Chapter - III
A functional-link neural (FLN) network based short-term load forecasting model for real-time implementation is presented in this chapter. The load and weather parameters are modeled as a nonlinear time series whose parameters are obtained using the function approximation capabilities of an auto-enhanced functional link network. Numerous and significant advantages accrue from using a *flat net*, which included - *rapid quadratic optimization in the learning of weights, simplification of hardware and computational procedures*. The functional link network based model is capable of forecasting load for a lead time varying from one hour to seven days and does not vary from season to season or hour to hour. The nonlinear weight adjustment algorithm adapts the weights every 24-hour or 168-hour depending on the forecasting requirement. The new algorithm is tested with a load and weather data for a period of two years and produces a very robust and accurate forecast with a MAPE mostly less than 2% for a 24-hour ahead forecast and 2.5% for a 168-hour ahead forecast. The functional link net based load forecasting system accounts for seasonal and daily load characteristics, as well as abnormal conditions and holidays. The results indicate that the functional link net based load forecasting system produces robust and more accurate load forecasts in comparison to simple adaptive neural network or statistical based approaches.
3.1 Introduction

Many algorithms have been proposed in the last few decades for performing accurate load forecasts [3.1]-[3.2]. The most commonly used techniques include statistically based techniques, expert system approaches and Artificial Neural Network (ANN) algorithms. The time series and regression techniques are the two major classes of conventional statistical algorithms, and have been applied successfully in this field for many years. [3.3]-[3.7]. The expert systems based algorithm [3.8] for short-term load forecasting use a symbolic computational approach for automating intelligence. This approach takes advantage of the expert knowledge of the operator which, however is neither easy to elicit nor articulate. A major advantage of using ANN [3.9]-[3.12] over expert systems is its non-dependency on an Expert. Furthermore, ANN can performs non-linear regression among load and weather patterns, and also be used to model the time series method or as a combination of both. Neural networks are 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. Furthermore ANN can perform non-linear regression among load and weather patterns, and also can be used to model the time series method.

Generally, time series approaches assume the load can be decomposed into two components. One is weather dependent and the other weather independent. Each component is modeled separately and the sum of these two gives the total load forecast. The behavior of weather independent load is mostly represented by a Fourier series or trend profiles in terms of time functions. The weather sensitive portion of the load is arbitrarily extracted and modeled by a predetermined functional relationship with weather variables.

An adaptive neural network approach has been recently proposed by Peng et.al.[3.13] which incorporates the familiar Box and Jenkins time series model. Instead of off-line simulation, Adalines are used to update the model parameters and simulation results indicate that one-week ahead hourly forecasts can be generated with a mean absolute percentage error(MAPE) less than 3.4 percent. This forecasting model does not consider the weather dependency of the load, particularly the temperature and uses only past load.

The objective of the present approach is to study the Functional-Link Net architecture to identify a time-series load model incorporating the nonlinearity due to temperature
variations. The foundation of functional-link network is given in the Appendix at the end of this chapter. The functional-link-network has an input vector comprising the Fourier series functions and nonlinear components comprising the temperature functions and their enhancements. The model parameters are identified during training and once the convergence is achieved, the forecasting model is ready for prediction. This new approach is totally adaptive and generalized and does not depend upon the season and day type. The forecasting accuracy of functional link-net compared to the Adaline for one week ahead forecasting is less than 2.5 percent instead of 3.4 percent for the later. The approach is highly flexible and Sundays and holidays can be easily 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, New Year are treated as Sundays. Other holidays, including Good Friday, Memorial Day, Independence Day, Labor Day, the days preceding Christmas and New Year and the day after are treated as Saturdays.

The approach presented in this paper is amenable for real-time implementation as hourly or daily adaptation of model parameters can be done.

3.2 Overview of the Proposed Approach

The short-term load forecasting model for a power utility is assumed to generally to consist of four components:

(i). \textit{Weather independent component}, \( L_i(t) \) (which represents the system intrinsic load component).

(ii). \textit{Weather dependent load component}, \( L_w(t) \)

(iii). \textit{Noise residual component}, \( L_r(t) \)

(iv). \textit{Deterministically known load component}, \( L_d(t) \) (which is the a priori information regarding any abnormal load changes, e.g. proposed tripping or shutdown).

