MANTECH Publications
Journal of Electrical Power System Engineering and Technology
Volume 3, Issue 2, May-August, 2018

Neural Network-based Electricity Load Forecasting using Constructive Technique

Kazi Rafiqul Islam[1], Md. Shahid Iqbal[2], Md. Monirul Kabir[3]
Department of Electrical and Electronic Engineering,
Dhaka University of Engineering and Technology, Gazipur, Bangladesh
***Corresponding Authors' email id**: krafiquleee@gmail.com[1], shahidiqbal_05@yahoo.com[2], munir@duet.ac.bd[3]*

Abstract

This paper presents a new electricity load forecasting (ELF) model based on feed-forward neural network (FFNN) using the constructive technique in course of training. The vital aspect of this model is to determine the FFNN architecture automatically during training in order to forecast the electricity load. Thus, the strength of standard FFNN increases in forecasting the electricity load. Furthermore, the proposed model overcomes efficiently the existing shortcomings of FFNN to predict loads of holidays and fast load changes. We call this model as constructive approach for electricity load forecasting (CAELF) as per short term basis. In order to evaluate the performance of CAELF, the daily electricity load demand data of Spain has been used. Extensive experimental results and comparisons show that CAELF has a significant capability to forecast the electricity load compared to the other standard FFNN models.

Keywords: *Electricity load forecasting (ELF), CAELF, FFNN models.*

I. INTRODUCTION

The forecasting of the operation and control of power systems is sensitive to system demand. The effect of a large forecast error is reflected in terms of over conservative or over risky operation. Thus, improvement in load forecasting accuracy leads to the cost savings and increases in the system security. It is noted that, in power systems, the every next day's power generation needs to be scheduled for the power dispatch. Thereby, the day-ahead

short-term electricity load forecasting (STLF) is a necessary task. In case of reliable planning and operation of the power systems, on the other hand, load forecasting analysis is a very vital issue. Particularly, daily electricity demand as per short-term basis is important in power plant assurance and maintenance, power interchange and task scheduling of both power generation and distribution facilities [1]. The load forecasting accuracy can allow utilities to operate at the minimum cost that may result significant savings in electric power companies. It has been reported in [2] is that a 1% increase in the forecasting error would cause an increase 10 million pounds in operation cost per year of 1984 in the UK. Thus, we can say that, the competitive electricity markets might be looser by a great profit loss because of a little forecasting error.

In terms of time period, forecasting of electricity load can be classified into four ways [1],[3],[4]: (a) long-term forecasting with the time period of more than one year, (b) mid-term forecasting includes six month to one year, (c) short-term load forecasting (STELF) includes one week to six month, and (d) very short-term load forecasting with the time period of shorter than one day. Different categories of forecasting have different purposes, in where this paper provides the short-term electricity load forecasting that serves the next day's unit commitment and reliability analysis. Among these areas, different authors try to perform the electricity load forecast using various techniques, such as, support vector machines (SVMs) [5], pattern recognition [6], fuzzy time series[7], fuzzy inference model [8], fuzzy neural networks [9], and so on. Some recent hybrid techniques, such as, combination of SVM and genetic algorithm (GA) [10] as well as GA and ant colony optimization (ACO) [11], and so on.

A number of approaches presented in the literature (e.g., [12]-[21]), where they try to solve STELF problem using neural networks (NNs). It has been confirmed that, the usage of NN in STELF always outperforms any human-based computational analysis in terms of accuracy, easy maintenance for users. The reason is that, NN has a good capability for mapping between input and output although load (i.e., output) is being increased day by day [4]. Particularly, feed forward NN (FFNN) has been used in [16]-[22] to solve the ELF problem for different regions with a reasonable computational cost. It is noted that, FFNNs are much suitable for mapping static

relationships between inputs and outputs and ultimately providing good results in ELF. However, FFNNs need large historical data and have a limited capability to predict loads of holidays and fast load changes [23]. To overcome the shortcomings of FFNN, a number of efforts have been done in [13]-[16] recently, among which echo state NN, radial basis function NN, recurrent NN, and nonlinear autoregressive NN are used, respectively. It is noted here that, the performances of aforementioned NN models are satisfactory in predicting the electricity load comparing to the FFNN, but computationally expensive. Thereby, huge requirements are necessary for the hardware setups as well as experts are needed for maintenances.

This paper describes a new ELF approach using FFNN, called constructive ELF approach (CAELF). The idea incorporated in this model was originally introduced in our earlier work [24].This approach differs from previous works in a way that, CAELF determines the appropriate NN architecture in advance before the ELF starts using constructive NN training. In contrast to the previous approaches (e.g., [17]-[22]), they generally use a fixed NN architecture with randomly selecting the hidden neuron in the hidden layer during training before the ELF starts. It is well known that, the random selection of hidden neurons affects the generalization performance of NNs. The reason is that, the performance of any NN is greatly dependent on its architecture [25], [26].Thus determining hidden neurons' number automatically provide a novel approach in building learning models using NNs for ELF.

The remainder of this paper is organized as follows. Section 2 describes elaborately about our proposed model (CAELF), including the computational complexity of different stages. Section 3 presents the results of our experimental studies, including the experimental setup, results, analyses, and comparison with other ELF models. Finally, Section 4 concludes the paper with a brief summary and a few remarks.

