Long-Term Load Forecasting Approach Using Dynamic Feed-Forward Back-Propagation Artificial Neural Network

Amin Masoumi, Farkhondeh Jabari, Saeid Ghassem Zadeh and Behnam Mohammadi-Ivatloo

Abstract In recent years, due to increasing rate of electricity demand and power system restructuring with a limited investment in transmission expansion, large power systems may closely be operated at their stability margins. Meanwhile, uncertain and intermittent nature of electricity demand with traditional load forecasting error seriously effects on operation and planning of bulk power grids. Hence, this chapter aims to present a novel approach based on dynamic feed-forward back-propagation artificial neural network (FBP-ANN) for long-term forecasting of total electricity demand. A feed-forward back-propagation time series neural network consists of an input layer, hidden layers, and an output layer and is trained in three steps: (a) Forward the input load data, (b) Compute and propagate the error backward, (c) Update the weights. First, all examples of the training set are entered into the input nodes. The activation values of the input nodes are weighted and accumulated at each node in the hidden layer and transformed by an activation function into the node's activation value. Hence, it becomes an input into the nodes in the next layer, until eventually the output activation values are found. The training algorithm is used to find the weights that minimize mean squared error. The main characteristics of FBP-TSNN are the self-learning and self-organizing. The proposed algorithm is implemented on Canada's power network to prove its accuracy along with effectiveness, and then compared with real historical data.

Keywords Feed-forward back-propagation artificial neural network (FBP-ANN) · Long-term forecasting · Electricity demand

A. Masoumi (✉) · F. Jabari · S. Ghassem Zadeh · B. Mohammadi-Ivatloo
Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran
e-mail: a.masoumi95@ms.tabrizu.ac.ir

F. Jabari
e-mail: f.jabari@tabrizu.ac.ir

S. Ghassem Zadeh
e-mail: g_zadeh@tabrizu.ac.ir

B. Mohammadi-Ivatloo
e-mail: bmohammadi@tabrizu.ac.ir

M. Pesaran Hajiabbas and B. Mohammadi-Ivatloo (eds.),
Optimization of Power System Problems, Studies in Systems, Decision and Control 262,
https://doi.org/10.1007/978-3-030-34050-6_11

1 Introduction

Nowadays, interconnected power systems have been developed from reliability and stability aspects. Studies indicate that increasing rate of population all around the world leads to a growth in electricity consumption [1]. Meanwhile, uncertain nature of electrical demand and renewable energy sources such as solar and wind adds some limitations to dynamic stability of large electricity grids [2]. Hence, forecasting of electrical demand becomes more and more critical for assisting power system operators in electricity grid management, whether for short-term analysis or long-term applications such as economic emission dispatch, unit commitment, optimal scheduling, etc. [3–5].

Recently, many scholars have proposed different short and long-term load forecasting algorithms. In this context, authors of [6] have simulated a specific aggregated state prediction for electrical consumption of interconnected power networks with 1% error in 700 h. In [7], combination of genetic algorithm and neural network is an illustrative example with accuracy of 98.95% for expanding a feed backward neural network for forecasting of heterogeneous demand time series in very-short and short time intervals. Application of support vector machine (SVM) has been presented in [8] for one hour ahead demand forecasting. In addition, this two-phase technique consisting of artificial neural network (ANN) and SVM has demonstrated the resolution of speed and accuracy through precise experiment on real historical data of 4th July 2012. Son et al. [9] has evaluated the application of support vector regression (SVR), fuzzy logic, and particle swarm optimization (PSO) with mean demand scaling of 149.28754 kW for short-term electrical demand forecasting. Guo et al. [10] has introduced a self-learning algorithm for load forecasting process which benefits from economic factors. This approach inserts some economic elements in searching process to reduce computational error. According to overall implementation of automation system in residential consumption as demand response strategies, it is found that modified algorithms, which made up SVR, can fix the intermittent nature of internal loads (i.e. cooling, heating, and ventilation). Thereby, poly-phase prediction can practically actualize the demand response strategies in such programming. In addition, Le Cam et al. [11] has aimed to forecast total electricity cost of automation system in a benchmark building by providing a poly-stage prediction model that 14.2–22.5% optimum absolute error has been observed. Li et al. [12] has combined a wavelet decomposition technique with ANN to diminish the negative impacts of volatile load data. Plus, in the noted study, it can be given as advanced intelligent algorithm with 2.4% mean absolute percentage error (MAPE). Effectiveness of ANNs in Poland's natural gas consumption forecasting has been described in [13]. This approach has been investigated on historical time series of Szczecin city with characteristics of 22 input, 36 hidden layer and 1 output and 8% MAPE. Authors of [14] have used ANNs for hour ahead prediction of universal solar radiation with 7.86% RMSE through filtering the low frequency of input data set. Reference [15] runs two different learning rules on ANNs. Therefore, it is observed that integration of back propagation (BP) and extreme learning method (ELM) for decoding the fluctuation of wind speed prediction leads to 1.33% and 1.1965% RMSE, respectively. In a similar standpoint, conducting PSO and genetic algorithm for optimal selecting of weight vectors of ANN in solar irradiance estimation system can be found in [16]. It indicates that the combination of BP-ANN and PSO technique has demonstrated correlated results as 0.78 RMSE and 0.685 mean absolute error (MAE). Authors of [17] have aimed to apply a proper orthogonal decomposition (POD) to ANNs for wind and demand forecasting of high altitude towers. It is found that such complex algorithm can reaches RMSE and mean error of 4% and 0.98%, respectively. As mentioned, ANNs support both regression based and computational methods under various prediction scales. Ramasamy et al. [18] has formed a unique wind power forecasting with respect to ANNs through speed estimation experiment in western Himalayan. To prove its robustness, output series have compared with real historical set considering some environmental and geographical factors such as temperature, air pressure, latitude, and longitude. Resiliency of this method against the time variant nature of ANN's input parameters has been revealed by 6.489% as MAPE. Yadav and Chandel [19] have identified relevant input variables for predicting of 1-min time-step photovoltaic module power using ANNs and multiple linear regression models with 2.15–2.55% MAPE. da Silva et al. [20] has reached to important point that using Bayesian Regularization (BR) and Levenberg Marquardt (LM) as training of ANNs has real time result in comparison with others for solar power estimation in a way that MAPE and RMSE are equal to 0.02%, 0.11% (BR), and 0.31%, 0.74% (LM), respectively.

This chapter aims to present a dynamic feed-forward back-propagation ANNs based method for long-term forecasting of electrical demand. In addition, the highly features of compatibility and accuracy of the proposed algorithm is revealed using a comparison between the forecasted and the actual electrical demands of Canada's, Ontario independent electricity operator system (IESO), low voltage grid. The remainder of this chapter can be organized as follows: Sect. 2 presents the problem formulation. Simulation result and discussions are provided in Sect. 3. Finally, concluding remarks appear in Sect. 4.