Thus total load is the sum of the above four components and is given by

\[ L(t) = L_i(t) + L_w(t) + L_r(t) + L_d(t) \] (3.1)

104
Normally the forecasted load will be

\[ L(t) = L_i(t) + L_o(t) + L_r(t) \]

but if it is known that a particular utility will remain shut or require an additional load with a known consumption \( L_d(t) \), then the total should read,

\[ L(t) = L_i(t) + L_o(t) + L_r(t) + L_d(t) \]

Obviously, \( L_d(t) \) will be positive for additional load demand or be negative for a shut down. Out of the above four components, the deterministic component does not require any prediction. The weather independent load \( L_i(t) \) is modeled as a Fourier series and comprises of non-stationary Fourier coefficients \( a_0(t), a_1(t), \ldots, b_1(t), b_2(t), \ldots, b_m(t) \), and a time varying part that is a function of \( m \) frequencies with a daily periodicity. This component is represented by

\[ L_i(t) = a_0(t) + \sum_{i=1}^{m} a_i(t) \cos(\omega_i t) + b_i(t) \sin(\omega_i t) \]

\( \omega_i = \frac{2\pi}{T} \) radians per hour. The fundamental time period \( T \) is defined as 24 hours for 24-hour ahead forecasting and 168-hours for one week ahead forecasting. The most suitable value of \( m \) is either chosen by spectral analysis or by trial-and-error. For the present implementation \( m \) is chosen as 12. The residual noise component \( L_r(t) \) is non-deterministic in nature. However, the functional expansion of the past error component has shown to support the residual component, which is inherent to the model.

In general weather conditions affecting electric load behavior include temperature, humidity, wind speed, cloud cover and other abnormal situations such as thunderstorms, etc. Investigations have shown that, in most situations temperature is usually the leading factor affecting the load behavior among all weather variables. However, relative humidity has some importance when the temperature condition is abnormal in summer and wind speed is an important factor in winter with low temperature conditions. Thus the weather dependent load component is modeled as

\[ L_s(t) = \beta_1 T_o^1(t) + \beta_2 T_o^2(t) + \beta_3 T_o^3(t) \]
where, $\beta_1, \beta_2,$ and $\beta_3$ are the nonlinear temperature model coefficients. It may be worthwhile to use an equivalent temperature $T_{eq}(t)$ to account for humidity. Here $T_a(t)$ represents the temperature ($^\circ C$) for the hour $t$ of the day of forecast.

3.3 Auto-Enhanced Functional Link Network

The load forecasting model described in equation (3.1) is realized by using a special kind of neural network called "Auto enhanced functional link net"[3.16]. Functional Link Net (FLN) represents the network architecture that allows unsupervised learning, supervised learning and associative retrieval to be carried out with the same net configuration and with the same data structure. The basic idea behind a Functional Link Net is the use of links for affecting the nonlinear transformations of the input pattern before it is fed to the input layer of the actual network. The concept of the functional link network is illustrated in Fig. 3.1. The idea is that activation of node $K$ offers the possibility that different additional processes $f_1(y(k)), \ldots, f_n(y(k))$, may also be activated through a functional expansion. In a normal feed forward neural network, vector-to-vector mapping can be represented by a series of linear matrix multiplications, each of which is followed by a nonlinear activation function transformation. The form for the activation function is fixed from several standard varieties and only threshold parameter values are varied in the learning process. In contrast to this, the learning of the functional form of mapping, for approximating the unknown nonlinear function $y(k)$ in terms of an expansion with random coefficients is attempted. In this way the generation of an enhanced pattern in place of an actual pattern is performed. The flat architecture of the FLN helps to achieve better learning and in some applications like the load forecasting, this reduces the convergence time drastically. A single layer auto-enhanced functional-link net is described in this section to represent the load forecasting model as shown in Fig. 3.2. The input to the net comprises a linear harmonic series and a nonlinear weather sensitive portion as

$$X = [X_1X_2X_3]^T \quad (3.4)$$

where,
\[
X_1 = \begin{bmatrix}
1 \\
\cos \omega t \\
\sin \omega t \\
\vdots \\
\cos n \omega t \\
\sin n \omega t
\end{bmatrix}
\]
\[
X_2 = \begin{bmatrix}
T_a(t) \\
T_a(t)^2 \\
T_a(t)^3
\end{bmatrix}
\]
\[
X_3 = [G(A_1X_1 + b_1), \ldots, G(A_nX_1 + b_n)]^T
\]  
(3.5)

\(T_a\) = temperature of the forecasted hour and \((A_jX_1 + b_j)\) represents a linear transformation of the input temperature pattern vector with \(G\) representing any squashing function. The estimated output of the net is given by

\[
\hat{y} = W^T X
\]  
(3.6)

where, \(W\) is the weight vector.

In the one-hidden layer neural network, two sets of weights \(W\) and \(A_j\), and thresholds \(b_j\) need to be learned, through back propagation of error. But in the auto-enhancement version of the functional-link net, \(\{A_j\}\) and \(b_j\) are generated randomly. This results in a flat-net architecture for which only weights \(W\) need to be learnt. The hyperbolic tangent function is used as the activation function \(G\) in (3.4). Thus the total input vector \(X\) becomes

\[
X^T = [1 \ \cos \omega t \ldots \cos n \omega t \ \sin \omega t \ \sin n \omega t \ \ T_a(t) \ \ T_a(t)^2 \ \ T_a(t)^3 \ \ tanh(a_{11}T_a(t) + a_{12}T_a(t)^2 + a_{13}T_a(t)^3 + b_1) \\
\vdots \\
\tan(a_{n1}T_a(t) + a_{n2}T_a(t)^2 + a_{n3}T_a(t)^3 + b_n)]
\]  
(3.7)

where, \(n\) = number of auto-enhancements, 
\(a_{11}, a_{12}, \ldots, a_{n3}\) and \(b_1, \ldots, b_n\) are the coefficients which are generated randomly in the beginning.

