International Journal of Computer Applications (0975 – 8887)
Volume 30– No.4, September 2011

Artificial Neural Network based Short Term Load Forecasting of Power System

Salman Quaiyum, Yousuf Ibrahim Khan, Saidur Rahman, Parijat Barman
Department of Electrical and Electronic Engineering,
American International University – Bangladesh,
Banani, Dhaka – 1213.

ABSTRACT

Load forecasting is the prediction of future loads of a power system. It is an important component for power system energy management. Precise load forecasting helps to make unit commitment decisions, reduce spinning reserve capacity and schedule device maintenance plan properly. Besides playing a key role in reducing the generation cost, it is also essential to the reliability of power systems. By forecasting, experts can have an idea of the loads in the future and accordingly can make vital decisions for the system. This work presents a study of short-term hourly load forecasting using different types of Artificial Neural Networks.

General Terms

Artificial Intelligence, Neural Networks.

Keywords

Load Forecasting, Power System, Particle Swarm Optimization.

1. INTRODUCTION

Load forecasting is one of the central functions in power systems operations and it is extremely important for energy suppliers, financial institutions, and other participants involved in electric energy generation, transmission, distribution, and supply. Load forecasts can be divided into three categories: short-term forecasts, medium-term forecasts and long-term forecasts. Short-term load forecasting (STLF) is an important part of the power generation process. Previously it was used by traditional approaches like time series, but new methods based on artificial and computational intelligence have started to replace the old ones in the industry, taking the process to newer heights.

Artificial Neural Networks are proving their supremacy over other traditional forecasting techniques and the most popular artificial neural network architecture for load forecasting is back propagation. This network uses continuously valued functions and supervised learning i.e. under supervised learning, the actual numerical weights assigned to element inputs are determined by matching historical data (such as time and weather) to desired outputs (such as historical loads) in a pre-operational "training session". The model can forecast load profiles from one to seven days.

Evolutionary algorithms such as, Genetic Algorithm (GA) [1, 2], Particle Swarm Optimization (PSO) [3-5], Artificial Immune System (AIS) [6], and Ant Colony Optimization (ACO) [7] have been used for training neural networks in short term load forecasting applications. These algorithms are better than back-propagation in convergence and search space capability.

2. ARTIFICIAL NEURAL NETWORK

An Artificial Neural Network (ANN) is a mathematical or computational model based on the structure and functional aspects of biological neural networks. It consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. An ANN mostly is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.

2.1 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a class of neural network where connections between units form a directed cycle. Unlike feed-forward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. A recurrent neural network consists of at least one feedback loop. It may consist of a single layer of neurons with each neuron feeding its output signal back to the inputs of all the other neurons.

In this work, Elman's recurrent neural network has been chosen as the model structure which has shown to perform well in comparison to other recurrent architectures [8]. Elman's network contains recurrent connections from the hidden neurons to a layer of context units consisting of unit delays which store the outputs of the hidden neurons for one time step, and then feed them back to the input layer.

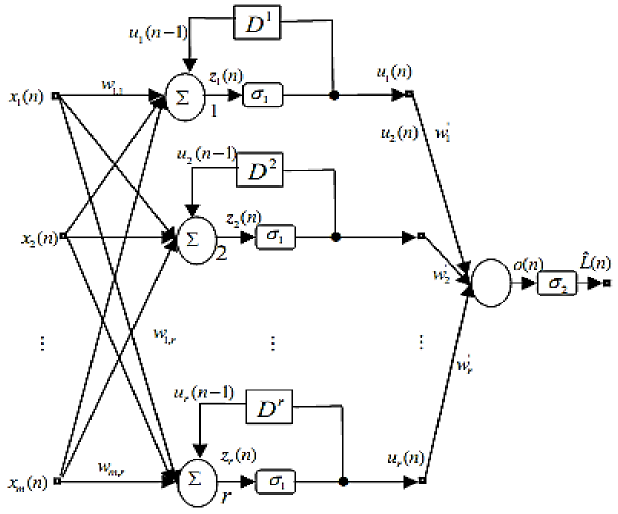

Fig 1: Elman recurrent neural network topology.