2. PROPOSED MODEL-CAELF

In this paper, our proposed model CAELF uses a training approach in association with incremental training to find a minimum number of hidden neurons for the NN model. Hidden neurons (HNs) are added simultaneously one by one in constructive fashion during the training process of a NN. If the addition of HN does not improve the NN's accuracy, it is

then removed. The major steps of CAELF are summarized in Fig. 1, which are explained further as follows:

Step 1) At first, choose a feed-forward NN with minimal size. Precisely, size of the input layer and output layer are decided by the total number of input variables and the output load of the given ELF data samples, respectively, whereas, size of the hidden layer is initialized using one hidden neuron.

Step 2) Start the partial training of NN on the training data sample up to τ epoch using the back-propagation (BP) algorithm [27]. The number of training epochs, τ, is specified by the user. Partial training, which was first used in conjunction with an evolutionary algorithm [28], means that the NN is trained for a fixed number of epochs regardless whether it has converged or not.

Step 3) Check the termination criterion of NN training. If it is satisfied, the current NN architecture is the outcome of CAELF for a given data samples. Otherwise, follow the next step. In this work, calculate average training error [29], E_a on the validation samples. In other word, the average training error is considered here as mean squared error (MSE). Thus, the error, E_a, is calculated as,

$$E_a = \frac{1}{2p\tau}\sum_{c-1}^{c}(t_c(p) - o_c(p))^2 \quad (1)$$

Where, $t_c(p)$ and $o_c(p)$, respectively, are the actual and predicted responses of the *c-th* output neuron for the validation pattern *p*. The symbols *P* and *C* represent the total number of validation patterns and of output neurons, respectively.

Step 4) Check the performance criterion of the network training. If the criterion is satisfied then the network is assigned to be trained further and go to Step 2. Otherwise, go to the next step.

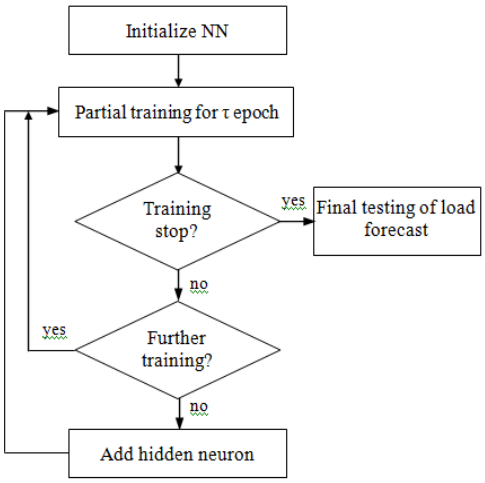

Fig. 1: Flowchart of CAELF. Here, NN and HN refer to neural network and hidden neuron, respectively.

Step 5) Add a hidden neuron to the network and go to Step 2 for following the partial training again.

Step 6) NN is then tested with the unseen testing pattern. Finally, get the electricity load forecasting from the current NN.

CAELF uses only one cost function that is the training error on validation data samples. CAELF finally tries to design a better load forecaster using NN. Details about some basic steps of CAELF are further given in the following sections.

2.1 Performance criterion of NN training

If the average training error on validation samples reduces by a predefined amount ε, after the training epoch τ, it is assumed that the training process is progressing well, thus further training is necessary and go to the Step 2. The reduction of training error can be described as,

$$E_a(t-\tau) - E_a(t) > \varepsilon, \quad t = \tau, 2\tau, 3\tau..... \quad (2)$$

Where, τ and t are positive integer number specified by the user.

2.2 Termination criterion of NN training

Since CAELF adds hidden neurons one by one during the training process of a NN, the training error would reduce as the training process progresses. However, the objective of CAELF is to improve generalization ability of the NN. This means that, the training error may not be a right choice to be used for terminating the training process of the NN. Generally, a separate data samples, called the validation samples, is widely used for termination. It is assumed that the validation error gives an unbiased estimate because the validation data are not used for modifying the weights of the NN.

Table 1: A sample of data showing the Log (Load), HDD, CDD, and dummy variables

OBS	Log(Load)	HDD	CDD	Tue	Wed	--	Sun	Holiday	Easter	Feb	Mar	--	Dec
1/1/93	12.64	11.9	0	0	0	--	0	1	0	0	0	--	0
--	--	--	--	--	--	--	--	--	--	--	--	--	--

--	--	--	--	--	--	--	--	--	--	--	--	--	--
31/12/97	12.93	7.47	0	0	1	--	0	0	0	0	0	--	1

In order to achieve good generalization ability, CAELF uses average training error on validation samples in its termination criterion. It measures validation error after every τ epochs of training, called strips. It terminates training when the average training error increases by a predefined amount (λ) for T successive times, which are measured at the end of each of T successive strips [30]. Since the average training error on validation samples increases not only once but T successive times, it can be assumed that such increases indicate the beginning of the final over fitting not just the intermittent. The termination criterion can be expressed as,

$$E_a(\tau+i)-E_a(\tau)>\lambda, \quad i=1,2,3,.....,T \quad (3)$$

Where, τ and T are the positive integer number specified by the user. Our model, CAELF tests the termination criterion after every τ epochs of training and stops the training when the condition described by the Eq. (3) is satisfied. In this work, the value of T is chosen as 3.

