mu_{A_i^j}(x_i)$ in the premise part. If w^j represents the j^{th} consequence link weight, then the inferred value y* is obtained from the weighted sum of its inputs. Thus, the inference in the proposed fuzzy neural network is realising as

$$\begin{aligned} &j^{th} \text{ rule: IF } x_1 \text{ is } A_1^j, \ldots\ldots, \text{and } x_n \text{ is } A_n^j, \text{ THEN } y = w^j,\ j=1,2,\ldots,m \\ &y^* = \sum_{j=1}^{m} a^j w^j, \quad a^j = \prod_{i=1}^{n} \mu_{A_i^j}(x_i) \end{aligned} \tag{39}$$

The class of fuzzy inference system under consideration is a simplified type which uses a singleton to represent the output fuzzy set of each fuzzy logical rule Thus w^j is the consequence singleton of the j^{th} rule. We now consider the basic function of each node in each layer.

Layer 1: Layer 1 is an input layer whose nodes represent input variables. The nodes just transmit input values to the next layer directly. Hence, for the jth node of layer 1, the net input and output are represented respectively as:

$$net_j^1 = x_i^1, \quad i=j, \ y_j^1 = net_j^1 \tag{40}$$

Layer 2: This is an input term (partition) layer whose nodes represent the membership functions associated with each linguistic term of the input variable. The Gaussian function, a particular example of radial basis functions, is adopted here as a membership function. Hence, the output of the *j*th term node associated with x_i is

$$\mathrm{net}_j^2 = \mu_{A_{ij}}(m_{ij},\sigma_{ij}) = -\frac{(x_i^2 - m_{ij})^2}{(\sigma_{ij})^2}, \quad y_j^2 = \exp(\mathrm{net}_j^2) \tag{41}$$

where m_{ij} and σ_{ij} are respectively, the mean and the variance of the Gaussian function in the *j*th term of the *i*th input linguistic variable x_i^2.

Layer 3: Layer 3 is a rule node layer, where each node represents a rule of the fuzzy system. Thus the nodes in Layer 3 form a rule base. The links in this layer are used to implement the antecedent matching. The matching operations or the fuzzy AND aggregation operation is chosen as the simple PRODUCT operation instead of the MIN operation. Thus, for the jth rule node

$$\mathrm{net}_j^3 = \prod_i^n x_i^3 \;, \quad y_j^3 = \mathrm{net}_j^3 \tag{42}$$

Layer 4: This is an output layer, whose nodes represent the partitions of the output variables. All consequence links are fully connected to the output nodes and interpreted directly as the strength of the output action. This layer performs centroid defuzzification to obtain the numerical output:

$$\mathrm{net}_j^4 = \sum_i^n w_{ij}^4 x_i^4, \quad y_j^4 = \mathrm{net}_j^4 \tag{43}$$

where the link weight w_{ij}^4 is the output action strength of the jth output associated with the ith rule. Thus, the overall net output is treated as a linear combination of the consequences of all rules instead of the complex composition of a rule of inference and the defuzzification process.

The adjustment of the parameters in the proposed FNN can be divided into two tasks, corresponding to the IF (premise) part and THEN (consequence) part of the fuzzy logical rules. In the premise part, we need to initialise the center and width for Gaussian functions. To determine these initial terms, a self-organisation map (SOM) and fuzzy-c-means are commonly used. Another simple and intuitive method of doing this is to use normal fuzzy sets to fully cover the input space. Since the final performance will depend mainly on supervised learning, normal fuzzy sets are chosen for this work. In the consequence part, the parameters are output singletons. These singletons are initialised with small random values, as in a pure neural network.

For our problem, four inputs were selected from the model of Eq.(1) by setting the parameters p and q equal to 2. Hence, the requested load is forecasted both with the load data of the previous two days and also with the forecasted load data for the same day at previous two time steps. Each input variable was assigned to four fuzzy partitions. Figs. 5-6 illustrate the load prediction results for the 14.00h and 2.00h which correspond to hours with maximum and minimum load consumption respectively.