The weight vector \(W\) in equation (3.6) is adapted using an adjustment law given by Widrow-Hoff delta rule [3.14].
\[ W(k+1) = W(k) + \frac{\alpha e(k) \theta(k)}{\lambda + X^2 \theta(k)} \]  
(3.8)

where, 
\( k \) = iteration count
\( \alpha \) = learning parameter
\( \lambda = 0.00001 \)
\( e(k) = \) error at the iteration count \( k = y(k) - \hat{y}(k) \)
\( y(k) = \) desired output during training of the net
\( \hat{y}(k) = \) estimated output

\[ \theta(k) = \begin{cases} SGN(X_i) & 1 1 1 \tanh(a_{11}T_a(t) + a_{12}T^2_a(t) + a_{13}T^3_a(t) + b_1) \\ & ... \tanh(a_{n1}T_a(t) + a_{n2}T^2_a(t) + a_{n3}T^3_a(t) + b_n) \end{cases} \]

The harmonic series portion \( (X_i) \) of the input vector is passed through a nonlinearity which is a signum \( (SGN(.) ) \) function.

This is given by

\[ SGN(x_i) = \begin{cases} 1, & x_i > 0 \\ 1, & x_i = 0 \\ -1, & x_i < 0 \end{cases} \]  
(3.9)

Using a nonlinear function \( \theta(k) \) in the weight adjustment algorithm, a fast convergence is achieved for \( 0 < \alpha < 2 \). The proof is given in the following section.

### 3.3.1 Proof of Convergence

A Lyapunov stability formalism is applied to the design of the learning rule for the functional-link net estimator. The positive definite function used for the purpose is

\[ V(k) = \|W(k) - W(0)\|^2 \]  
(3.10)

where,

\( W(0) \): Converged weight vector after learning.
\( W(k) \): Weight vector at \( k^{th} \) iterations.

Equation (3.10) is rewritten as
\[ V(k) = (W(k) - W(0))^T (W(k) - W(0)) = \tilde{W}^T(k)\tilde{W}(k). \]

where,

\[ \tilde{W}(k) = \text{weight error vector} = W(k) - W(0). \]

Clearly, \( V(k) \) is the square of Euclidean distance between the converged weight vector \( W(0) \) and weight vector at \( k^{th} \) iterations \( W(k) \).

By using the weight adjustment algorithm (3.8), we obtain

\[ \Delta V(k) = \frac{-2\alpha \tilde{W}^T(k)\theta(k)}{\lambda + X^T(k)\theta(k)} e(k) + \frac{\alpha^2 \theta^T(k)\theta(k)}{\lambda + X^T(k)\theta(k)} e^2(k). \]

The discrete error at the \( k^{th} \) step is

\[ e(k) = y(k) - \hat{y}(k) = (W^T(0) - W^T(k))X(k) = -\tilde{W}^T(k)X(k). \]

Again, it can be shown that

\[ e^2(k) \leq -\tilde{W}^T(k)\theta(k)e(k). \]

Using this inequality, eqn(3.13) is rewritten as

\[ \Delta V(k) = \left(-2\alpha + \frac{\theta^T(k)\theta(k)}{\lambda + X^T(k)\theta(k)}\right)\left(\frac{\alpha e^2(k)}{\lambda + X^T(k)\theta(k)}\right) \]

With \( 0 < \alpha < 2, \lambda > 0, \) and \( \left(2 - \frac{\alpha \theta^T(k)\theta(k)}{\lambda + X^T(k)\theta(k)}\right) > 0 \)

\[ \Delta V(k) \leq 0 \]

As the gradient of \( V(k) \) is negative, the weight vector \( W(k) \) will attain a minimum as \( k \to \infty \).

### 3.3.2 Optimization of Hyperbolic Coefficients

A gradient descent method is used to optimize the coefficients \( a_{lm} \) and \( b_i \) of functional expansion. The optimization is done according to following formulae,
here, \( \mu_a \) and \( \mu_b \) are the learning parameters.