2.3 Hidden neuron addition

CAELF adds a hidden neuron to the existing network architecture according to the Eq. (4). The reason is that, the existing network architecture is not capable to acquire the all information of the data samples; thereby increasing the size of the network is necessary. Then, train the modified architecture for a certain number of τ epochs.

$$E_a(t-\tau)-E_a(t)\le\varepsilon, \quad t=\tau,2\tau,3\tau..... \quad (4)$$

Where, ε is the predefined amount specified by the user.

2.4 Computational Complexity

Computational complexity is a measure by which it can be understood about a model how much complexity is available in the computational process. However, computational complexity theory is a branch of the theory of computation in

theoretical computer science and mathematics that focuses on classifying computational problems according to their inherent difficulty, and relating those classes to each other. A computational problem is understood to be a task that is in principle amenable to being solved by a computer, which is equivalent to stating that the problem may be solved by mechanical application of mathematical steps.-

The analysis of computational complexity helps to understand the actual computational cost of an algorithm. As Kudo and Sklansky [33] showed such an analysis in the form of big-O notation, we are inspired to compute the computational cost of our CAELF. The following few paragraphs present the computational complexity of CAELF to show that the inclusion of different techniques does not increase computational complexity of training NNs.

(i) **Partial Training:**

In our thesis, we use standard back propagation (BP) algorithm [27] for training. Each epochs of BP takes $O(w)$computations for training one example. Here, *W* is the number of weights in the current NN. Thus training all examples in the training set for τ epochs needs $O(\tau \times p_{t\times}w)$ computations, where P_t denotes the number of examples in the training set

(ii) **Termination Criterion:**

The termination criterion employed in CAELF for stopping training of the NN that uses both training and validation errors. Since the training error is computed as a part of the training process, the termination criterion takes computations,

$$O(p_v \times w)$$

Where P_v denotes the number of examples in the validation set. Here $P_v<P_t$,

$$O(p_v \times w) < O(\tau \times p_t \times w)$$

(iii) **Further Training:**

Our CAELF uses Eq. (4) to check whether τ further training for the added HN is necessary. The evaluation of Eq. (4) takes a constant computation $O(1)$, since the error values used in Eq. (4) have already evaluated during training.

(iv) **Adding a Hidden Neuron:**

The computational cost for adding a hidden neuron is $O(N_1 + C)$ for initializing its connection weights, where N_1 is the number of added input features and *C* is the number neurons in the output layer. It is also noted that, $O(N_1 + C) < O(\tau \times p_{t\times}w)$

All the above mentioned computation is done for a partial training consisting of *t* epochs. In general, CAELF needs several, say *M*, such partial trainings. Thus, the total computational cost of CAELF for training a total of T epochs,

$T = \tau \times M$ is $O(N^2 \times p_t) + O(\tau \times M \times p_{t\times} w)$

However, in practice, the first term, i.e., $N^2 \times p_t$ is much less than the second one. Hence the total computational cost CAELF is,

$$O(\tau \times M \times p_{t\times} w),$$

Which is same for training a fixed network architecture using BP [27]. It is clear that the incorporation of several techniques in CAELF does not increase its computational cost.

3. EXPERIMENTAL STUDY

In this section, the performance of CAELF for predicting the electricity load at near future was presented using the daily load data sample. The data used in this study is the daily electricity demand in megawatts/hour in Spain [31], [32]. The CAELF's performance was evaluated in terms of predicted error. Precisely, predicted error refers to the error of existing NN on testing data samples. For more clarification about the performance evaluation of CAELF, this section is organized by the following subsections.

3.1 Description of Data

The data used in the experimental analysis of this paper is the daily electricity demand in megawatts/hour in Spain from January 1, 1993 to June 30, 1998 for a total of 2007 days. To clarify about the data, Table 1 shows a partial sample of the data sheet. Using this data, the architecture of a feed-forward NN is also shown in Fig. 2.

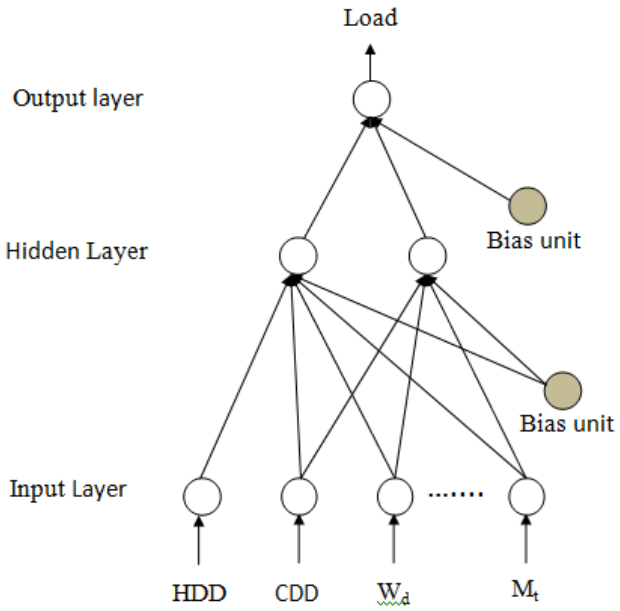

Fig. 2: Model of FFNN for forecasting the electrical load. Here, HDD and CDD refer to the exogenous variables of degree days that are calculated as heating degree days and cooling degree days, in that order. On the other hand, W_d and $M_{t\ are}$ the dummy variables that represent all the weeks and monthly seasonalities, respectively. For more information about these input variables can be found in [21].