The main advantages of the proposed adopted method are the ability to learn from experience and a high computation rate. The average percentage relative error approaches its optimal value after 5-epoch training. This is due to the fact that the consequence parameters have converged. This implies that the convergence of consequence parameters play a dominant role in system estimation accuracy. The remaining time is just for fine-tuning the premise parameters. Thus the training required to achieve acceptable accuracy was very fast compared to the other techniques.

4. DISCUSSION OF RESULTS

This section presents the results and the statistics of forecasts obtained from the application of the developed STLF models on the power system of the island of Crete. Case studies for the proposed methods were carried out for a 24-hour load forecasting.

Several structures of neural networks with algorithms ranging from backpropagation learning to OLS methods were tested. Note that the neural models differ from the usual time-series models. Since they include loads for both previous times and previous days, they better represent the hourly variations than other existing methods.

An obvious advantage of the proposed modular architecture is that since the complete system consists of 24 neural blocks, each one with a single output, training is easier and faster compared to traditional neural approaches which treat the output as a 24x1 vector. The results were analysed based on the following indices:

Standard error deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[y_i - y_i]^2}$$

Percent relative error

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}|y_i - y_i| \cdot 100 / y_i$$

Root mean square error

$$\varepsilon_1 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[\{y_i - y_i\} / y_i]^2} \cdot 100$$

The complete results for the STLF problem, for the hours with minimum and maximum load consumption, are illustrated in Tables 1 and 2 respectively.

The widely used for such an application, standard MLP with ABP learning algorithm, was considered in this work as a testbed case. In an alternative representation, the SE structure has been shown to enable a network to maintain a higher degree of accuracy compared with the classic MLP structure. Although this considerable improvement in performance is generally at the expense of a larger in size network, the use of the proposed structure has significant advantages in applications requiring long-range predictions. The performance of a classical MLP will severely deteriorate when it is acting as a recurrent model because any errors of the predicted output will tend to accumulate. This problem is avoided in this specific structure, by splitting the error to several nodes, thus exploiting the network's fault tolerance.

A faster in training method was demonstrated with the use of WRAWN algorithm using a sliding window least-squares estimator. Using a proper set of parameters, the results were found to be similar to those from a classic MLP using the ABP learning rule.

An alternative method to MLPs could be consider a neural model employing radial basis functions. Both aspects of training time and improved accuracy were satisfied with the use of ROLS. An additional advantage of the specific algorithm was the overfitting problem avoidance by using the regularised parameter *ë*. Due to this factor, the RBF network enjoys a very compact structure compared to the other proposed neural architectures.

The use of dynamic neural networks present an innovation to the specific problem. Here, the objective was to use a dynamic network (ARNN) that was given some kind of memory to encode past history, with the additional requirements of short training time.

The improved, compared to the standard MLP structures, results reveal the advantages of using memory neuron structures. The inclusion of five memories and the related recurrence in the first hidden layer, enable the network to carry out five-steps ahead accurate predictions. Although this method is depedent on the number of 'memories' and therefore it can be considered as a partially recurrent network, its use to the proposed modular architecture will allow the expansion of the prediction horizon beyond the limit of 24 hours.

However, the introduction of hybrid learning algorithms imposed a new dimension to STLF. The main advantages of the proposed Neuro-Fuzzy (N-Fuzzy) method are the ability to learn from experience and a high computation rate. The average percentage relative error approaches its optimal value after 5-epoch training. This is due to the fact that the consequence parameters have converged. This implies that the convergence of consequence parameters play a dominant role in system estimation accuracy. The remaining time is just for fine-tuning the premise parameters. Thus the training required to achieve acceptable accuracy was very fast compared to the other techniques.

5. CONCLUSIONS

This paper is based on the comparative analysis of neural network based STLF techniques. These methods were developed for a one-day-ahead prediction of hourly electric load using a modular architecture. Several neural architectures were tested including, multilayer perceptrons, WRAWNs, fuzzy-neural-type networks, radial basis and memory neuron networks. The building block of the existing forecasting systems is an MLP trained with the BP algorithm. However, in such dynamic applications, the real computing power of connectionist models could be exploited only if the networks themselves are dynamic in nature. Two approaches to processing inputs in the time domain have been applied: One to window the inputs and then treat the time domain like another spatial domain, with the result the development of the SE network. The obtained results from the SE model, reveal the advantage of the proposed neural approach compared to the widely used multilayer perceptron employed with the standard BP algorithm.