2 Problem Formulation

2.1 *Artificial Neural Network (ANN)*

Artificial neural network (ANN) is a powerful and extensive tool for engineering applications such as fitting, pattern recognition, clustering, and prediction. The performance of this algorithm is that, by applying weigh matrix and bias vectors to the input vectors and mathematical operations, we can reach the desired output which is demand consumption as in Fig. 1. The aim is to learn our ANN for obtaining the desired output, and also updating the error vectors in any steps. Then, after several iterations, we can regain the optimal weight values for the input vectors. The type of learning in our ANN is supervised learning rule which it benefits from three dynamic training techniques. Hence, the learning rule will be discussed in the next section. During learning, we update the weight matrix to determine the best error as well as observing a tolerable output dependency. Then, ANN will initiate with parameter load and time series for the network as an input, and then variables would have multiplied by weight matrix. Finally, they would have added by bias vectors. The next step begins in such order that, a specific mathematical function is presented to start the calculation as in Fig. 2.

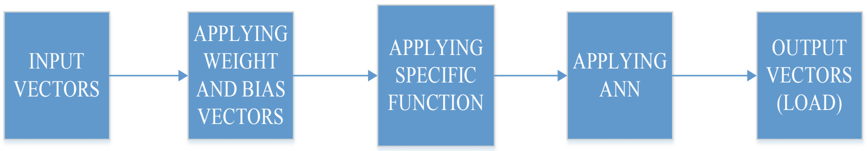


Fig. 1 The block diagram of proposed algorithm

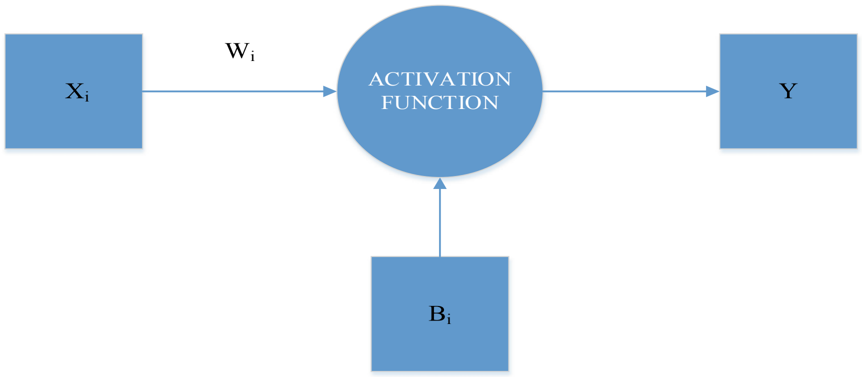


Fig. 2 The operation of the proposed algorithm

Where, x_i, w_i, B_i are the input vectors, weight matrixes and bias vectors respectively and Y is the output of neural network as is depicted in Eq. (1).

$$y_i = F(x_i \times w_i + b_i) \tag{1}$$

In order to satisfy the convergence condition, the algorithm is constructed based on supervised learning rule. In supervised learning at any moment in time K input $x(k)$ is applied to the network. Network desired response $\hat{Y}$(k) is given and Couples ($x(k)$, $\hat{Y}$(k)) belong to a given set of learning that are pre-selected. The $x(i)$ and $\hat{Y}(i)$, $i = 1 \ldots$, N (N is number of neurons) are used in supervised learning rule when $\hat{Y}$(k) $= \hat{Y}(i)$, $x(k) = x(i)$. Our desired network is Multi-Layer Perceptron (MLP) which has a group of vectors like input, output (validation), and network response (Y(k)). MLP is a computational unit in the ANN architecture that is consisted of input layer, hidden layer, and one output layer. After the combination of this input, calculation process will begun as in Fig. 3.

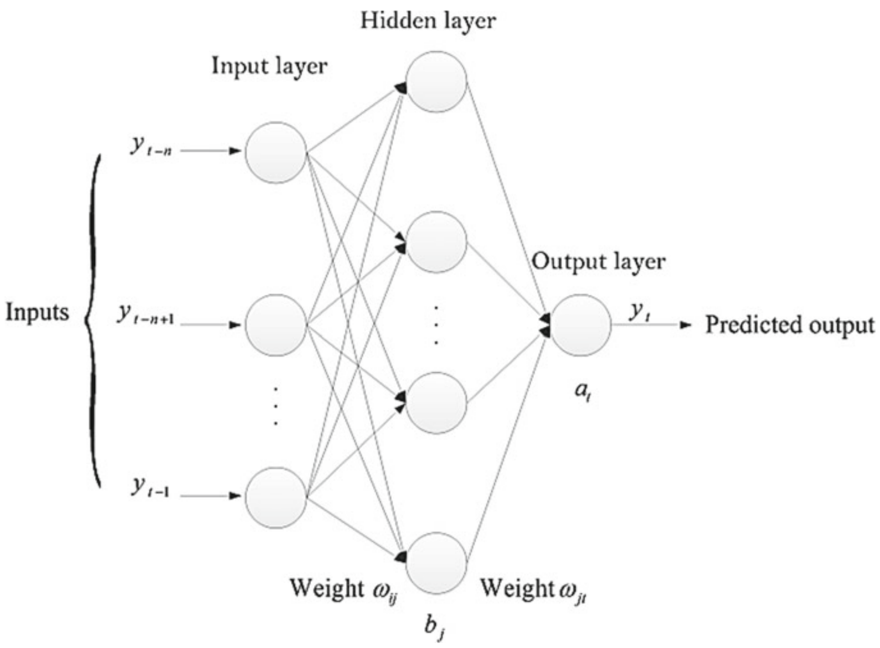


Fig. 3 Different layers of MLP

2.2 Dynamic Artificial Neural Network (DANN)

Dynamic artificial neural network is a computational operating system in which the continuous integration between its elements and the training processes improves the power of prediction. As it is depicted in Fig. 3, the conventional neural network has been made up of one input layer, one hidden layer and, one output layer. In addition, the relation among the parameters, layers, and also the training steps, leads to the prediction purpose. According to the nonlinear feature of input data set, using effective learning rules which they will define in the following section, can help the propagation procedure to be more dependable in comparison with individual learning network. Plus, the volume of calculation is a very critical point that must be considered if the number of sample data is notable for the convergence application. From this view point, overall conformity of forecasting algorithm will lead to the better understanding of estimated output. As it is clear in Eq. (1), the weight matrix and bias vectors are the stimulation parameters of ANN that are needed to generalize the diversity of input vectors. They simply interconnect the intermittent based inputs which are electric demand in this case, into the correcting steps of predicting. The activation function F is the operator of the net that is accommodated with both correcting and training processes. In another word, the F will demonstrate the predicting based on the termination of gradient process. The connectors, which are neurons, transfer the optimized values of weight and bias to the output layer in specific order. Moreover, dynamic neural network is defined as the combination of three regression based learning method which are Levenberg Marquardt (LM), Bayesian Regulation (BR), and Scaled Conjugated Gradient (SCG). First of all, the learning system operator will initiate with LM to train the portion of input data set (70%) and will allocate primary weight and bias vectors. After the generalization, the output trained is employed to the second propagation network (BR) to be normalize and filter the white noises of set with respect to the error performance (MSE) of the early neural network with 70% as training ratio. The next view is to conjugate the performance of two aforementioned learnings techniques to scale the searching process of optimal weight and bias vectors selecting in the hypothesis space. In another word, scaled conjugated will find the specific vectors for minimizing the error (MSE). Plus, this algorithm uses desired values of weight matrix at each stage to change them so that, the downward slope of the error curve is going to be descent. The flowchart of proposed algorithm for load forecasting is depicted as in Fig. 4.