3.2 Experimental Setup

The data used for training of the NN model in CAELF was from January 1, 1993 to December 31, 1997 for a total 1826 days, whereas the NN was validated during training using the data from July 1, 1998 to December 31, 1998 in total 184 samples. Precisely, these samples were called “in-sample” data as they used in NN model for training. On the other hand, the data sample period of 120 days from January 1, 1999 to April 30, 1999, that were used to test the forecasting performance by comparing model output (i.e., predicted load) with the actual load. These data samples are called as “out-of-sample” as they were not used during the training of NN.

In all experiments, one bias unit with a fixed input +1 was connected to the hidden layer and output layer. The learning rate and momentum term for training of NN were chosen as 0.05–0.1 and 0.4–0.7, respectively. The initial connection weights for an NN were randomly chosen in the range between -1.0 and 1.0. A sigmoid function was used as an activation function.

3.3 Experimental Results

The performance of CAELF in terms of forecasting the out-of-samples was measured by making a comparison between actual values and model outputs during the same period. Furthermore, we measured the mean absolute percentage error (MAPE) for the best relative accuracy measure among the various forecasting accuracy criteria [11]. However, MAPE was calculated here as,

$$MAPE = \frac{1}{N}\sum_{n=1}^{N}\left|\frac{P_n - \overline{P}_n}{p_n}\right| \times 100 \quad (5)$$

Where, P_n and $\overline{P}_n$ represent the actual and predicted electricity load, respectively and N is the total number of samples available. In this context, Fig. 3, Fig. 5 and Fig. 6 represent the forecasting analysis for the period of 120 days (including errors in percentage), 30 days, and 7 days, respectively

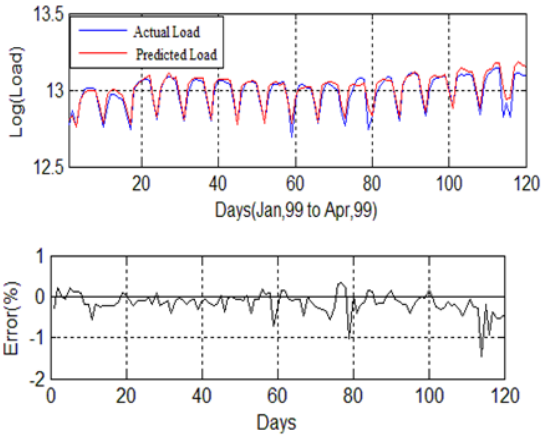

Fig. 3(a): Comparison between the actual load and the predicted load obtained from CAELF used by constructive feed forward neural network and (b): corresponding their errors in percentage.

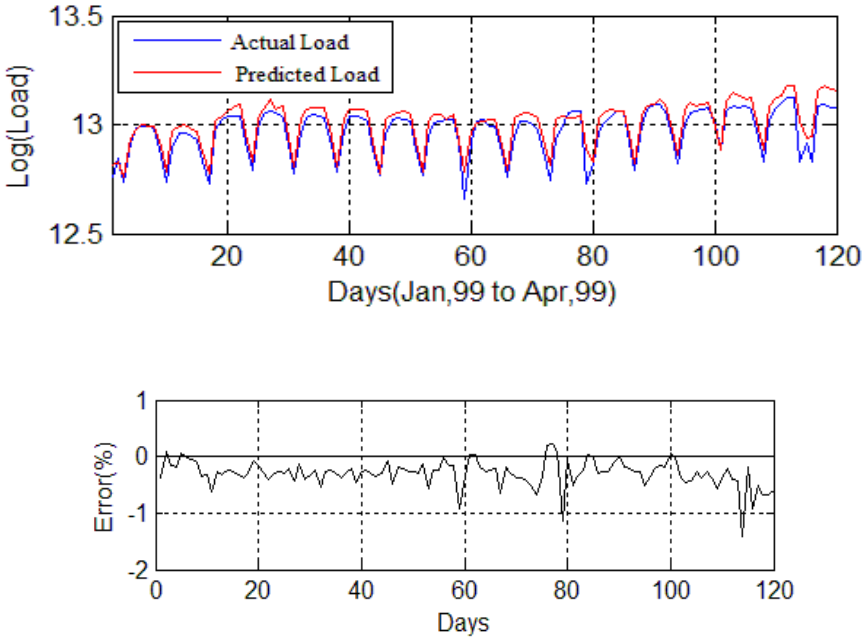

Fig. 4(a): Comparison between the actual load and the predicted load for 120 days obtained from standard model (SELF) used by feed forward neural network and (b): corresponding their errors in percentage.

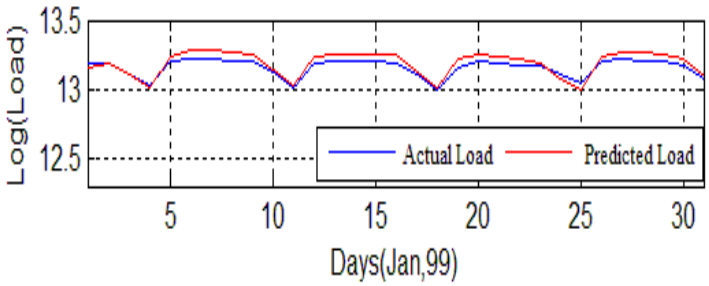

Fig. 5: Comparison between the actual load and the predicted load for 30 days obtained from CAELF used by constructive feed forward neural network.