Alternatively to introduce dynamics into the network by transferring the regular network neuron state to another set of '*duplication*' neurons called memory neurons. The autoregressive neural network is an example of such architecture. Its performance is characterised by a high degree of accuracy as well as fast training time, similar to the performance of the RBF network based on the ROLS algorithm.

The introduction of hybrid learning algorithms imposed a new dimension to STLF. The main advantages of the proposed Neuro-Fuzzy (N-Fuzzy) method are the ability to learn from experience and a high computation rate. The average percentage relative error approached its optimal value after

5-epoch training. Thus the training required to achieve acceptable accuracy was very fast compared to the other implemented techniques.

It should be noted that the proposed structures differ from the usual time-series prediction models since they include only loads for both previous times and previous days for hourly load forecasting. In a future work, the present approach will be enhanced by using advanced neuro-fuzzy models and additional load and weather information such as illuminations level, temperature, humidity, wind direction and industrial load.

ACKNOWLEDGEMENT

The authors would like to thank Professor G. Stavrakakis, for providing access to load data for the Power System of the Greek Island of Crete.

REFERENCES

1. Bakirtzis, A., Petridis, V., Klartzis, S., A neural network short term load forecasting model for the greek power system. *IEEE Trans. on Power Systems*, 1996, **11**(2), 858-863.
2. Chen, S., Cowan, C.F., Grant, P., Orthogonal least-squares learning algorithm for radial basis function networks. *IEEE Trans. on Neural Networks*, 1991, **2**, 302-309.
3. Chen, S., Chng, E., Regularised orthogonal least squares algorithm for constructing radial basis function networks. *Int. J. Control,* 1996, **64**(5), 829-837.
4. Cybenko, G., Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 1989, **2**, 303-314.
5. Gross, G, Galiana, F., Short term load forecasting. *Proc. IEEE*, 1987, **75**(12), 1558-1573.
6. Hertz, J., Krogh, A., Palmer, R.G., *Introduction to Theory of Neural Computation*, Addison-Wesley, 1991.

7. IEEE Committee Report, Load forecasting bibliography Phase I. *IEEE Trans. on Power App. and Sys.,* 1980, **99**, 53-58.

8. Irisarri, G, Widergren, S., On-line load forecasting for energy control center applications. *IEEE Trans. on Power App. and Sys.,* 1982, **101**(1), 71-78.

9. Jabbour, K., Riveros, J.F.V., Landsbergen, D., ALFA: automated load forecasting assistant. *IEEE Trans. PWRS*, 1988, **3**, 908-914.

10.Kariniotakis, G.N., Kosmatopoulos, E., Stavrakakis G., Load forecasting using dynamic high-order neural networks. *Proc. of IEEE-NTUA Joint Int. Power Conf.*, 1993, 801-805.

11.Kodogiannis, V.S., Lisboa, P.J.G. and Lucas, J., A Neural Predictive Controller for Underwater Robotic Applications. *Proc. 2nd IEEE/BCS Workshop on Neural Network Applications and Tools*, Liverpool, September 1993, IEEE Computer Society Press, pp. 126-133.

12.Kodogiannis, V.S., Neural network techniques for modelling and learning control of an underwater robotic vehicle. PhD thesis, Liverpool University, May 1994.

13. Kodogiannis, V.S., Lisboa, P.J.G. and Lucas, J., Neural network modelling and control for underwater vehicles. *Artificial Intelligence in Engineering*, 1996, **1**, 203-212.

14. Kodogiannis, V.S., Lisboa, P.J.G., Lucas, J., Long Range Predictive Controller for Underwater Robotic Vehicles using Recurrent Neural Networks. *2nd IEEE Mediterranean Symposium on New Directions in Control & Automation*, Chania, Crete, Greece, June 1994, pp. 217-224.