Figure 1 is an Elman recurrent neural network topology where **w** denotes a vector of the synaptic weights, **x** and **u** are vectors of the inputs to the layers, **m** is the number of input variables, and **r** is the number of neurons in the hidden layer.

The weighted sums for the hidden and the output layers are:

$$Z_k(n) = \sum_{l=1}^{m} w_{l,k}\, x_l + w_{D^k} u_k(n-1) (1)$$

$$o(n) = \sum_{k=1}^{r} w'_k u_k (2)$$

where,k = [1,r], n = [1,N], and N is the number of data points used for training of the model. The outputs of the neurons in the hidden layer and output layer are computed by passing the weighted sum of inputs through the tan sigmoid and pure linear transfer functions respectively.

Mathematically, the outputs of the hidden layer and the output layer can be defined as:

$$\sigma_1\big(z_k(n)\big) = \frac{1}{1+e^{-z_k(n)}} = u_k(n) (3)$$

$$\sigma_2 o(n) = Ko(n) = L(n) (4)$$

where,K is a coefficient of the pure linear transfer function.

Another training parameter considered is the momentum factor as an attempt to prevent the network to get stuck in a shallow local minimum. Equation (5) shows how the synoptic weights are adjusted and how the network determines the value of the increment$\Delta w(n)$on the basis of the previous value of the increment

$$\Delta w(n) = \frac{\delta \epsilon(n)}{\delta w} + \psi\, \Delta w(n) (5)$$

The other important training parameter is the learning rate which controls the amount of change imposed on connection weights during training and to provide faster convergence.

Mathematically, the weights are updated using the equation:

$$w(n+1) = w(n) - \eta \Delta w(n) (6)$$

where, η is the learning rate.

3. PARTICLE SWARM OPTIMIZATION

PSO as a tool that provides a population based search procedure in which individuals called particles change their position (state) with time. In a PSO system, particles move around in a multidimensional search space. During flight, each particle carries out position adjustment in accordance to its own experience, to the experience of a neighboring particle. This helps make use of the best position of the particle encountered by itself and its neighbor. Thus, a PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation.

The basic concept is that for every time instant, the velocity of each particle, also known as the potential solution, changes between its **pbest**(personal best)and **lbest**(local best) locations. The particle associated with the best solution (fitness value) is the leader and each particle keeps track of its coordinates in the problem space. This fitness value is stored which is referred to as **pbest**. Another "best" value tracked by the particle swarm optimizer, is the best value obtained so far by any particle in the neighbors of the particle. This location is called **lbest**. When a particle takes all the population as its topological neighbors, the best value is a global best and is called **gbest**.

$$presLocation = prevLocation + V_i\, \Delta t (7)$$

$$V_i = \omega\, V_{i-1} + c_1\, rand()\, (pbest - presLocation) + c_2\, rand()\, (gbest - presLocation)\ (8)$$

where, $\mathbf{V_i}$ is the current velocity, $\mathbf{\Delta t}$ defines the discrete time interval over which the particle will move, **ω** is the inertia weight, $\mathbf{V_{i-1}}$ is the previous velocity, **presLocation** is the present location of the particle, **prevLocation** is the previous location of the particle and **rand( )** is a random number between 0 and 1. $\mathbf{c_1}$ and $\mathbf{c_2}$ are the learning factors, stochastic factors and acceleration constants for "gbest" and "pbest" respectively.