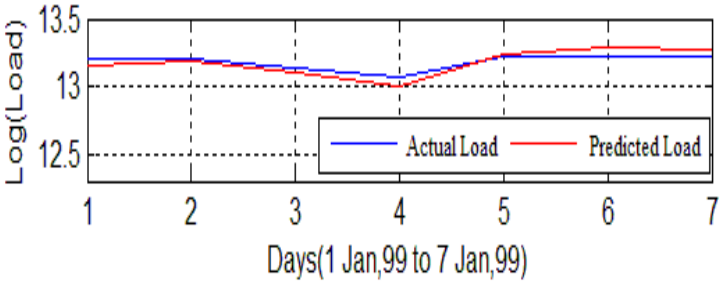

Fig. 6: Comparison between the actual load and the predicted load for 7 days obtained from CAELF used by constructive feed forward neural network.

Particularly, a comparison between actual load and predicted load (i.e., forecasting load) was made in Fig. 3 (a) using CAELF. We calculated MAPE between these two loads and found that it was 0.2132. In addition, more analytical forecasting results of CAELF can be found in Figs. 5 and 6, where the forecasting of electricity load was done in between 30 days and 7 days, respectively.

In a closed observation among these figures, it has been found that, forecasting of electricity load using CAELF is satisfactory as the predicted load curve is very much closed to overlap the actual load curve.

In case of holydays load demand, predicting such demand perfectly is really a difficult task for any kind of model. The reason is that, the consumers enjoy their time in different places in various ways,

that's why, the fluctuation of electricity load is nonlinear. In order to observe such issue, CAELF conducted one experiment for 30 days, where the forecasting results especially for the holydays are pointed out between the actual load curve and predicted load curve in Fig. 7. It has been seen that, the load variations are satisfactory except some cases. Furthermore, Table 3 shows that the difference between predicted load and actual load is very low.

3.4 Comparison with other works

The obtained forecasting result of CAELF on Spanish daily electricity load data has been compared with the results of three electricity load forecasting models, such as, (i) standard electricity load forecasting (SELF), (ii) NNELF-1 [16], and (iii) NARx-2 [16].

Table 2: Comparisons with other models for the ELF problem in terms of the next 120 Days Here, the comparisons are made according to MAPE

Models	Mean	SD	Max	Min
NNELF-1[16]	0.6426	0.0551	0.7686	0.4946
NARx-2 [16]	0.3852	0.1132	0.6455	0.2367
SELF	0.3151	0.0009	0.3168	0.3142
CAELF	0.2132	0.0008	0.2136	0.2110

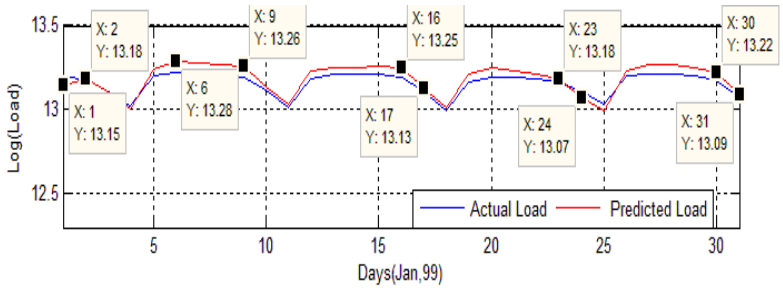

Fig. 7: Comparison between the actual load and the predicted load for Holidays of January 99 obtained from CAELF used by constructive feed-forward neural network.

The first two models used standard feed-forward NN for electricity load forecasting, where a fixed number of hidden neuron in the hidden layer of NN and a fixed number of iteration for the NN training have been considered. The third one is the nonlinear autoregressive model that composed by two parts: first, the true available output is fed as an input to train the NN; second, the resulting network has a purely feed forward architecture and BP algorithm is used for training. We used one parameter for comparisons here, that is to say, MAPE.

In SELF, the whole setup of CAELF was used except the constructive approach and partial training. In this case, 5 hidden neurons were considered in the hidden layer of NN and 200 iterations for the NN training. For comparisons, we run SELF from 10 times and averaged the forecasting results. On the other hand, the model NNELF-1 [16] and NARx-2 [16] used 10 hidden neurons in the hidden layer and the forecasting results were averaged by 20 individual runs. Table 2 shows the comparison result among these four models including CAELF. It has been found that, the value of MAPE is the most reduced one for CAELF among the other models. On the other hand, the minimum value of SD signifies the robustness of CAELF.

3.5 Analysis

A rigorous analysis about the experimental performance of CAELF is done in different aspects in order to measure the generalization ability. In this regard, three synthetic time series data samples were used.

3.5.1 Mackey-Glass Time Series

The Mackey-Glass series, based on the Mackey-Glass differential equation is widely regarded as a benchmark for comparing the generalization ability of different methods.