15. Kosmatopoulos, E., Polycarpou, M., Christodoulou, M, Ioannou, P., High - order neural network structures for identification of dynamical systems. *IEEE Trans. on Neural Networks*, 1995, **6**(2), 422-432.

16. Lee, K., Cha, Y., Ku, C., A study on neural networks for short-term load forecasting. *Proc. of ANNPS '91*, Seattle, July 1991, 26-30.

17. Lin, C.T., lee, C.S., Neural-network-based fuzzy logic control and decision system. *IEEE Trans. Computers*, 1991, **40**(12), 1320-1336.

18. Lu, C.N., Wu, H.T., Vemuri, S., neural network based short term load forecasting. *IEEETrans. on Power Systems*, 1993, **8**(1), 337-342.

19. Marr, D., Hildreth, E., Theory of edge detection. *Proc. Roy. Soc. B*, 1980, 207.

20.Mehra, R., On the Identification of variance and Adaptive Kalman filtering. *Proc. of JACC*, 1969, 494-505.

21.Moghram, I., Rahman, S., Analysis and evaluation of five short-term load forecasting techniques. *IEEE Trans. on Power Systems*, 1989, **4**(4), 1484-1491.

22. Moody, J., Darken, C., Fast learning in networks of locally tuned processing units. *Neural Computation*, 1989, **1**, 281-294.

23. Orr, M., Regularised centre recruitment in radial basis function networks. Research report No 59, Centre for Cognitive Science, University of Edinburgh, U.K, 1993.

24. Papalexopoulos, A., How, S., Peng, T.M., An implementation of a neural network based load forecasting model for the EMS. Paper 94 WM PWRS presented at the IEEE/PES 1994 Winter Meeting, 207-209.

25.Park, D.C, El-Sharkawi, M.A., Marks II, R.J., Atlas, L. & Damborg, M., Electric load forecasting using an artificial neural network. *IEEE Trans. on Power Systems*, 1991, **6**(2), 442-449.

26. Park, J.H., Park, Y.M., Lee, K., Composite modeling for adaptive short term load forecasting. *IEEE Trans. on Power Systems*, 1991, **6**(2), 450-456.

27. Peng, T.M., Hubele, N.F., Karady, G.G., Advanement in the application of neural networks for short term load forecasting. *IEEE Trans. on Power Systems*, 1992, **7**(1), 250-258.

28. Pouliezos, A., Recursive sliding window estimators. *Information and Decision Technologies*, 1993, **19**, 19-30.

29. Press, W., Teukolsky, S., *Numerical Recipes in C: The art of scientific computing*. Cambridge University Press, 1992.

30. Rahman, S., Bhatnager, R., An expert system based algorithm for short term load forecast. *IEEE Trans. PWRS*, 1988, **3**, 392-399.

31. Ross, T., *Fuzzy Logic with Engineering Applications*, McGraw Hill, 1995.

32. Rumelhart, D., McClelland, T. L., (eds), *Parallel Distributed processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, MIT Press, 1986.

33. Vemuri, S., Huang, W., & Nelson, D., On-line Algorithm for forecasting hourly loads of an electric utility. *IEEE Trans. on Power App. and Sys.*, 1981, **100**, 3775-3784.

34. Vogl, T.P., Mangis, J.K, Rigler, A.K., Accelerating the convergebce of the backpropagation method. *Biological Cybernetics*, 1988, 257-263.

35. Zadeh, L.A., Fuzzy sets. *Information and Control*, 1965, **8**, 338-353.

	Models	AR	WRAWN	ABP	SE	RBF	N-Fuzzy	ARNN
Statistical Properties								
Relative Error %		3.86297	2.75417	2.7346	1.5174	1.35119	1.202	1.2233
Standard Deviation Error		5.90894	4.19438	3.91741	2.1345	1.96127	1.962	1.83768
RMSE %		5.80199	4.0633	3.80634	2.0314	1.86722	1.815	1.71771

Table 1

	Models	AR	WRAWN	ABP	SE	RBF	N-Fuzzy	ARNN
Statistical Properties								
Relative Error %		11.1837	10.59024	11.9674	3.4001	2.79244	2.452	2.73292
Standard Deviation Error		25.6068	21.74416	24.3568	7.716	5.77855	5.503	5.92847
RMSE %		17.2049	15.19347	17.6778	4.6194	3.60609	3.318	3.68308