The error function \( E \) is defined as an error of the form

\[
E = \frac{1}{2} (y - \hat{y})^2
\]

(3.19)

Substituting the values of \( \frac{\partial E}{\partial a_m} \) and \( \frac{\partial E}{\partial b_m} \) into (3.18) we get

\[
a_m(k+1) = a_m(k) + \mu_a e(k) W(k) a_m(k) \sec^2 h(a_{11}(k) T_a + a_{12}(k) T_a^2 + a_{13}(k) T_a^3 + b_1)
\]

\[
b_l(k+1) = b_l(k) + \mu_b e(k) W(k) \sec^2 h(a_{11}(k) T_a + a_{12}(k) T_a^2 + a_{13}(k) T_a^3 + b_1)
\]

(3.20)

The forecasting error generated by the autoenhanced flat link net model can be used to improve the accuracy of the final forecast. This error is based on the model’s most recent performance for 24-hour ahead forecast; the forecasting error is available at the end of each day when the actual load for all the 24-hours will be available. Thus the final model will be

\[
X(k) = [X_1(k) \ X_2(k) \ X_3(k) \ e(k-1)]^T
\]

(3.21)

### 3.4 Simulation Results and Evaluation

For identifying the parameters of the load model, four data categories are used depending upon their seasonal characteristics i.e. Winter, Spring, Summer and Fall. Further, data of the weekdays (Monday through Friday) are treated separately from the weekends (Sundays or Saturdays). For this purpose the data on the weekends are separated from the load data-base for the purpose of identification of the parameters of the week days load model during neural network training. In a similar way, the parameters of the weekends load model are identified by using the proper load database. Similar load patterns are used during the training of the network to obtain a faster convergence.
To evaluate the performance of the proposed network architecture, it is used to forecast both one-week ahead hourly load and 24-hours ahead hourly load. In the simulations, the data from an utility in the state of Virginia, USA are used. A mathematical software, MATLAB is used for obtaining load forecasts. The input vector contains twenty five Fourier series functional inputs and three temperature inputs $T_a(t)$, $T_a^2(t)$, $T_a^3(t)$. Here $T_a$ either represents the hourly temperatures or equivalent hourly temperatures including the effect of humidity measurements. In addition there are auto-enhancements which are hyperbolic tangent functions of random linear combinations of temperature inputs and one recurrent error input. Thus the total number of inputs is thirty one for two extensions and forty four for fifteen extensions.

### 3.4.1 Training and Testing

The training of the auto-enhanced functional link network was started with a random set of weights along with a random set of tangent hyperbolic coefficients. The number of enhancements is initially chosen as 2. For sets of given input data and output load patterns, an optimum weight vector is found with the minimum Root Mean Square Error (RMSE). Then the number of hyperbolic tangent extensions is increased and the tuning of $a_m$ and $b_i$ coefficients is attempted to reduce the prediction error to within 1% of the actual load on the forecasted day. The load model presented in this paper showed the best result with 15 extensions during Winter/Spring season and 18 extensions during Summer/Fall. During training nearly 200 iterations are required to reduce the error between the computed load and actual load to a value less than $10^{-14}$. The learning parameter $\alpha$ is chosen between $0<\alpha<1$ and is flexible enough to be varied for faster convergence and lower error.

### 3.4.2 Adaptive Mechanism

While operating in real-time environment, it is imperative that the load forecasting system should be able to adapt to changing conditions. In order to achieve this objective, daily, weekly or monthly adaptation is performed on the load forecast model.

For daily adaptation, the optimized weights are used from the training set to forecast the first day load. After the end of the day of forecast model, parameters are updated till the error becomes insignificantly small. The number of iterations required for this purpose is
extremely small as the error is reduced at the rate of \((1-\alpha)\) every iteration. Once the new weights are established, the forecast for the next day is attempted using this weight vector and the new set of input data for day. As the weekends are excluded from the 1st set of data base, the weight vector obtained after the forecast of Friday load is used to predict the load on Monday. For one week ahead forecasts, the adaptation is done once a week, i.e. at the end of the week when the entire load profile for the whole week will be available.

The mean absolute percentage error (MAPE) is used to compute the performance of the real-time algorithm and is defined as follows:

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

where, \(N\) is the number of patterns in the data set used to evaluate the forecasting capability of the model.

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

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

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 weekdays.

The special holiday data occurring in the past and the weekend data are used to train the functional link net before the prediction for the holiday is attempted. By collecting similar days from the past, we ensure that the characteristic of only that type of holiday is reflected in the data set.

3.4.3 Forecasting Results

The results of the 24-hours ahead load forecast over one week period for all the seasons (Winter, Spring, Summer, Fall) are shown in the Fig. 3.3 and Fig. 3.4. In Fig. 3.3
MAPE of each season are shown and the PEs corresponding to those weeks are shown below them. The actual vs predicted load of those weeks are shown in Fig. 3.4. From the figures it is observed that MAPE is lowest in January (Winter) and is less than 2%; for all other seasons it is around 2.5%. The PE is found to fluctuate heavily in 168-hours ahead forecast; however it is bounded mostly within ±4% and has touched a value of 6% only once over the entire week. This can also be reduced by smoothening the data prior to processing or fine tuning the parameters of the flat-net extensions.

From the results presented in Fig. 3.4, it can be observed that the estimated load follows the actual load during the week under consideration for all the seasons very closely except in July. The average MAPE for all the four seasons in the year of forecast for all day types (Monday through Sunday) reveals excellent accuracy and is under 2%, which is a very significant improvement in comparison to the forecast results with ANN approach.