Table 3: Comparisons between the actual and predicted load for Holidays of January 99

Holidays	Predicted	Actual	Differences
Jan'1st	13.1888	13.1482	0.0406
2nd	13.187	13.1809	0.0061
6th	13.2189	13.2833	-0.0644

9th	13.1974	13.2569	-0.0595
15th	13.2077	13.2582	-0.0505
16th	13.1921	13.2542	-0.0621
23rd	13.1645	13.1842	-0.0197
24th	13.1078	13.0741	0.0337
30th	13.1744	13.2209	-0.0465
31st	13.0745	13.0928	-0.0183

This series is a chaotic time series generated from the following time-delay ordinary differential equation: Mackey-Glass time series refers to the following differential equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} - bx(t) \qquad (7)$$

delayed differential equation using the 4th order Runge-Kutta. The generated data sample are presented in the following Fig. 8, where the value of $\tau = 17$, $a = 0.2$, $b = 0.1$.

3.5.1.1 Forecasting Results

Using the Mackey-glass time series model, we generated some data samples that were used as training and testing data samples. It can be numerically solved using, for example, the 4thorder Runge-Kutta method, at discrete, equally spaced time steps:

$$x(t+\Delta t) = mackeyglass\_rk4(x(t), x(t-\tau), \Delta t, a, b)$$

Where, the function *mackeyglass_rk4* numerically solves the Mackey-Glass

After training CAELF with the training data samples, the testing samples were applied to the trained NN model in order to find the forecasting result. After that, we found a promising result using Mackey-glass time series data samples, which are exhibited in Fig. 9 and Table 4.

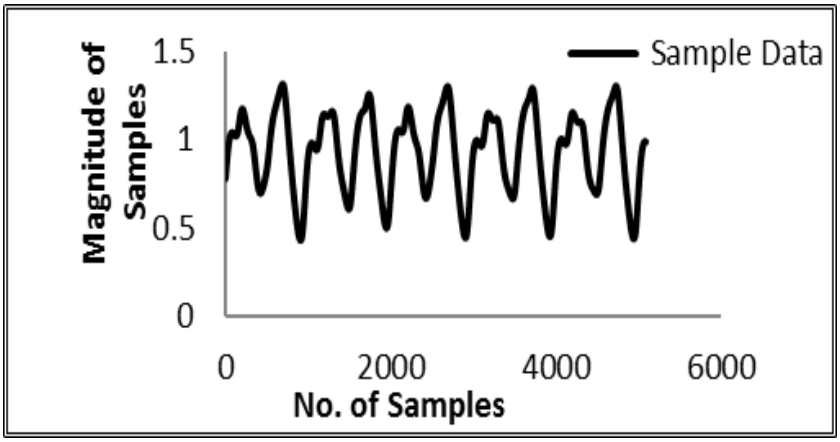

Fig. 8: Sample data of the Mackey-glass time series

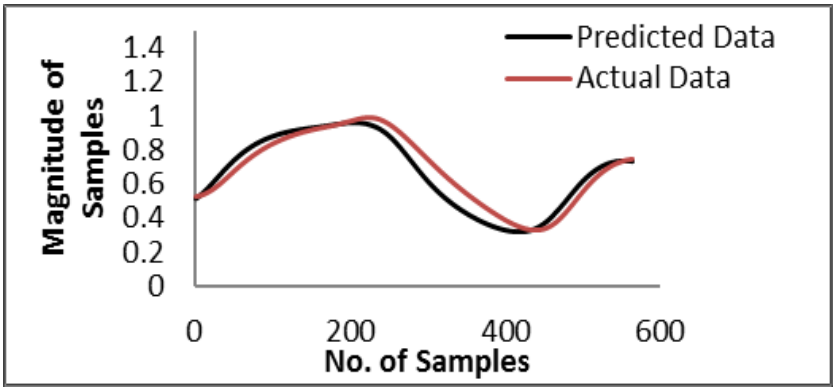

Fig. 9: Comparison between the actual data and the predicted data for Mackey-glass data samples obtained from CAELF

In accordance with Fig. 9, it has been found that, the two curves, that is to say, predicted and actual data curves are closely over lapped each other. The reason is that, the MAPE in this case is very low, i.e., 0.10823 as well as the value of SD is quite low, which are presented in Table 4.

Table 4: The value of MAPE of CAELF on Mackey-glass data, here SD refers to standard deviation

	Mean	**SD**	**Max^m**	**Min^m**
MAPE	0.108234	0.014332	0.127487	0.087631

Thus, we can say that, our CAELF model is robust and well-performed in predicting the value of Mackey-glass time series problem.

3.5.2 Lorenz Time Series

The Lorenz system is a system of ordinary differential equations (i.e., the Lorenz equations) first studied by Edward Lorenz. It is notable for having chaotic solutions for certain parameter values and initial conditions. In particular, the Lorenz attractor is a set of chaotic solutions of the Lorenz system which, when plotted, resemble a butterfly or figure eight.

In 1963, Edward Lorenz developed a simplified mathematical model for atmospheric convection. The model is a system of three ordinary differential equations now known as the Lorenz equations, which are:

$$(i)\ \frac{dx}{dt} = \sigma(y - x),\ (ii)\frac{dy}{dt} = x(\rho - z) - y,$$
$$(iii)\frac{dz}{dt} = xy - \beta z.$$

Here, *x*, *y* and *z* make up the system state, *t* is time and σ, ρ, β, are the system parameters. From a technical point of view, the Lorenz system is nonlinear, three-dimensional and deterministic.