Table 2

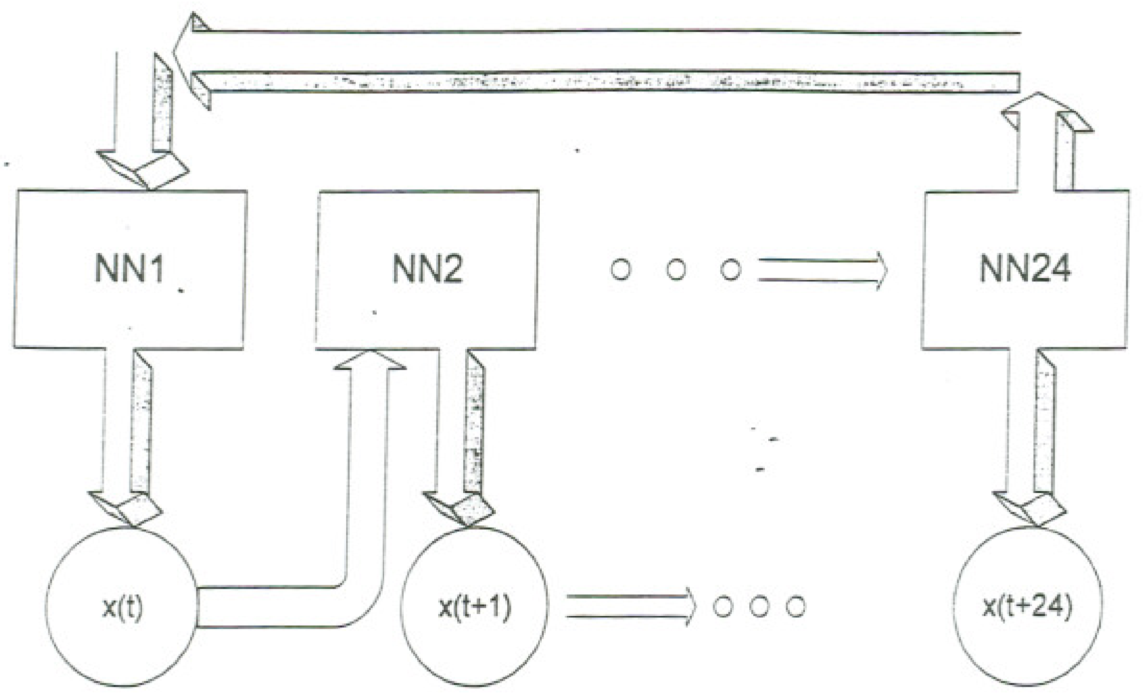

Figure 1: Proposed modular architecture for the STLF problem

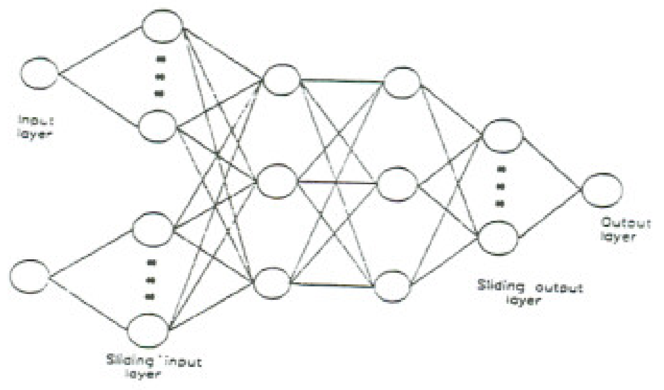

Figure 2: Spread encoding neural architecture