The MAPE and PEs of 168-hour ahead forecast for Winter and Spring season are shown in Fig. 3.5. The MAPE values for the above two seasons are found to be within 2.8%, which are much improved in comparison to the results obtained in reference [3.12].

The actual and forecasted loads are shown in Fig. 3.6 over a week for all the season showing excellent accuracy in predicting the 168-hour ahead load magnitudes.

Fig. 3.7 shows the number of occurrences of a particular MAPE value over a year for both 24-hours and 168-hours ahead load forecasting.

The average MAPE for all the four seasons of the year is shown in Fig. 3.8. The MAPE for 24-hour ahead is shown in Fig. 3.8 (a) and the MAPE for 168-hour ahead is shown in Fig. 3.8 (b) respectively. Here the MAPE of 168-hour ahead forecast is less than 3.5%, which is very significant in comparison to the results obtained using simple Adalines and ANN based algorithms.

The absolute percentage of error(PE) for 24-hours and 168-hours ahead forecast over a year is shown in Fig. 3.9. The percentage of number of hours at which a particular PE occurs is plotted for all hours for an error value of 1.5%, 3.00% and 6.0%. It shows that for almost 100% of hours are within a PE value of 6.00%, 80-90% of hours are within a PE value of 3.00% and above 50% of hour are within a PE value of 1.5%. Similarly, for 168-hours ahead forecast 90% of hours are within a PE value of 6.00%, 70% of hours are within a PE value of 3.00%.
Fig. 3.10 shows the percentage of MW error for 24-hours ahead forecast for all hours over a year. Here we find that almost 99% of the whole forecast is within 80MW error range at some hours and the average error comes to 94% at the said MW range. Again, the average error range is found to be 85% within 60MW range and 71% within 40MW range.

Similarly, the percentage of MW error for 168-hours ahead forecast for all hours over a year is shown in Fig. 3.11. Here we find that almost 85% of the whole forecast is within 80MW error range. The average error range is found to be 75% within 60MW range and 60% within 40MW range.

The error distribution for 24-hours ahead forecasting over a year is shown in Fig. 3.12. The distribution is over the range of ±20MW to ±80MW, where the actual load has a maximum value of 3588MW and minimum of 931MW. It can be seen from the figure that most of the results are within ±40MW. The number of hours the forecast results were within ±40MW or less out of the total are 5557 hours. A total of 6843 hours were within ±60MW and 7539 hours were within ±80MW or less. These savings in megawatts at each hour give validity of the forecasting model which in turn can mean a lot of savings for a power utility company in term of money.

Fig. 3.13 shows the standard deviation and MAPE (3-D chart) pictorially for each month over a one-year period taking into account the forecasts for all the hours of a given day of the month. As shown in the figure, the maximum value of the MAPE is below 2% throughout the whole year considered for prediction. The value of the Standard Deviation (SD) is found to be less than 1.6% for the entire forecasts over one year period.

**Peak Load forecast**

The forecasted load and the actual load of 24-hour ahead peak load forecast over 30 days period for both Winter and Fall seasons are shown in Fig. 3.14 (a) and Fig. 3.14 (b). The forecasting results for the peak load show excellent accuracy and the MAPE is found to be less than 1.3% and 1.7% respectively for Winter and Fall seasons over a 90 day period. Fig. 3.15 (a) and Fig. 3.15 (b) shows the forecasted load and the actual load for 168-hour ahead peak load forecast during the same seasons over a 30 day period. The maximum MAPE is found to be less than 2% both in Winter and Fall seasons. In the results presented in Fig. 3.14 and Fig. 3.15, it can be observed that the estimated load follows the actual load during the 30-day period in both the seasons very closely except at some peaks in Fall.
Fig. 3.16 shows the standard deviation and MAPE (3-D chart) pictorially for each month over a one-year period. As shown in the figure, the maximum value of the MAPE is below 2% for 24-hours ahead peak load forecast throughout the year considered for prediction. Similarly, for the 168-hours ahead peak load forecast it is below 2.5%. The value of the standard deviation is found to be less than 1.3% and 2.2% respectively for both 24-hours and 168-hours ahead peak load forecasts spanning a data over one year period.

3.5 Discussions

From the results of load forecasts presented above, it is observed that the functional link net approach provides accurate forecasts for all seasons under changing weather conditions throughout a year. This is mainly due to the nonlinear ARMA modeling and superior tracking performance of the auto-enhanced net which is capable to track load and weather condition changes. The mean absolute percentage of error (MAPE) is around 2% and could rise to 2.5% during seasonal changes. The accuracy obtained for 168-hour ahead forecast is found to be significant in comparison to other methods. As the algorithm provides hourly forecast, the peak load forecasts are obtained as a consequence of the 24-hour or 168-hour ahead load predictions in a continuous basis starting from January 1st of the year of interest. The proposed FLN model used for forecasting of electric load takes only 200 iterations to train the network in Spring and Summer seasons. During Winter and Fall seasons, when the data does have some noise content and the load variation is relatively high, the network takes nearly 250 iterations to converge the training error to a value less than $10^{-14}$. Fuzzy-Neural Networks (FNN) model proposed by Lin et al.[3.19] takes nearly 2800 iterations to train the network having four rule nodes and the number of iterations goes up to 5000 to achieve the desired performance with a network having seven rule nodes. This shows the effectiveness of the proposed flat-net model in terms of convergence speed.