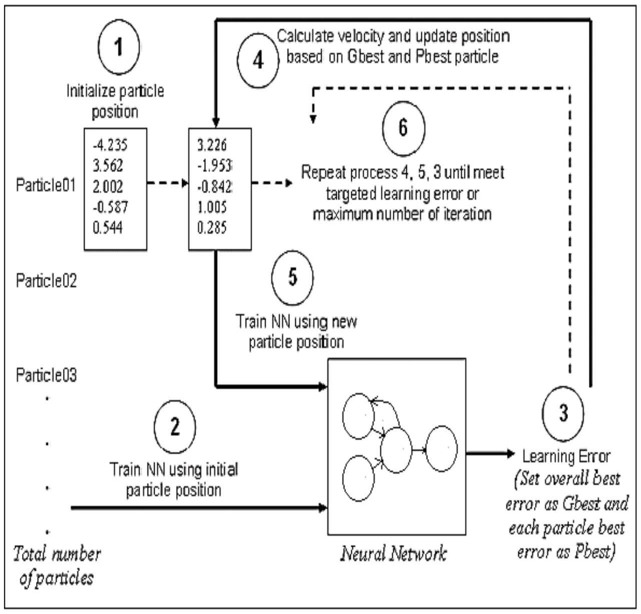
1
Initialize particle position
4
Calculate velocity and update position based on Gbest and Pbest particle
Particle01
-4.235
3.562
2.002
-0.587
0.544
3.226
-1.953
-0.842
1.005
0.285
6
Repeat process 4, 5, 3 until meet targeted learning error or maximum number of iteration
Particle02
5
Train NN using new particle position
Particle03
2
Train NN using initial particle position
3
Learning Error
(Set overall best error as Gbest and each particle best error as Pbest)
Total number of particles
Neural Network

Fig 2: PSO-ERNN Learning Process

Figure 2 shows the learning process of PSO-ERNN (Particle Swarm Optimized – Elman Recurrent Neural Network). The learning is initialized with a group of random particles in step 1, which are assigned with random PSO positions (weight and bias). The PSO-ERNN is trained using the initial particles position in step 2. Then, it produces the learning error (particle fitness) based on initial weight and bias in step 3. The learning error at current epoch is reduced by changing the particles position, which updates the weight and bias of the network. The "pbest" and "gbest" values are applied to the velocity update according to (7) to produce a value for positions adjustment to the best solution or targeted learning error in step 4. Step 5 has the new sets of positions (weight and bias) produced by adding the calculated velocity value to the current position value. Then, these new sets of positions are used to produce new learning error in PSO-ERNN. This process is repeated until the stopping conditions of either minimum learning error or maximum number of iterations are met which is shown in step 6.

3.1 Global best PSO

Global version of PSO is faster but might converge to local optimum for some problems. The Local version, though slower does not easily get trapped into a local optimum. We implemented the global version to achieve a quicker result. The position of a particle is influenced by its best visited position and the position of the best particle in its neighborhood.

Particle position, x_i, was adjusted using

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (9)$$

where the velocity component, v_i, represents the step size.

For the basic PSO,

$$v_{i,j}(t+1) = w\, v_{ij}(t) + c_1 r_{1,j}(t)\left(y_{i,j}(t) - x_{i,j}(t)\right) + c_2\, r_{2,j}(t)(\hat{y}_i(t) - x_{ij}(t)) \quad (10)$$

where, w is the inertia weight, c_1 and c_2 are the acceleration coefficients, $r_{1,j}$, $r_{2,j} \sim U(0,1)$, y_i is the personal best position of particle i, and $\hat{y}_i$ is the neighborhood best position of particle i.

If a fully-connected topology is used, then $\hat{y}_i$ refers to the best position found by the entire swarm. That is,

$$\hat{y}_i(t) \in \{y_o(t), y_1(t) \ldots y_s(t)\} = min\{f(y_o(t)), f(y_1(t)) \ldots f(y_s(t))\} \quad (11)$$

where, s is the swarm size.