3.5.2.1 Forecasting Results

Using the Lorenz time series model, we generated some data samples that were used as training and testing data samples. After training CAELF with the training data samples, the testing samples were applied to the trained NN model in order to find the forecasting result. After that, we found a promising result using Lorenz time series data samples, which are exhibited in Fig. 10 and Table 5.

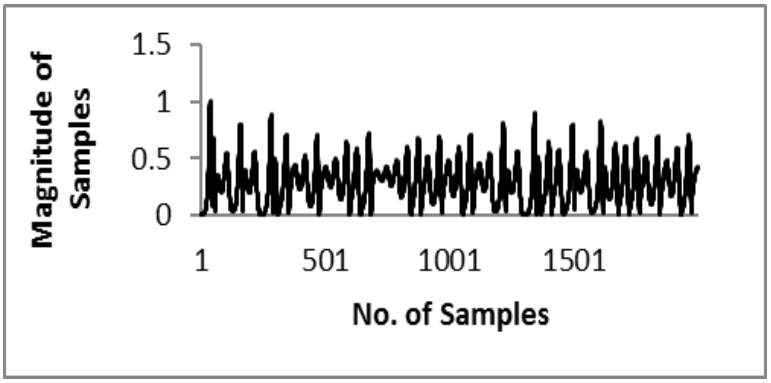

Fig. 10: Sample data of the Lorentz time series

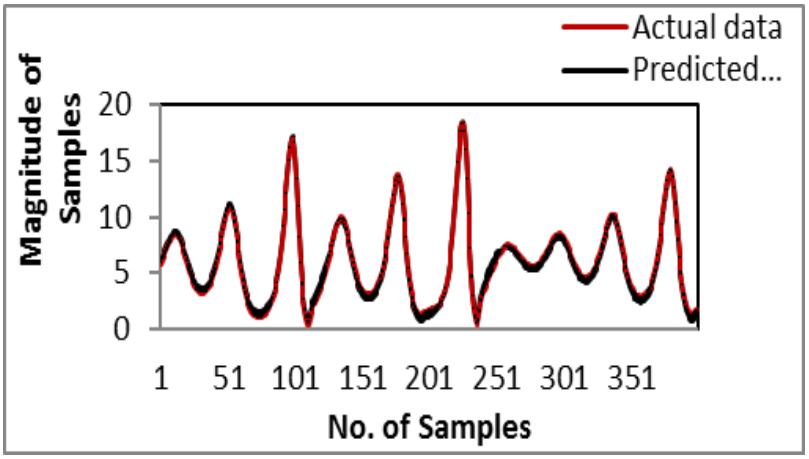

Fig. 11: Comparison between the actual data and the predicted data for Lorentz data obtained from CAELF

Table 5: The value of MAPE of CAELF on Lorentz data

	Mean	**SD**	**Max^m**	**Min^m**
MAPE	0.916056	0.015022	0.93433	0.89491

In accordance with Fig. 11, it has been found that, the two curves, that is to say, predicted and actual data curves are closely over lapped each other. The reason is that, the MAPE in this case is very low, i.e., 0.916056 as well as the value of SD is quite low, which are presented in Table 5. Thus, we can say that, our CAELF model is robust and well-performed in predicting the value of Lorenz time series problem.

CONCLUSION

In this paper, a new short-term electricity load forecasting model has been proposed, CAELF that utilizes the constructive approach in feed-forward NN training. Thus, the FFNN automatically determines the size of the hidden layer during training on basis of the ELF problem domain.

Experimental results presented in Fig. 3, Fig. 5 and Fig. 6 show that, CAELF performs well in ELF problem in respect to the short-term basis (e.g., 120 days, 30 days, and 7 days). The reason is that, the forecasting curve between predicted load and actual load (Fig.3 (a)) is closed to similar. Furthermore, most of the points of errors in this error curve showed in Fig. 4 (b) are closed to zero. In comparison to the other models, Fig. 3 and Fig. 4 show that, the result of our model is much better than that of SELF. In addition to the concept of

MAPE, Table 2 confirms that, our model's performance is better than other models, such as, SELF, NNELF-1 [16], and NARx-2 [16].

Furthermore, in CAELF, the forecasting results are not so satisfactory in the last portion of out-of-sample showed in Fig. 3 (a). The possible reason might be the nonlinearity of the load variations in the data samples. In order to reduce such limitations, incorporating more heuristic techniques in CAELF is left for further works.