2.3 Back Propagation Technique (BP)

Back propagation is a learning and adjusting method which conveys several partial derivatives from the basic parameter of neural network. In this method, we try to minimize the objective function and obtain mean square error (MSE) between the output of net and the desired output of electrical demand using dynamic algorithm. It

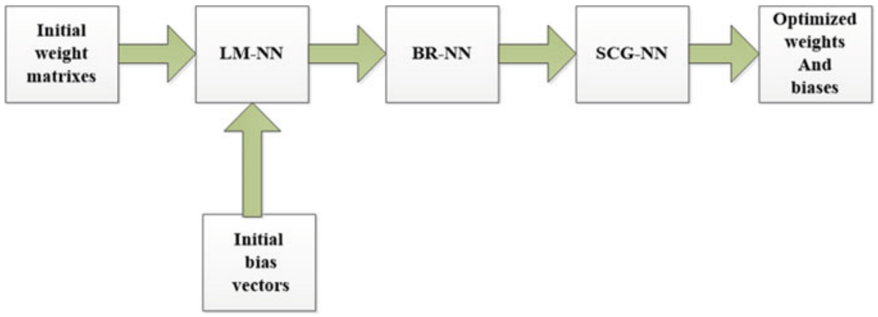


Fig. 4 The proposed algorithm

is observed that the hypothesis searching space is a large space that defines all possible values of the weights. The ANN tries to optimize the error to achieve reasonable states. However, there is no guarantee that it will reach to the absolute minimum. Therefore, the training algorithm (DANN) is used to find the weights that minimize mean squared error (MSE). Broadly, the mechanism of BP is based on the operating of tansigmoid as a sigmoid function for the hidden layer as well as pure linear function for the output layer. In this context, Eqs. (1) and (2) are the illustrative of hidden and output structure, respectively. As the same manner, the predicted output of the net can be achieved through the Eqs. (3) and (4) and algorithms were trained in three steps:

1. Forward the input data
2. Compute and propagate error backward
3. Update the weights

$$x_j = \sum_{i=t-1}^{t-n} \sum_{j=1}^{h} \omega_{ij} \times y_i + b_j \tag{2}$$

$$y_j = \frac{1}{1 + \exp(-x_j)} \quad j = 1, 2, \ldots, h \tag{3}$$

$$x_t = \sum_{j=1}^{h} \omega_{jt} \times y_j + a_t \quad t = 1, \ldots, T \tag{4}$$

$$y_t = x_t \quad t = 1, \ldots, T \tag{5}$$

where,

x_j, y_j	Input and output of the *j*th node of the hidden layer
ω_{ij}	Weight between *i*th input layer neuron and *j*th hidden layer neuron
b_j, a_j	Bias of the input and the hidden layers which are within the range of [−1, 1]
n, h, T	Number of input, hidden, and output layer nodes
x_t, y_t	Input and output values of the output layer at time horizon *t*
ω_{jt}	Connection weights of the *j*th hidden and output layers.

The mean square error of per cycle or epoch (Total square error for all learning models) and the norm of the gradient error is less than a predetermined value. The BP's view rest on the assumption of error gradient technique in the weight space. Hence, there is possibility to catch in Local minimum. To avoid such obstacle, we can use stochastic gradient with different values for weights. Considering the aforementioned concept, the weight adjustment rule in *i*th iteration depends on the size of the weight in the previous iteration as in Eq. (6).

$$\text{MSE} = \frac{1}{2}\sum_{t=1}^{T} (\hat{y}_t - y_t)^2 \tag{6}$$

where, $\hat{y}_t$ and y_t are the predicted results and expected output of the neural network, respectively. As a result, trapping in local minimum and placing on flat surfaces can be avoided, however, the search speed increases with gradual increase of step modification. It is observed from BP's properties that it can show undetected features of input data in hidden layer of network. Hence, the adjusting procedure is initiated by Eqs. (7) and (8) to propagate the weights of hidden and input neurons as follows:

$$\Delta\omega_{jt} \propto -\frac{\partial \text{MSE}}{\partial \omega_{jt}} \tag{7}$$

$$\begin{aligned}\Delta\omega_{jt} &= -\eta\left(\frac{\partial \text{MSE}}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial x_t}\right)\left(\frac{\partial x_t}{\partial \omega_{jt}}\right) = \eta(\hat{y}_t - y_t)\left(\frac{\partial\left((1+\exp(-x_t))^{-1}\right)}{\partial x_t}\right)y_j \\ &= \eta(\hat{y}_t - y_t)y_t(1-y_t)y_j \\ &\text{for } j = 1, \ldots, h, t = 1, \ldots, T\end{aligned} \tag{8}$$

In which,

$\Delta\omega_{jt}$	Weights of hidden neurons
η	Learning rate
$\frac{\partial \text{MSE}}{\partial y_t}$	Derivative of the error with respect to the activation
$\frac{\partial y_t}{\partial x_t}$	Derivative of the activation with respect to the net input

$\frac{\partial x_t}{\partial \omega_{jt}}$ Derivative of the net input with respect to a weight.

According to the association of error (MSE) in each step of iteration, if the algorithm continues until the error is less than a certain amount, BP will terminate which can lead us to over-fitting. Over-fitting is caused by the weight adjusting that may've not a conformity with overall distribution data. By increasing the number of iteration, the complexity of the hypothesis space learned by the algorithm becomes more and more comprehendible until it can able to evaluate noise and rare example of a network in the training set properly. The solution is to import approved collection called validation set to stop learning when the error is small enough in this series and also to network for simpler hypothesis spaces. Then, the amount of weight in each iteration can be reduced. After determining the optimized values of weights, the error in all nodes can participate as follows:

$$\Delta\omega_{ij} \propto -\frac{\partial \mathrm{MSE}}{\partial \omega_{ij}} \tag{9}$$

Consequently,

$$\begin{aligned}\Delta\omega_{ij} &= -\sum_{t=1}^{T}\left[\left(\frac{\partial \mathrm{MSE}}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial x_t}\right)\left(\frac{\partial x_t}{\partial y_j}\right)\right]\left(\frac{\partial y_j}{\partial x_j}\right)\left(\frac{\partial x_j}{\partial \omega_{ij}}\right)\\ &= \eta\sum_{t=1}^{T}\left[(\hat{y}_t - y_t)y_t(1-y_t)\omega_{it}\right]y_j(1-y_j)y_i\\ i &= t-n,\ldots,t-1 \quad j = 1,\ldots,h\end{aligned} \tag{10}$$