The pseudo-code for PSO is shown below:

```
for each particle $i \in 1,...,s$ do
Randomly initialize $x_i$
  Set $v_i$ to zero
  Set $y_i = x_i$
endfor
Repeat
  for each particle $i \in 1,...,s$ do
Evaluate the fitness of particle $i$, $f(x_i)$
    Update $y_i$
    Update $\hat{y}$ using equation (11)
for each dimension $j \in 1,...,N_d$ do
Apply velocity update using (10)
    endloop
Apply position update using (9)
  endloop
Until some convergence criteria is satisfied
```

4. DATA COLLECTION AND PREPROCESSING

A simple data collection method was employed to ensure adequate historical samples of the load. Load Data used in this work were collected from the Bangladesh Power Development Board (BPDB) and ISO New England.

4.1 Data Scaling Methods

Data scaling is carried out in order to improve interpretability of network weights. The equation below has been adopted and implemented to normalize the historical load data.

$$Normalizedoutput = \frac{Actualvalue}{Maximumvalue} \quad (12)$$

The load value is normalized into the range between 0 and 1 and then the neural networks are trained using the suitable algorithm. Neural networks provide improved performance with the normalized data. The use of original data as input to the neural network may increase the possibility of a convergence problem.

4.2 Data Storage

Data can be stored in many ways. In this work, the collected data was stored in Microsoft Excel worksheets. The worksheets were then imported into MATLAB using the command "xlsread".

4.3 Composition of the Input Vector of the prediction models

The structure of the input vector specifies the selected endogenous and exogenous variables. In the work of T. Gowri Monohar et al. [9] five inputs were selected from the previous day and five each from the previous weeks on the same day were selected to predict the load of the next day. If data points were insufficient, the forecasting would be poor. If data points were useless or redundant, modeling would be difficult or even skewed [10].

The input vector (IV) structure used here, consists of two previous hours active power values, *L(t-1)* and *L(t-2)* and some homologous consumption past load values of the previous week *L(t-168)* and *L(t-169)*. The inclusion of these values provides information regarding the consumption trend in the past homologous periods [11]. It was found that load for 24 hours and 168 hours are highly correlated. The structure of IV is given in Figure 3.

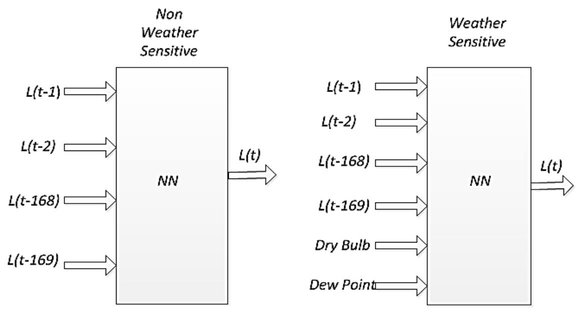

Fig 3: Composition of the input vector (IV) (non-weather and weather sensitive model)

The ISO New England historical load data was first collected from their website. The load data was taken every hour for a period of one week. The training data was defined from Monday, 31st March till 6th April, 2008 and the corresponding target was defined for the period from 7th to 13th April, 2008. The models were also evaluated for generalization capability, thus testing data set was defined from the first week of March 3rd to 9th. After correlation analysis, Dry Bulb and Dew Point temperatures were used as the input for the Elman Weather Sensitive Model.

5. SIMULATION AND RESULTS

For evaluating our proposed load forecast model, several neural network architectures were implemented in MATLAB version 7.10.0.499 (R2010a). Feed Forward Network and Elman Recurrent Network – these two networks were implanted according to their default architectures as provided in MATLAB. In addition, Jordan Recurrent Network was created using the custom network creation process. Before starting the training of the networks, each layer was individually initialized using the *initnw* function.