3.6 Conclusion

The paper presents an on-line load forecasting technique using functional link neural network. The short-term load forecasting model combines the familiar autoregressive moving average time series model with the theory of flat net computing. The functional link net based load forecasting produces a robust and accurate on-line forecast and is capable of taking the weather and seasonal variations using an adaptive mechanism built into the network.
Numerical results obtained with a load data from a typical Virginia utility in U.S.A. reveal the superior performance of the autoenhanced net in predicting the load in 24-hours and 168-hour ahead time frame.
3.1(a) Schematic illustration of Function Link

3.1(b) Flat Net architecture

Figure 3.1 Functional Link Net schematic representation.

Fig. 3.2 Load Forecasting Model

P.B. — Polynomial Block
F.B. — Functional Link Net Block
H.B. — Harmonic Block
Fig. 3.3 MAPE and PEs for 24-hour Ahead Load Forecast over one week period in Winter, Spring Summer and Fall
Fig. 3.4 Estimated vs Forecasted Load for 24-hours ahead forecast over one week period:
(a) Winter (6th-12th, January), (b) Summer (14th-20th, July), (c) Spring (7th-13th, April),
(d) Fall (6th-12th, October).

Fig. 3.5 MAPE and PEs for 168-hours ahead forecast over one week period in Winter and Spring.
Fig. 3.6 168-hours ahead Actual vs Predicted load over the same one week period for (a)Winter and (b)Spring.

Fig. 3.7 Number of same MAPE-values, occurring throughout a year for both 24-hours and 168-hours ahead load forecast.

Fig. 3.8 Average MAPE of 4 seasons for all day types (Monday through Sunday), for (a) 24-hours ahead and (b) 168-hours Ahead Load Forecast
Fig. 3.9 Absolute Percentage of Error for 24-hours and 168-hours Lead Forecast over a year for all Hours.

Fig. 3.10 Percentage of MW Error for 24-hours Lead Forecast over a year for all Hours.

Fig. 3.11 Percentage of MW Error for 168-hours Lead Forecast over a year for all Hours.

Fig. 3.12 Error Distribution for 24-hours ahead forecast over a year.

Fig. 3.13 Standard Deviation and MAPE for each month over one-year period for 24-hours ahead forecast.
Fig. 3.14 24-hours ahead Peak Load forecast over 30-days period: (a) Winter, (b) Fall season

Fig. 3.15 168-hours ahead Peak Load forecast over 30-days period: (a) Winter, (b) Fall season

Fig. 3.16 MAPE and Standard Deviation for each month over one-year period for 24 and 168-hours ahead Peak Load forecast.
A.3.1 Overview of Functional-Link Neural Networks

There have been a lot of research activities on Artificial Neural Networks (ANNs) throughout the world since the middle of 1980s. Numerous advances have been made in developing intelligent systems, some inspired by biological neural networks. Many researchers have designed ANNs to solve a variety of problems in pattern recognition, forecasting, optimization, associative memory, and control.

Some conventional approaches have been proposed to solve these problems. Although successful applications can be found in certain well-constrained environments but, none are flexible enough to perform well outside their domain. ANNs provide exciting alternatives, and many applications could benefit from using them. Artificial neural networks are used to mimic many desirable human characteristics like- massive parallelism, distributed representation, learning ability, generalization ability, adaptivity, inherent contextual information processing, and fault tolerance.

This can also be used in the problem of load forecasting where, we see - how a set of \( n \) samples \( \{y(t_1), y(t_2), \ldots, y(t_n)\} \) in a time sequence, \( t_1, t_2, \ldots, t_n \), can predict the sample \( y(t_{n+1}) \) at some future time \( t_{n+1} \).

![Graph showing stock values and time sequence](image-url)
The ANNs are capable of generating complex mapping between the input and the output space and thus, arbitrarily complex nonlinear decision boundaries can be formed by these networks. Generally, ANNs are characterized by their network topology, i.e. by the number of interconnections, the node characteristics that are classified by the type of nonlinear elements used and the kind of learning rules implemented.

A basic building block of nearly all artificial neural networks, and most other adaptive systems, is the adaptive linear combiner. The adaptive linear combiner is diagrammed in figure A.3.1 below. The ANNs are composed of numerous simple nonlinear computational elements called nodes or neurons operating in parallel and are arranged in patterns reminiscent of biological neural networks. In each node, a weighted sum of its inputs is computed and then, passed through a nonlinearity to produce its output.