2.4 Levenberg Marquardt Algorithm (LM)

The LM is a computational approach for data mining problems of NN which include uncertain parameter structure. In this premise, LM categorize the input data set by learning the NN algorithm to adapt with the previous state of parameter through the error expected (MSE). This method is basically drowned out by the popular Gaussi-Newton technique [21] in non-singularity functions (tansig) as in Eqs. (11) and (12):

$$x^{k+1} = x^k + \Delta x \quad k = 1,\ldots,N \tag{11}$$

In which, x^{k+1}, x^k and Δx represent the current state, historical recent state, and the deviation with time step of time series, respectively. The deviation can be modeled in the LM concept in which the Jacobians of errors train each node of neural network as follows:

$$\Delta x = \left[J^T J + \eta I\right]^{-1} J^T MSE \tag{12}$$

where, the J, η and MSE represent the first derivative of errors with respect to the back propagation process, learning rate (70%) and the mean square error, respectively. The merit of LM is the speed of convergence that aims to escape from the local minima for the sake of prediction [22]. According to the abovementioned equation, LM method has been conducted a correcting system on error (MSE) instead of using Hessian matrix. It is noted that the main point in the weight adjusting of NN is the propagation of neurons of hidden layer which over fitting may have been occurred, if the covariance of data set is contaminated with heterogeneous pattern [23]. Hence, the propagation search is described as Eqs. (13) and (14):

$$\omega_{ij}^{k+1} = \omega_{ij}^{k} + \Delta\omega_{ij} \quad k = 1, \ldots, N \tag{13}$$

$$\Delta\omega_{ij} = \left[J^T J + \eta I\right]^{-1} J^T MSE \tag{14}$$

2.5 *Bayesian Regularization (BR)*

After considering the standardizing steps of LM in the propagation process, Bayesian Regularization (BR) is applied for the over-fitting problem of weight allocation in NN [24, 25]. Meanwhile, BR detects the unregulated weights considering their error (MSE) as well as accelerates the search speed for classifying the weights by the help of reducing their possibility from the state space. In another word, BR filters out the unbiased weights which are selected randomly. Plus, by determining such weight that are the white noises of NN, the optimum values can be more achievable than its former state. Then, by adding the extra term to the propagation equations as the sum of all weights of net, the decision function for the learning is described as follows [26]:

$$\begin{gathered} \text{Min } \Delta\omega_{ij} = \alpha E(w) + \beta MSE \\ \alpha, \beta > 0 \end{gathered} \tag{15}$$

$$E(\mathrm{w}) = \frac{1}{2}\sum_{i=1}^{N} \omega_i^2 \tag{16}$$

In which, MSE, $E(\mathrm{w})$, α and β are the mean square error of NN, total sum of all weights, and filtering variables, respectively [27, 28]. Hence, when the possibility of unbiased weights decreases, the convergence of forecasting increases till it is turned to a resistant computational unit against the local optimums. According to the volume of input data as well as the learning interactions, the training rate of BR technique has been set to 70%. As it can be inferred from the Eq. (15), the propagation procedure is converted to the quadratic equation optimization which the filtering variables play a regularization role in this problem. By solving this equation and finding the minimum point for the feasible solution of variables, the propagation process will be improved as follows [29]:

$$\begin{cases} \alpha^{op} = \frac{\gamma}{2E_{\omega}(\omega^{op})} \\ \beta^{op} = \frac{N-\gamma}{2MSE} \\ \gamma = K - \alpha \text{Trace}(\text{A})^{-1} \\ \alpha = \frac{1}{\sigma_{\omega}^{2}} \end{cases} \tag{17}$$

The cooperation of γ that is the optimum diagnostic number of regularized weights with the covariance factor of input data set leads to the refinement of feasible solutions of quadratic problem. In this equation, K is the weight matrix of neural network and A is the Hessian matrix of the quadratic problem which acts as variance operator to determine the error deviation as well as α has an inverse relation with diversity of weights. It is noted that, the effective number of γ can vary from 0 to K because of the priority of input data set. Hence, the suitable set of solutions which are the best-fitted in the quadratic problem enhance the propagation modelling by diminishing the noises from the Eq. (15).

2.6 Scaled Conjugated Gradient (SCG)

In this step, the conjugate scope is used to maximize the optimization feature of dynamic technique. The concept of SCG is based on the arrangement of overall minima of quadratic problem which aims to decrease the slop of errors. In this category, we consider a gradient operator for both errors and gradient of errors. After conducting the two aforementioned techniques, SCG initialize with x_0 as the primary point of linear searching algorithm for weights in accordance to Eq. (18) [30]. The combinatory gradient can be checked with Eq. (19).

$$x_0 \in R, \ f(\text{x}) \leq f(\text{x}_0) \tag{18}$$

$$\begin{aligned} &\|\nabla f(x) - \nabla f(y)\| \leq \text{L}\|x - y\| \\ &y = g_{k+1} - g_k \\ &g_k = \nabla f(x_k) \end{aligned} \tag{19}$$

In which $\nabla f(x)$, $\nabla f(y)$ and L are the error gradient, gradient of error gradient of weight matrixes as well as x and y are the demonstration of input weights, and error gradient, respectively [31]. It has to be noted that the procedure can be achieved under the differentiability of objective function (15). In this assumptions, the propagating process of SCG can be expressed as follows (20) [32]:

$$\begin{aligned} & x_{k+1} = x_k + \alpha_k d_k \rightarrow (\omega_{k+1} = \omega_k + \alpha_k d_k) \\ & d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k \\ & \theta_{k+1} = \frac{s_k^T s_k}{y_k s_k} \\ & s_k = x_{k+1} - x_k \end{aligned} \tag{20}$$

In which d_k and α_k are the direction and step counter of searching technique [33]. According to the quasi-newton theorem, if $\beta_k = 1$ then the possibility of θ_k for being a positive definite matrix increases. Therefore, we can call the first step to be innervated as:

$$g_0 = \nabla f(x_0), d_0 = -g_0, \alpha_0 = \frac{1}{\|g_0\|} \tag{21}$$

In addition, the searching algorithm updates every iteration until the Eq. (19) can be satisfied. Hence, the Eq. (20) indicates us that the propagating process of SCG is completely depends on the optimal selecting of d_k land α_k [34]. This premises imposes us that the step counter (α_k) must be determined originally for accelerating the computation search. Thereby, the Wolf condition is implemented on the objective function for this specific purpose as in Eq. (22) [35]:

$$\begin{aligned} & f(x_k + \alpha_k d_k) - f(x_k) \leq \sigma_1 \alpha_k g_k^T d_k \\ & \nabla f(x_k + \alpha_k d_k)^T \geq \sigma_2 g_k^T d_k \end{aligned} \tag{22}$$

where σ_1 and σ_2 are the positive constant considering $0 < \sigma_1 \leq \sigma_2 < 1$. At last, the configuration of three strong computational units which compensate the propagating search that is depicted in the following section.