Then each model was trained using *traingdx* (Gradient descent with momentum and adaptive rule backpropagation) training algorithm with the help of Neural Network Training tool. After that each network was simulated and their performance was observed. Finally, Elman Network was trained using Particle Swarm Optimization method. Mean Square Error (MSE) performance measure function was employed along with other functions like MPE (Mean Percentage Error) and MAE (Mean Absolute Error).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}[L_{actual}(n) - L_{predicted}(n)]^2 (13)$$

$$MPE = \frac{1}{N}\sum_{i=1}^{N}\frac{[L_{actual}(n) - L_{predicted}(n)]}{L_{actual}} X\ 100\% (14)$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}[|(n) - L_{actual}\ (n)|(15)$$

where, $L_{actual}\,(n)$ is the actual load, $L_{predicted}\,(n)$ is the forecasted value of the load, and N is the number of data points.

5.1 Daily Maximum Demand Prediction

For maximum demand prediction, the data that was collected from BPDB was insufficient for training of the network using *traingdx*. Therefore, the network was trained using *trainlm* (Levenberg-Marquardt backpropagaton). This network performed better – it achieved smaller MSE than the network trained with *traingdx*.

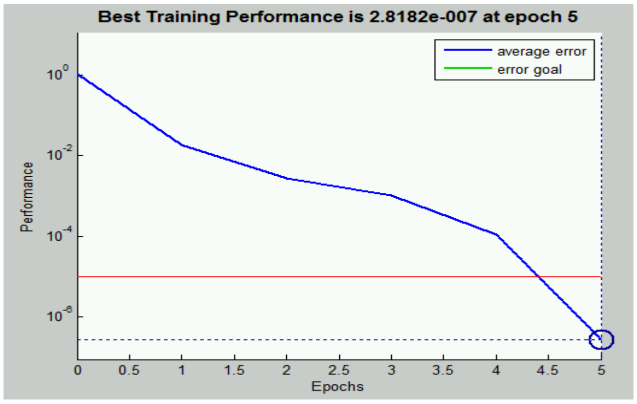

Fig 4(a): The performance goal met

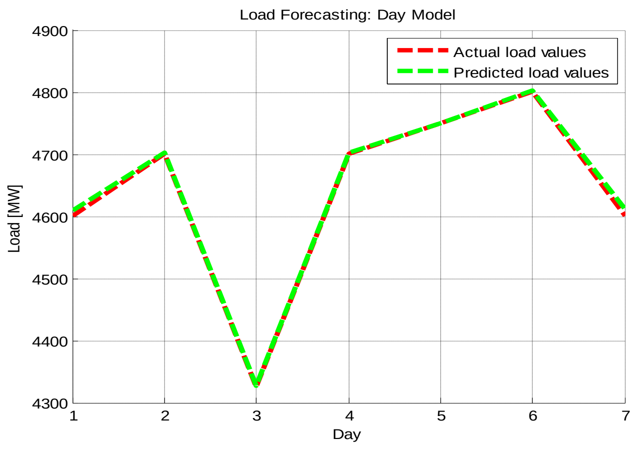

Fig 4(b): Forecasting Result of the maximum demand

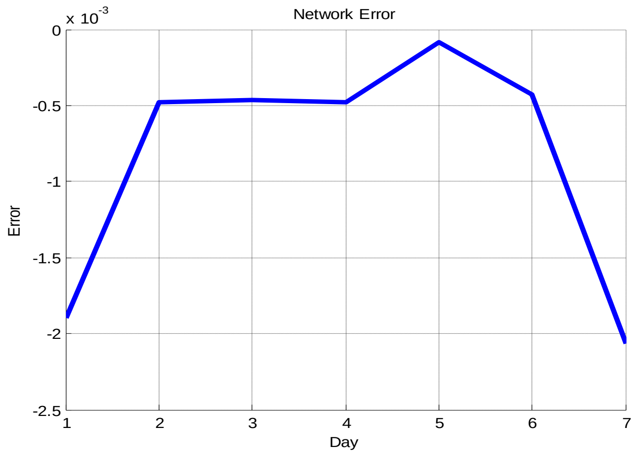

Fig 4(c): Actual Network Error

5.2 Forecasting with ISO New England Load Data

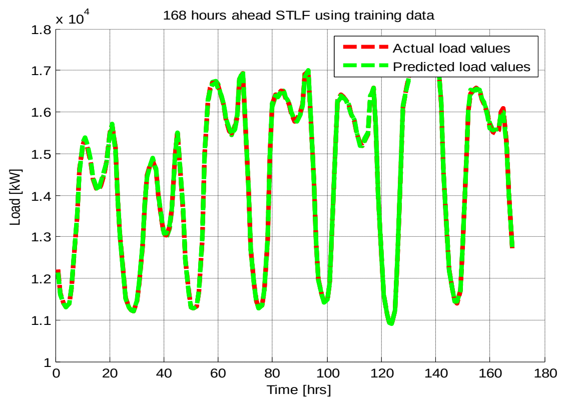

Fig 4(d): Output of Feedforward Network

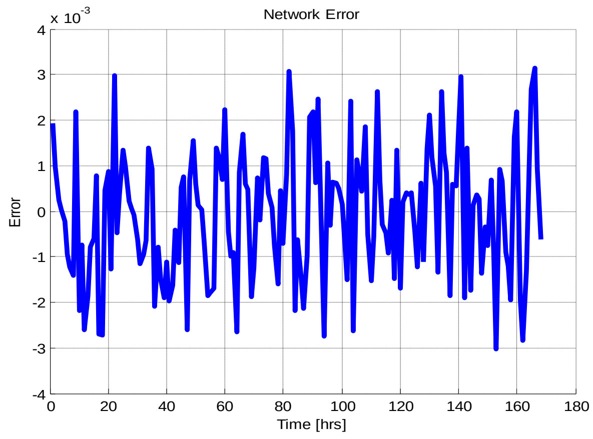

Fig 4(e): Network Error (Feedforward)

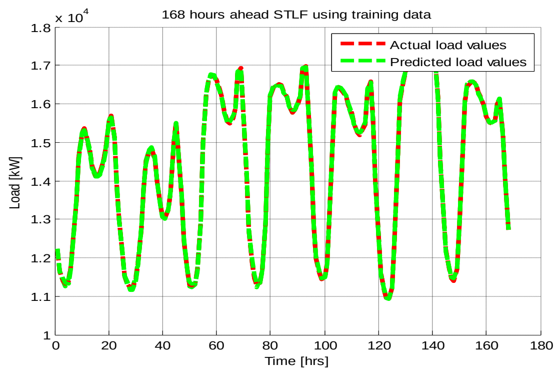

Fig 4(f): Output of Elman Network

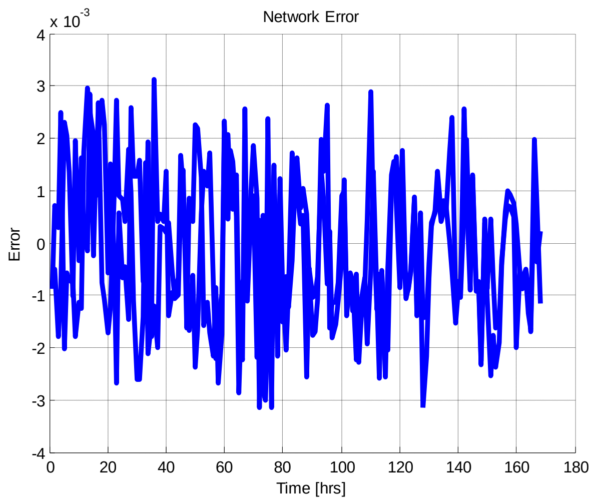

Fig 4(g): Network Error (Elman)