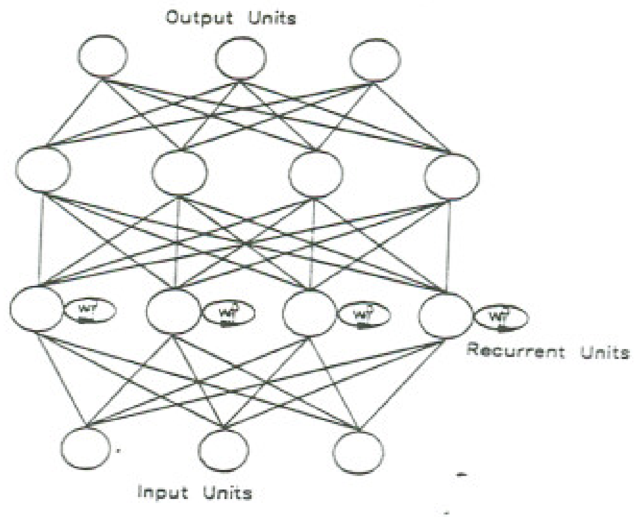

Figure 3: ARNN architecture

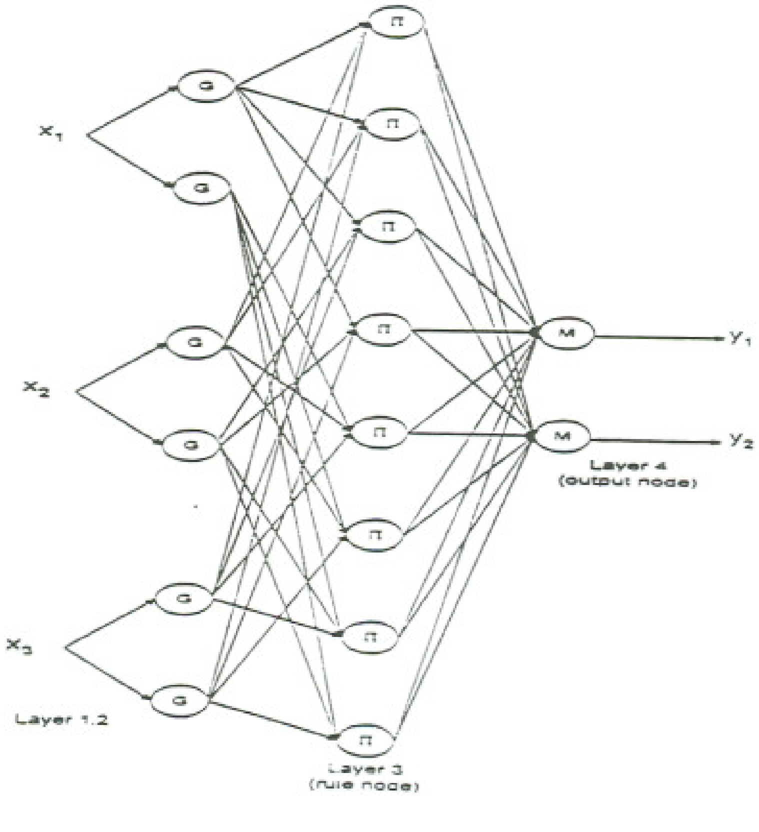

Figure 4: Neuro-fuzzy architecture