3 Numerical Result and Discussions

3.1 Resiliency of Hybrid Proposed Strategy

In the conducted survey, the set of electrical load demand are assembled by three steps as: the training, the validation, and the testing that are valued by a tentative options 70%, 15%, and 15%, respectively. In order to fit the assumption of proposed technique by NN, the MSE criterion serves as the best identification of error distance. This criteria is defined for each stages of DANN to reach the constraints satisfaction. Moreover, after aforementioned scaling standardize the output, the analogy between the real historical demand data set and the linearized prediction set is obtained to verify the compatibility of algorithm as shown in Fig. 5.

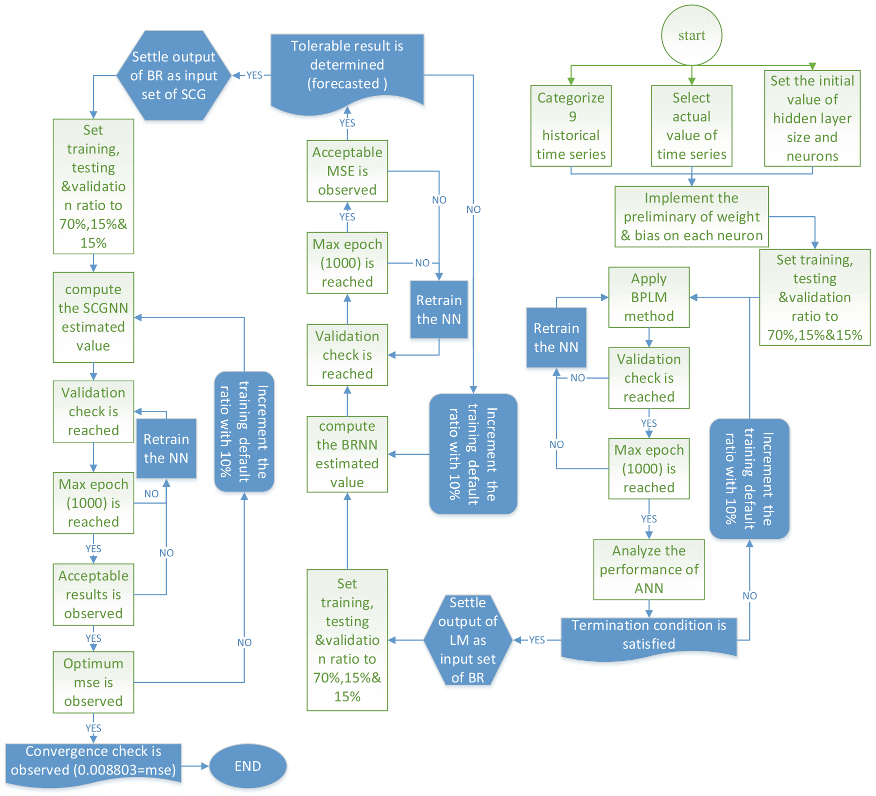

start
Categorize 9 historical time series
Select actual value of time series
Set the initial value of hidden layer size and neurons
Implement the preliminary of weight & bias on each neuron
Set training, testing &validation ratio to 70%,15%&15%
Apply BPLM method
Retrain the NN
Validation check is reached
Max epoch (1000) is reached
Increment the training default ratio with 10%
Analyze the performance of ANN
Termination condition is satisfied
Settle output of LM as input set of BR
Set training, testing &validation ratio to 70%,15%& 15%
compute the BRNN estimated value
Validation check is reached
Max epoch (1000) is reached
Retrain the NN
Acceptable MSE is observed
Tolerable result is determined (forecasted)
Increment the training default ratio with 10%
Settle output of BR as input set of SCG
Set training, testing &validation ratio to 70%,15%& 15%
compute the SCGNN estimated value
Validation check is reached
Retrain the NN
Max epoch (1000) is reached
Acceptable results is observed
Optimum mse is observed
Increment the training default ratio with 10%
Convergence check is observed (0.008803=mse)
END
YES
NO

Fig. 5 The flowchart of the proposed strategy

3.2 Robustness and Scalability

The forecasting operatory system is made by main body of NN which benefits from three learning technique that are LM, BR, and SCG. The selection of 4344 net input is allocated from the Canada, Ontario independent electricity operator system (IESO), during 1/1/2001 to 6/30/2001 till 1/1/2009 to 6/30/2009 in six month time horizons as well as 9 years which have been imported to DANN. The configuration of abovementioned structure are set as 10 hidden layer within 24 hidden neurons for each stage as well as 4320 output set iteratively. Furthermore, the comparison of predetermined and forecasted outputs in association with error functionality (MSE) are denoted as in Figs. 6, 7, 8, and 9. According to the Figs. 8 and 9, the result is accommodated with the actual historical data set. To qualify the