5.3 Particle Swarm Optimized Elman Recurrent Neural Network (PSO-ERNN)

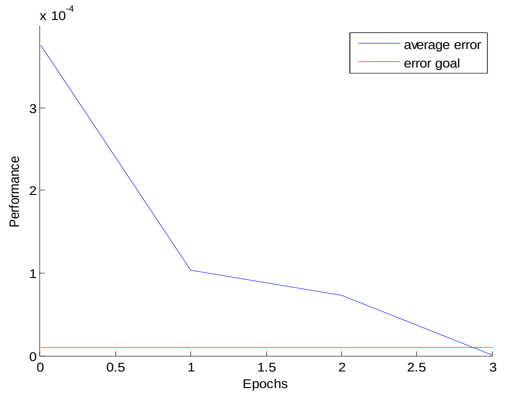

Fig 4(h): The performance goal met

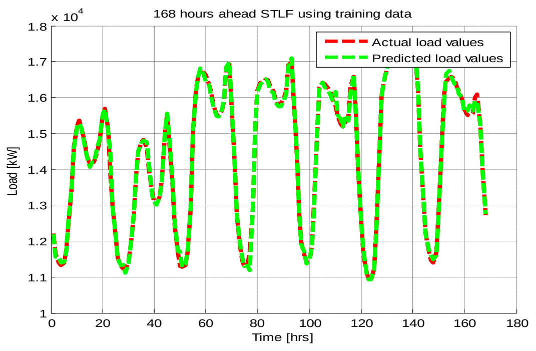

Fig 4(i): Output of PSO-ERNN

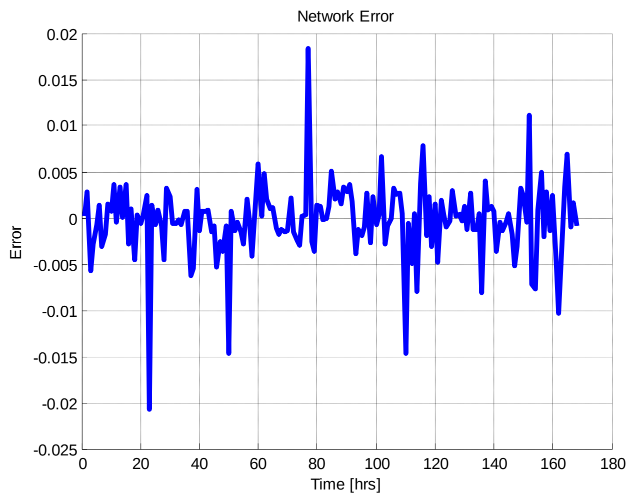

Fig 4(j): Network Error (PSO-ERNN)

Table 1: Comparison between different network performances

Network	**Training Epoch**	**MSE**	**MAE**	**MPE**
Feedforward	8	2.0351e-6	0.0012	0.0011
Jordan	3	4.6124e-6	0.0018	0.0021
Elman	7	1.868e-6	0.0012	9.1269e-4
Elman (WS)	8	2.3895e-6	0.0013	1.8288e-8
PSO-ERNN	3	1.6296e-5	2.6042e-3	7.1934e-3

From the above table, it can be seen that PSO-ERNN is faster but slightly more prone to errors (with MSE); but this model could outperform others if more data sets were used. This network could handle large amount of data in a short amount of time. The Elman Weather sensitive model performs better but it takes longer time than Simple Elman network. Dry bulb and dew point temperatures had a very weak correlation with the load, so their presence did not have any significant effect on forecasting. Providing different weather parameters would have definitely increased the network performance. Overall this table shows that there is always a trade-off between the networks and depends on the amount of data, quality of data, time required and most importantly design requirements and designers.

6. CONCLUSION

Several neural network models for short-term load forecasting were studied in this work. According to the discussion and the comparison of model forecast accuracy shows that Particle Swarm Optimized Elman Recurrent Neural Network (PSO-ERNN) is the best model for 168 hours ahead load forecasting. This type of network can be very efficient in terms of predicting future loads.

Though the simulations seemed very promising, the models developed here still need to be tested on data sets from other sources, so that reliability of these models can be verified for other load patterns.

For future works, the error in the network can be further reduced if a larger dataset is used for network training. Also, the load forecasting model can be improved by including other weather parameters like temperature, wind speed, rainfall etc.