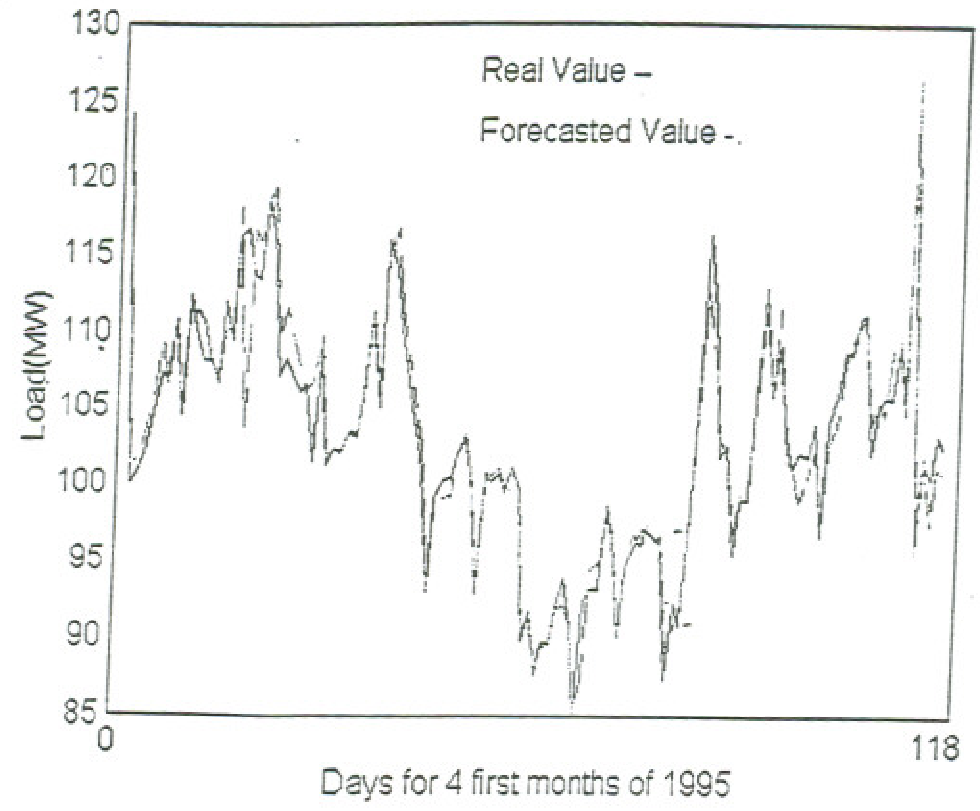

Figure 5: Load forecasting of 2:00h using the N-Fuzzy model

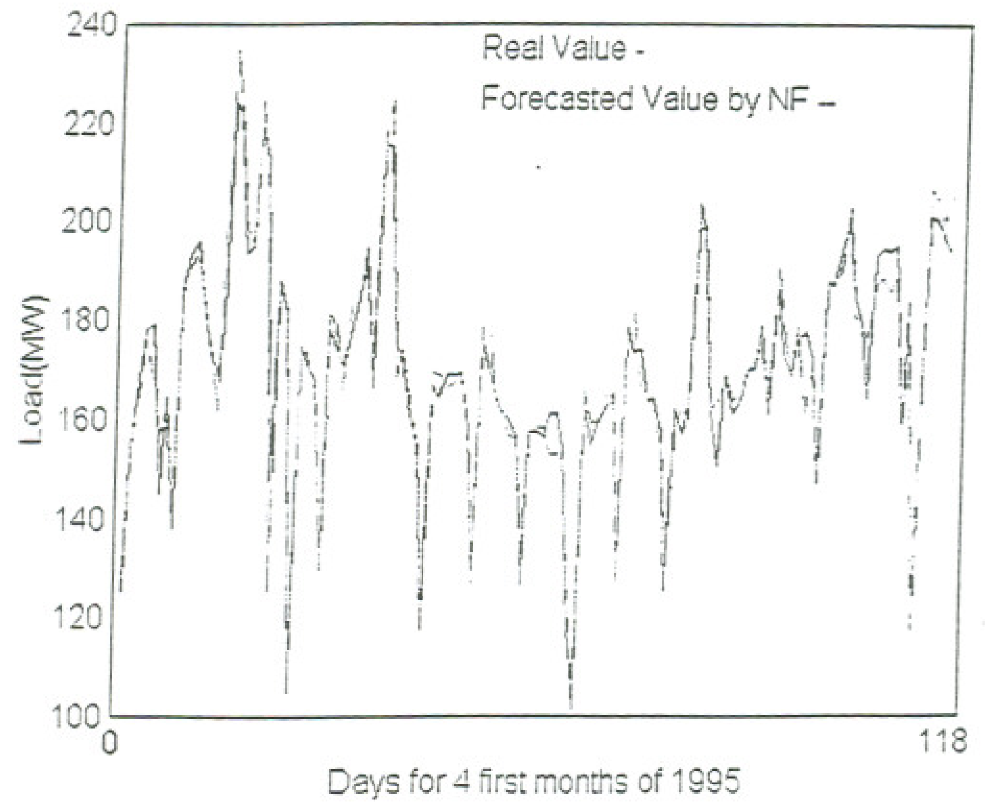

Figure 6: Load forecasting of 14:00h using the N-Fuzzy model