In the next section we describe a system architecture and a network computational approach compatible with a goal of devising a general-purpose artificial neural-net computation. However, we want to emphasize the advantage that accrue from simplification of net architecture. We find this results in a network with faster convergence rate and lesser computational load than an Multi-Layer Perceptron (MLP) structure. This network is termed as Functional-Link Network (FLN).

Figure A.3.1. Adaptive linear Combiner

A.3.2 Functional-Link Neural Networks

Originally, the Functional Link Neural network (FLN) was proposed by Pao [3.18]. He has shown that, this network may be conveniently used for functional approximation and pattern classification. This network is a flat net, and the need of the hidden layer is removed.
here. Hence, the learning algorithm used in this network becomes very simple. In an MLP, the backpropagation (BP) algorithm used is computationally intensive. This is due to, the back propagation of equivalent error to each of the nodes of the hidden layers, required for updating of the weights of the network. With the removal of the hidden layer in the FLN, substantial saving of computational load compared to an MLP can be achieved without sacrificing any mapping and representational capabilities. The functional expansion effectively increases the dimensionality of the input vector and hence the hyperplanes generated by the FLN provides greater discrimination capability in the input pattern space. Although, the original pattern may still exist in the augmented input pattern vector, the enhancement presents alternative regions for discrimination. In this case the augmentation can become a dominant factor, having significant connection weights, while the weights of nonseparable components are diminished in importance [3.21].

The FLN approach may be viewed same as that the MLP [3.22], except that the computation in the hidden layer are transferred to the input layer itself, considering them to be enhancements to the input pattern. Pao et al. have reported identification and nonlinear system using a FLN [3.21]. They have taken the higher order terms of the input vector for functional expansion and augmented it to the input vector and then used for the identification problem. For the identification of nonlinear plants, good performance is achieved using this FLN structure. The learning and generalization characteristics of a random vector FLN is reported in [3.22]. In this structure, the enhancement is carried out by augmenting the original pattern with the output of several nonlinear activation nodes. The input to an activation node was the inner product of an appropriately chosen random vector and the original pattern. The activation function selected was a sigmoid function. It has been shown that, the system identification and control using this structure is quite effective.

Herrmanninger and Pao [3.23] have reported the detection and classification of underwater acoustic transients using a FLN. The input pattern was enhanced by the joint activation, i.e. the entire input vector was multiplied by its transpose resulting an outer vector product which takes advantage of any interrelationships between the components of the input pattern. This augmented input vector thus becomes a representation of the input space within a higher dimensional space in which separability between classes is easier.

Chen and Billings [3.24] have reported very effective nonlinear dynamic system modeling and identification using three different ANN structures, namely - an Multi Layered
Perceptron structure (MLP), a Radial Basis Function (RBF) network and a Functional Link Network (FLN), and have obtained satisfactory results with all the three networks. The RBF network is a two layer structure [3.4], [3.28]; and the nonlinear processing of the signal in each node is carried out, with reference to a parameter vector. The output layer is essentially a set of linear combiners. The identification performance with this network was found to be quite satisfactory. In the FLN structure, they have used polynomials for functional expansion and a linear output node for system identification and have shown that this structure provides good agreement between the iterative model output and the system output. Further, there are several works reported on system identification using RBF networks in [3.25] and [3.26] using MLP networks in [3.27], [3.28], [3.29], [3.30], and [3.31].

Recently, recurrent neural network (RNN) models have been proposed for system identification, control, time-series prediction, channel equalization etc. with quite satisfactory results. The main advantage of these models is that with only a few number of nodes complex non-linear mapping is possible. In a RNN architecture, feedback connections between the nodes of the same layer or to the nodes of the preceding layers exist. However, exploitation of mapping and approximation capabilities of this structure have been hindered by the complexity of learning algorithms. The calculation of the dynamic derivatives of a RNN’s output with respect to its weights is computationally intensive. It is because of this reason that, on-line application of this network is very difficult. Therefore, we have preferred to use the objectives of a functional link network into a load forecasting model for an effective real time implementation.

The theoretical basis of our approach is the theorem derived by Hornik, Stinchcombe and White. They established that standard multilayer feed-forward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another, to any desired degree of accuracy provided that sufficiently hidden layers are available. In that sense, multilayer feed-forward networks are universal approximators.

The following definition and theorems from Hornik, Stinchcombe, and White’s work leads to one result, which is of principal interest to us.

Their definition states that for any measurable function \( G(\cdot) \) mapping \( \mathbb{R} \) to \( \mathbb{R} \) and \( \in \mathbb{N} \), let \( \Pi(G) \) be the class of functions
\[
\left\{ f: R' \rightarrow R: f(x) = \sum_{j=1}^{q} \beta_j \prod_{k=1}^{l_j} G(A_{jk}(x)), x \in R', \beta, \epsilon R, A_{jk} \in \mathcal{A}, l, \epsilon N, q = 1, 2, \ldots \right\} \tag{A.3.1}
\]

where, \( A_{jk} \) is the set of all affine functions from \( R_r \) to \( R \). For the special case of \( l_j = 1 \), we have the \( \Sigma \) networks. Specifically, the theorems holds if \( f \) is a mapping \( R' \rightarrow R \) (from \( r \)-dimensional space to one-dimensional space), then we can approximate that function arbitrarily well by

\[
f = \sum_{j=0}^{q} \beta_j G(A_j, x + b_j) \tag{A.3.2}
\]

where, \( (A_j, x + b_j) \) represents a linear transformation of the input pattern vector \( x \). We can understand the mapping of the above equation in terms of feed forward net that must learn both the two sets of weights\( \{ \beta_j \} \) and \( \{ A_j \} \) and thresholds \( b_j \), usually through back propagation of error.