7. ACKNOWLEDGMENT

We would like to thank Engr. K. M. Hassan, CSO, BPDB for providing us with necessary information.

8. REFERENCES

[1] Heng, E.T.H., Srinivasan, D., Liew, A.C., “Short term load forecasting using genetic algorithm and neural networks”, Energy Management and Power Delivery, 1998. Proceedings of EMPD '98. 1998 International Conference on, Volume 2, 3-5 March 1998, Page(s):576 - 581 vol.2.

[2] Worawit, T., Wanchai, C., “Substation short term load forecasting using neural network with genetic algorithm”, TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering; Volume 3, 28-31 Oct. 2002, Page(s):1787 - 1790 vol.3.

[3] Azzam-ul-Asar, ul Hassnain, S.R., Khan, A., “Short term load forecasting using particle swarm optimization based ANN approach”, Neural Networks, 2007. IJCNN 2007. International Joint Conference on ; 12-17 Aug. 2007, Page(s):1476 – 1481.

[4] Wei Sun, Ying Zou, Machine Learning and Cybernetics, “Short term load forecasting based on BP neural network trained by PSO”, 2007 International Conference on; Volume 5, 19-22 Aug. 2007, Page(s):2863 – 2868.

[5] Bashir, Z.A., El-Hawary, “Short-term load forecasting using artificial neural network based on particle swarm optimization algorithm”, Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on; 22-26 April 2007, Page(s):272 – 275.

[6] You Yong, Wang Sun'an, Sheng Wanxing, “Short-term load forecasting using artificial immune network”, Power System Technology, 2002. Proceedings. PowerCon 2002. International Conference on; Volume 4, 13-17 Oct. 2002, Page(s):2322 - 2325 vol.4.

[7] Chengqun Yin, Lifeng Kang, Wei Sun, “Hybrid neural network model for short term load forecasting”, Third International Conference on Natural Computation, 2007.

[8] Almedia L.B. et al., “Parameter adaptation in stochastic optimization”, On-line learning in neural Networks (Ed. D. Saad), Cambridge University Press, 1998.

[9] T.Gowri Monohar and V.C. Veera Reddy, “Load forecasting by a novel technique using ANN”, ARPN Journal of Engineering and Applied Sciences, VOL. 3, NO. 2, April 2008.

[10] Lendasse, J. Lee, V. Wertz, M. Verleysen, “Time Series Forecasting using CCA and Kohonen Maps - Application to Electricity Consumption”, ESANN'2000 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 329-334.

[11] P.J. Santos, A.G. Martins, A.J. Pires, J.F.Martins, and R.V. Mendes, "Short Term load forecast using trend information and process reconstruction”, International Journal of Energy Research, 2006; 30:811-822.

[12] Simaneka Amakali, "Development of models for short-term load forecasting using artificial neural networks", Cape Peninsula University of Technology Year, 2008.

[13] Ayca Kumluca Topalli, Ismet Erkme, and Ihsan Topalli, “Intelligent short-term load forecasting in Turkey”, International Journal of Electrical Power & Energy Systems, Volume 28, Issue 7, September 2006, pp. 437-447.

[14] Mohsen Hayati and Yazdan Shirvany, “Artificial Neural Network Approach for ShortTerm Load Forecasting for Illam Region”, International Journal of Electrical, Computer, and Systems Engineering, Volume 1, 2007, pp.1307-5179.

[15] T.Gowri Monohar and V.C. Veera Reddy, “Load forecasting by a novel technique using ANN”, ARPN Journal of Engineering and Applied Sciences, VOL. 3, NO. 2, April 2008.

[16] George I Evers, “An automatic regrouping mechanism to deal with stagnation in particle swarm optimization”, Graduate thesis for the degree of Master of Science, University of Texas-Pan American, pp. 35-40, 2009.

[17] S. Sumathi, Surekha P., “Computational Intelligence Paradigm: Theory and application using MATLAB”, CRC Press, 2010.