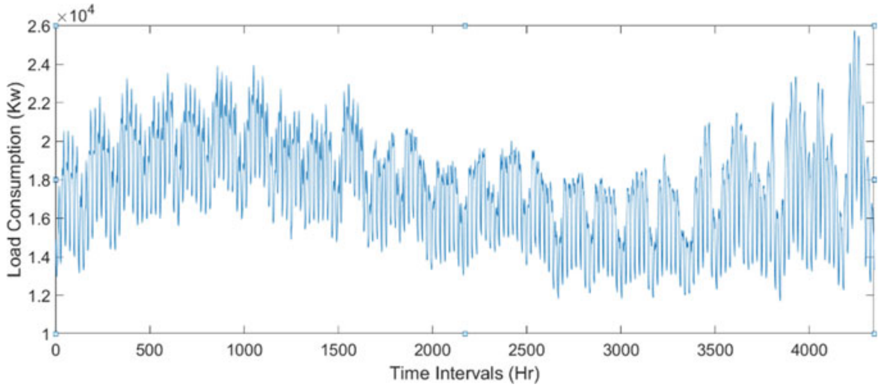


Fig. 6 Actual value of the power consumption from 1/1/2010 to 6/30/2010

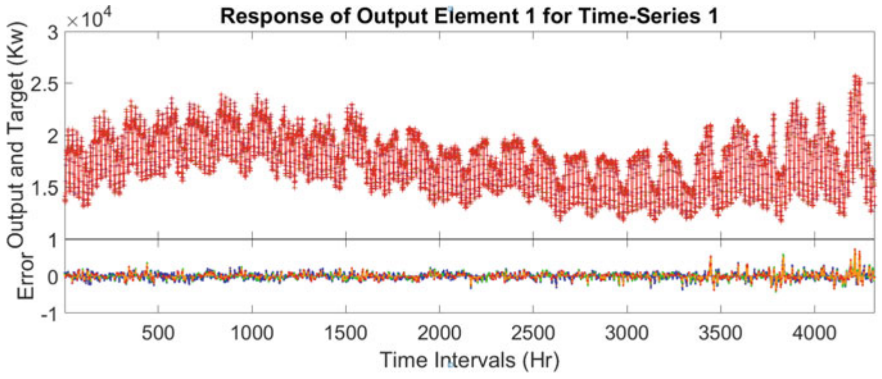


Fig. 7 The demonstration of predicted set of input

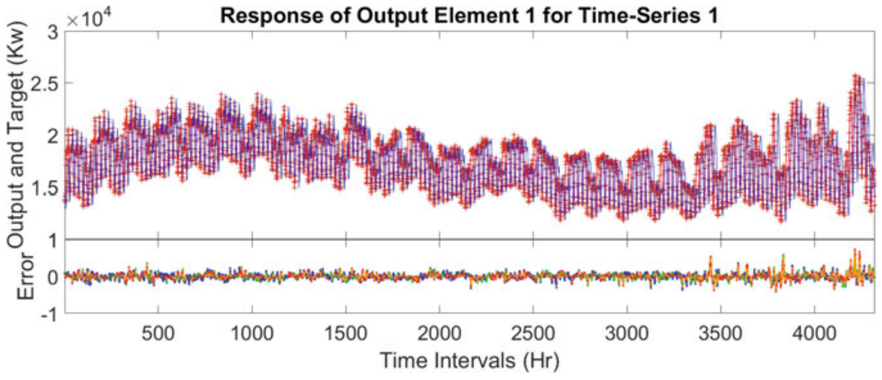


Fig. 8 The comparison of actual and estimated power consumption from 1/1/2010 to 6/30/2010

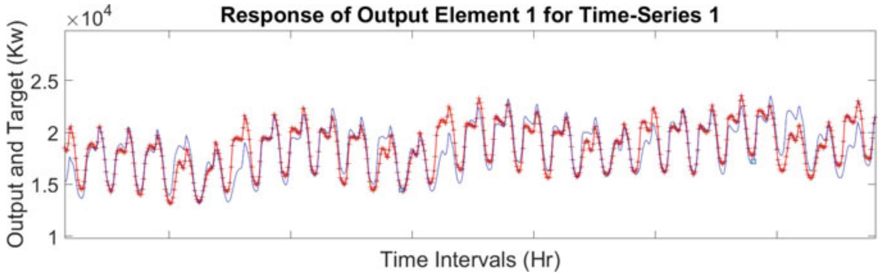


Fig. 9 The comparison of actual and estimated power consumption from 1/1/2010 to 2/1/2010

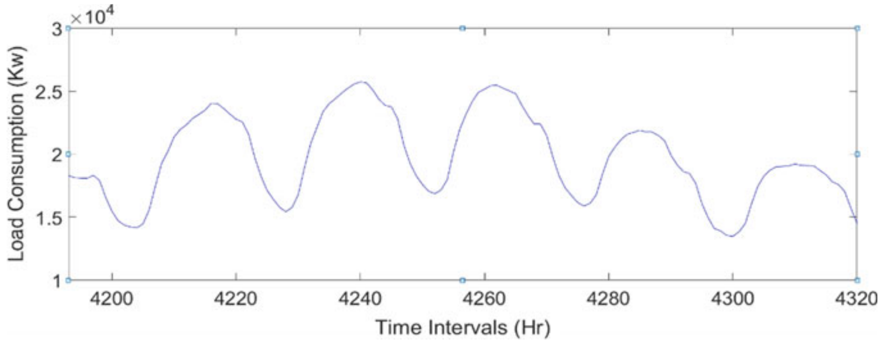


Fig. 10 The actual value of input data set for the last week during 6/24/2010 to 6/30/2010

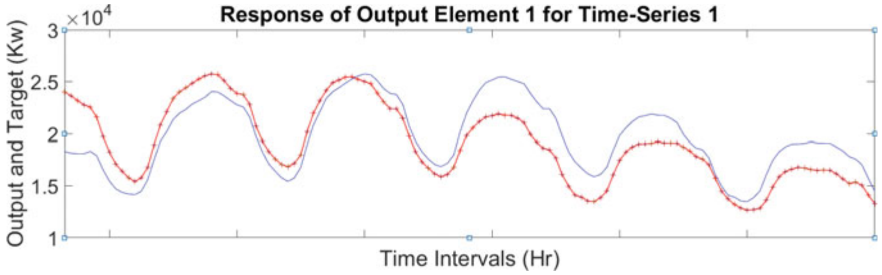
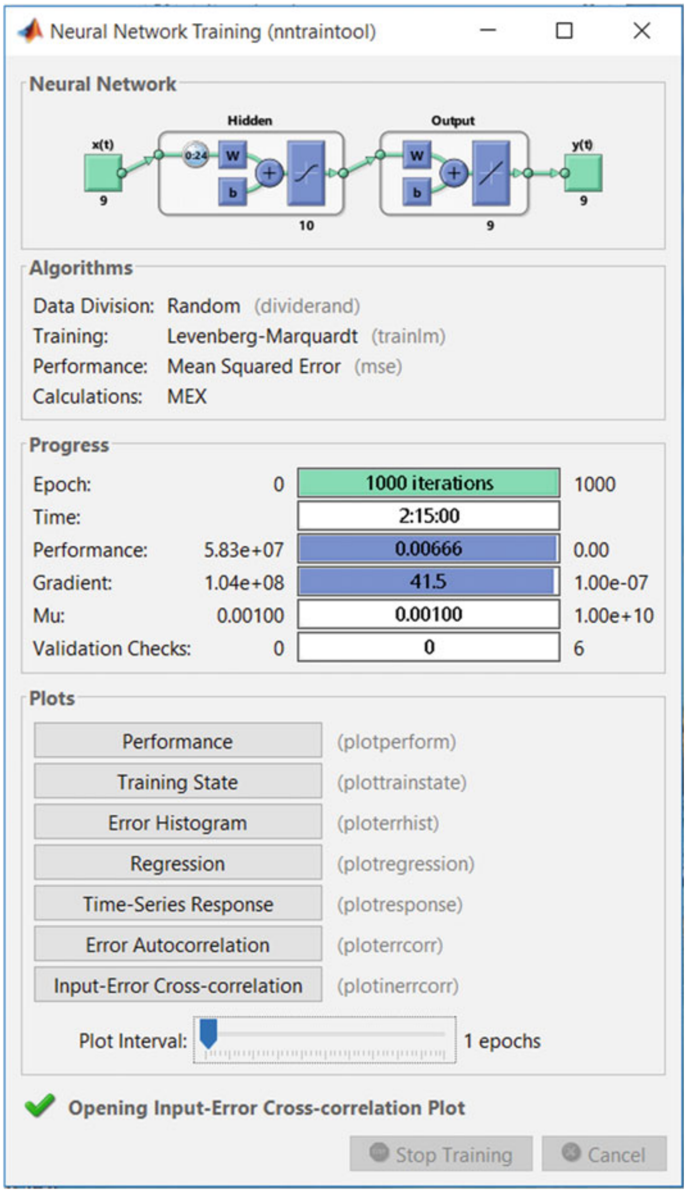


Fig. 11 The comparison of actual and estimated power consumption 6/24/2010 to 6/30/2010

contribution of simulation, the symmetric resolution of compared output of network are presented in Figs. 10 and 11 which convey the participation of measured dataset with forecasted output, respectively. As it is obvious, the blue line considered as the actual selection of measured data from IESO. In addition the red curve is defined to output value of simulated approach coherently. According to the identification of error trials of our correlated method, after 1000 epoch, the MSE is decreased to 8.803×10^{-3} (ε) which enables the conformity of strategy. The appraisal of NN construction incorporated with performance of NN are depicted in Figs. 12 and 13, respectively. Moreover Figs. 14 and 15 attached to clarify the feasibility of algorithm substantially. It is worth mentioning that, Fig. 16 is represented as linear regression view of simulated structure to fit all data set simultaneously.