(a) Functional-link net
The random-vector version of the functional-link generates \( \{A_j\} \) and \( \{b_j\} \) randomly, and must learn only \( \beta_j \). This results in a flat-net architecture for which only weights \( \{\beta_j\} \) must be learned. Learning by quadratic optimization, is extremely rapid.

Figure A.3.2 compares the architectures of the single hidden-layer net (Fig. A.3.1-a) and the random vector version of the functional-link net (Fig. A.3.1-b). In the functional-link net, the vectors \( A_j \) and thresholds \( b_j \) are generated randomly, not learned.

**A.3.3 System architecture**

The random-vector version of the functional-link net is well suited to large-scale hardware implementation. Figure A.3.3 schematically illustrates a network system architecture.
A.3.4 Overview of algorithms

The functional-link net can support many different functionalities with the same system architecture and with only minor changes in the algorithm.

For mapping or supervised learning, inputs are enhanced to \( x_p \) with elements \( x_{p1}, x_{p2}, \ldots, x_{pk} \). The target outputs are \( t_{pk} \) and the weights are \( w_{kj} \). For functional-link nets, the outputs \( t_k \) can be treated independently of each other. Therefore, we need only consider one output \( t \) and weight \( \beta_j \). All other outputs are treated in the same manner. Initially, the algorithm assigns the weights \( \beta_j \) random values. It calculates the output \( o_p \) linearly as \( o_p = \sum \beta_j x_{pj} \). For each input pattern the changes in the weights are taken to be:

\[
\Delta \beta_{pj} = \eta (t_p - o_p) x_{pj} \tag{A.3.3}
\]

The changes are calculated for all the patterns in the training set, and after each such presentation the weights are updated according to:

\[
\beta_j (k+1) = \beta_j (k) + \sum_p \Delta \beta_{pj} \tag{A.3.4}
\]
Updating is continued until the values of the weights $w_j$ do not change significantly. The value of the parameter $\eta$ may be increased as $(t_p - o_p)$ decreases. For self-organization or clustering, input patterns $y_p$ are enhanced to pattern $x_p$ and fed to the net one at a time. Initially, there is no cluster prototype. Pattern $x_1$ is therefore a cluster prototype with link weights $b_j = x_{yp}$. The algorithm introduces pattern $x_2$ to the net and evaluates the distance from $x_2$ to $x_1$ according to the expression:

$$d(x_1, x_2) = \sum_j |x_{ij} - x_{2j}|$$  \hspace{1cm} (A.3.5)

If $d(x_1, x_2) < \text{cluster radius}$, then $x_2$ belongs to the same cluster as $x_1$ and the cluster prototype value $b_j$ is updated. In general, after $k$ cluster centers have been formed, a new pattern is assigned to a particular cluster $k$ if $d(x, x_k) < d(x, x_m)$ for $m=1,2, \ldots$, and $m \neq k$, and if $d(x, x_k) < \text{cluster radius}$. Otherwise, a new cluster is formed. At all times, each cluster knows how many patterns are members of the cluster, and the cluster prototype is updated accordingly to

$$b_j(n+1) = \frac{n}{n+1} b_j(n) + \frac{n}{n+1} x_j$$ \hspace{1cm} (A.3.6)

where, $n$ is the number of patterns already associated with that cluster. For associative recall, each pattern $y_p$ may be enhanced to $x_p$ form. This generally results in higher memory capacity. For a set of stored patterns $x_p$, the Lyapunov function is

$$E(x) = -\sum e^{-\alpha/2}(x_p - x)'(x_p - x)$$ \hspace{1cm} (A.3.7)

For any cue $x$, the elements $x_j$ are changed repeatedly according to:

$$\Delta x_j = -\eta \frac{\partial E}{\partial x_j}$$

$$= -\alpha \eta \sum_p (x_{yp} - x_j) e^{-\alpha/2 \sum_j (x_{yp} - x_j)^2}$$

$$= -\alpha \eta \sum_p (x_{yp} - x_j) E(x)$$ \hspace{1cm} (A.3.8)

The random-vector version of the functional-link net is eminently suitable for realization with simple hardware network architectures. Numerous and significant advantages accrue from using a flat net, including rapid quadratic optimization in the learning of weights,
simplification in hardware as well as in computational procedures, and uniform system architecture for all four functionalities.
3.7 References