REFERENCES

1. H.S.Hippert, C.E.edreira, and R.C.Souza, Neural Networks for Short-Term Load Forecasting: A Review and Evaluation, IEEE Transactions on Power Systems, 16(1) (2001) 44-55.
2. Bunn DW, Forecasting loads and prices in competitive power markets, Proc IEEE 88 (2000) 163–169.
3. M. Sheikhan, N. Mohammadi, Neural-based electricity load forecasting using hybrid of GA and ACO for feature selection, Neural Computation & Application, 2011
4. Hippert H S, Pedreira C E, and Zareipour R C S, Neural Networks for Short-Term Load Forecasting: A Review and Evaluation, IEEE Transactions on Power Systems, 16 (2001) 44-55.
5. Pai PF, Hong WC, Support vector machines with simulated annealing algorithms in electricity load forecasting. Energy Conversion and Management, 46 (2005) 2669–2688.
6. Dehdashti AS, Tudor JR, Smith MC, Forecasting of hourly load by pattern recognition a deterministic approach, IEEE Trans on Power Apparatus and Systems, (1982) 3290-3294. [7] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.
7. Yang HT, Huang CM, A new short-term load forecasting approach using self organizing fuzzy ARMAX models, IEEE Transaction on Power Systems, 13 (1998) 217–25.
8. Mamlook R, Badran O, Abdulhadi E, A fuzzy inference model for short-term load forecasting, Energy Policy 37 (2009) 1239–48.

9. Ying LC, Pan MC, Using adaptive network based fuzzy inference system to forecast regional electricity loads, Energy Conversion and Management, 49 (2008) 205–211.
10. P.F. Paia, W.C. Hongb, Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms. Electric Power System Research, 74 (2005) 417–425.
11. M. Sheikhanand N. Mohammadi, Neural-based electricity load forecasting using hybrid of GA and ACO for feature selection, Neural Computation and Application, 2011. DOI 10.1007/s00521-011-0599-1.
12. Methaprayoon K, Lee WJ, Rasmiddatta S, Liao JR, Ross RJ, Multistage artificial neural network short-term load forecasting engine with front-end weather forecast, IEEE Transactions on Industry Applications, 43 (2007) 1410-1416.
13. Deihimi A, Showkati H, Application of echo state networks in short-term electric load forecasting, Energy, 39 (2012) 327-340.
14. Xia C, Wang J, McMenemy K, Short, medium and long term load forecasting model and virtual load forecaster based on radial basis function neural networks, International Journal of Electricity Power & Energy Systems, 32 (2010) 743-750.
15. Vermaak J, Botha EC, Recurrent neural networks for short-term load forecasting, IEEE Transactions on Power Systems, 13 (1998) 126-132.
16. Elias R S, Fang L, Wahab M I M, Electricity load forecasting based on weather variables and seasonalities: A neural network approach, 8th International conference on service systems and service management, Canada, 2011.
17. A.Malki, N.B.Karayiannis, M. Balasubramanian, Short-term electric power load forecasting using feedforward neural network, Expert systems, 21 (2004) 157-167.
18. D.O.Arroyo, M.K.Skov, Q.Huynh, Accurate electricity load forecasting with artificial neural networks, International conference on computational intelligence for modeling, control, and automotion-

International conference on intelligenct agents, web technologies, and internet commerce (CIMCA-IAWTIC'05), 2005.

19. A.G.Baklrtzls, V.Petrldls, S.J.Klartzls, M.C.Alexladls, A neural network short term load forecasting model for the greek power system, IEEE Transactions on Power Systems, 11 (1996) 858-863.
20. D.Srinivasan, A.C.Liew, and C.S.Chang, A neural network short-term load forecaster", Electric Power Systems Research, 28 (1994) 227-234.
21. C.C.Hsu, C.Y.Chen, Regional load forecasting in Taiwan-applications of artificial neural networks, Energy Conversion and Management 44 (2003)1941-1949.
22. G. Zhang, B. E. Patuwo, M. Y. Hu, Forecasting with artificial neural networks: The state of the art, International Journal of Forecasting 14 (1998) 35-62.
23. Y. Chen, P.B. Luh, C. Guan, Y. Zhao, L.D. Michel, M.A. Coolbeth, et al., Short-term load forecasting: similar day-based wavelet neural networks, IEEE Transactions on Power Systems 25 (2010) 322-330.
24. K.R.Islam, M.M.Kabir, K. Murase, Short-term electricity load forecasting using constructive feed-forward neural network, 2nd International Conference on Machine Learning and Computer Science (IMLCS'2013), pp. 40-44, August 25-26, Malyasia, 2013.
25. M.M.Islam, K. Murase, A new algorithm to design compact two hidden-layer artificial neural networks, Neural Network, 14 (1001) 1265-1278.
26. T.Y.Kwok, D.Y.Yeung, Constructive algorithms for structure learning in feed-forward neural networks for regression problems, IEEE Transactions on Neural Networks 8 (1997) 630-645.
27. D.E. Rumelhart, J. McClelland, Parallel Distributed Processing, MIT Press, 1986.
28. X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, IEEE Transactions on Neural Networks, 8 (1997) 694–713.
29. M.M.Kabir, M.M.Islam, K.Murase, A new wrapper feature selection approach using neural network, Neurocomputing, 73 (2010) 3273-3283.

30. Neural network benchmark problems and benchmarking rules, Technical Report 21/94, Faculty of Informatics, University of Karlsruhe, 1994.
31. Elias R S, Fang L, Wahab M I M, Electricity load forecasting based on weather variables and seasonalities: A neural network approach, 8th International conference on service systems and service management, Canada, 2011.
32. Pardo A, Meneu V, Valor E, Temperature and seasonality influences on Spanish electricity load, Energy Economics, 24(2002) 50-70.
33. M. Kudo and J. Sklansky, Comparison of algorithms that select features for pattern classifiers. Pattern Recognition, 33 (2000) 25–41.