Algorithms

Data Division:	Random (dividerand)
Training:	Levenberg-Marquardt (trainlm)
Performance:	Mean Squared Error (mse)
Calculations:	MEX

Progress

Epoch:	0	1000 iterations	1000
Time:		2:15:00	
Performance:	5.83e+07	0.00666	0.00
Gradient:	1.04e+08	41.5	1.00e-07
Mu:	0.00100	0.00100	1.00e+10
Validation Checks:	0	0	6

Plots

Performance	(plotperform)
Training State	(plottrainstate)
Error Histogram	(ploterrhist)
Regression	(plotregression)
Time-Series Response	(plotresponse)
Error Autocorrelation	(ploterrcorr)
Input-Error Cross-correlation	(plotinerrcorr)

Plot Interval: 1 epochs

✓ Opening Input-Error Cross-correlation Plot

Fig. 12 The configuration of LM method

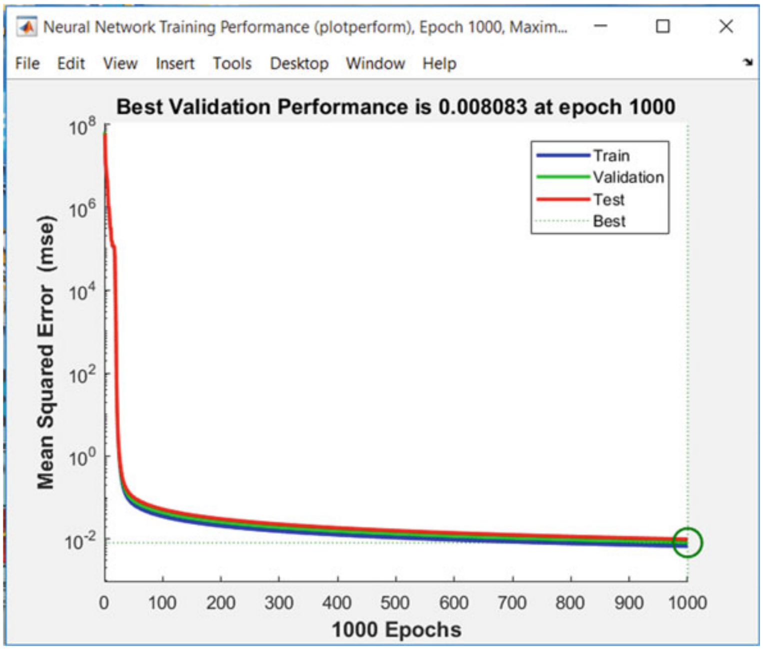


Fig. 13 The performance of LM-DANN

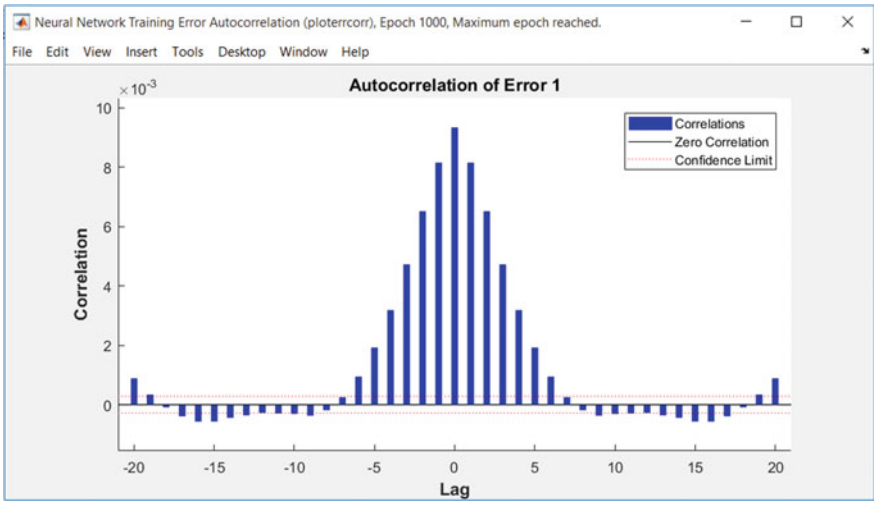


Fig. 14 The autocorrelation of error

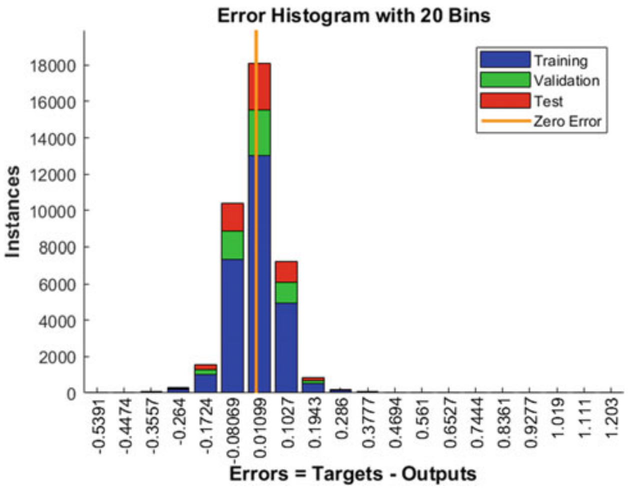


Fig. 15 The error histogram of proposed architecture

4 Conclusion

All in all, we have assumed the advantage of ANN for the long term forecasting of electrical consumption as well as to predict the desired data set using the error criterion. In this paper, the intermittent nature of our problem has been depicted us that implementing proposed method is applicable for the uncertain frequency data sets. Thereby, the historical sets is reported by IESO, Canada's power network, for the purpose of estimating . Plus, after determining the composition of DANN, the regulating steps which is guided by training progress of demand curve are applied to gain dependable results. Consequently, the simulation performance of DANN covers the sensitivity and practicable operating of proposed architecture which is obtained as tolerable minimum MSE.

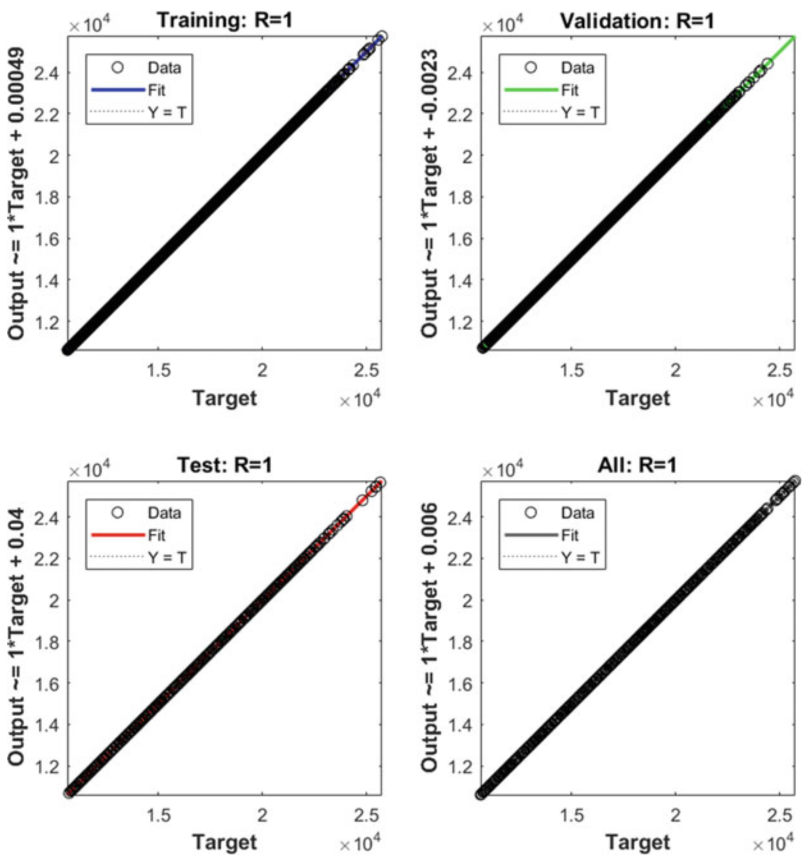


Fig. 16 The regression criterion for proposed architecture

MATLAB Code

Note that sheets 1 and 2 are the historical data and the time series data set of input, respectively. In addition, the training progress should be applied after each stages met their termination condition and gained acceptable performance. The dash sign is separated all training slops in order.

```
% Import the file which is the historical data set with
respect to month/day/year in Excel format

function [newData1] = importfile3(fileToRead1)
sheetName='Sheet1';
[numbers, strings] = xlsread(fileToRead1, sheetName);
if ~isempty(numbers)
    newData1.data =  numbers;
end
if ~isempty(strings)
    newData1.textdata =  strings;
end
function [newData1] = importfile3(fileToRead1)

% Import the file which is the actual data in Excel
format.

sheetName='Sheet2';
[numbers, strings] = xlsread(fileToRead1, sheetName);
if ~isempty(numbers)
    newData1.data =  numbers;
end
if ~isempty(strings)
    newData1.textdata =  strings;
end

% Construct the main body of NN and adjust the hidden
layer size and its neurons in another Matlab file.

inputSeries = tonndata(data,false,false);
targetSeries = tonndata(data,false,false);
inputDelays = 0:24;
hiddenLayerSize = 10;
net = timedelaynet(inputDelays,hiddenLayerSize);
[inputs,inputStates,layerStates,targets] = pre-
parets(net,inputSeries,targetSeries);

% set the primary values of training, validation and
testing stages.
```

```
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Determine the training approach.
net.trainFcn = 'trainlm'
[net,tr] =
train(net,inputs,targets,inputStates,layerStates);

% Check the validation in accord with the flowchart.

outputs = net(inputs,inputStates,layerStates);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Plot the forecasted output.

figure, plotperform(tr)
figure, plottrainstate(tr)
figure, plotresponse(targets,outputs)
figure, ploterrcorr(errors)
figure, plotinerrcorr(inputs,errors)

nets = removedelay(net);
[xs,xis,ais,ts] = pre-
parets(nets,inputSeries,targetSeries);
ys = nets(xs,xis,ais);
earlyPredictPerformance = perform(net,tc,yc)
% Make another Matlab files and apply the code for both
BRNN and SCGNN.

% Construct the main body of NN and adjust the hidden
layer size and its neurons in another Matlab file.

inputSeries = tonndata(data,false,false);
targetSeries = tonndata(data,false,false);
inputDelays = 0:24;
hiddenLayerSize = 10;
net = timedelaynet(inputDelays,hiddenLayerSize);
[inputs,inputStates,layerStates,targets] =
preparets(net,inputSeries,targetSeries);

% set the primary values of training, validation and
testing stages.

net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Determine the training approach.
net.trainFcn = 'trainbr'
[net,tr] =
train(net,inputs,targets,inputStates,layerStates);

% Check the validation in accord with the flowchart.
outputs = net(inputs,inputStates,layerStates);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Plot the forecasted output.

figure, plotperform(tr)
figure, plottrainstate(tr)
figure, plotresponse(targets,outputs)
figure, ploterrcorr(errors)
figure, plotinerrcorr(inputs,errors)

nets = removedelay(net);
[xs,xis,ais,ts] = pre-
parets(nets,inputSeries,targetSeries);
ys = nets(xs,xis,ais);
earlyPredictPerformance = perform(net,tc,yc)

% Construct the main body of NN and adjust the hidden
layer size and its neurons in another Matlab file.

inputSeries = tonndata(data,false,false);
targetSeries = tonndata(data,false,false);
inputDelays = 0:24;
hiddenLayerSize = 10;
net = timedelaynet(inputDelays,hiddenLayerSize);
[inputs,inputStates,layerStates,targets] =
 preparets(net,inputSeries,targetSeries);

% set the primary values of training, validation and
testing stages.

net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Determine the training approach.
net.trainFcn = 'trainscg'
```

```
[net,tr] =
train(net,inputs,targets,inputStates,layerStates);

% Check the validation in accord with the flowchart.
outputs = net(inputs,inputStates,layerStates);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Plot the forecasted output.

figure, plotperform(tr)
figure, plottrainstate(tr)
figure, plotresponse(targets,outputs)
figure, ploterrcorr(errors)
figure, plotinerrcorr(inputs,errors)

nets = removedelay(net);
[xs,xis,ais,ts] = pre-
parets(nets,inputSeries,targetSeries);
ys = nets(xs,xis,ais);
earlyPredictPerformance = perform(net,tc,yc)
```

References

1. Jabari, F., Nojavan, S., Ivatloo, B.M., Sharifian, M.B.: Optimal short-term scheduling of a novel tri-generation system in the presence of demand response programs and battery storage system. Energy Convers. Manag. **122**, 95–108 (2016)
2. Jabari, F., Nojavan, S., Ivatloo, B.M.: Designing and optimizing a novel advanced adiabatic compressed air energy storage and air source heat pump based μ-Combined Cooling, heating and power system. Energy **116**, 64–77 (2016)
3. Jabari, F., Shamizadeh, M., Mohammadi-ivatloo, B.: Dynamic economic generation dispatch of thermal units incorporating aggregated plug-in electric vehicles. In: 3rd International Conference of IEA Technology and Energy Management, pp. 28–29 (2017)
4. Jabari, F., Seyedia, H., Ravadanegh, S.N., Ivatloo, B.M.: Stochastic contingency analysis based on voltage stability assessment in islanded power system considering load uncertainty using MCS and k-PEM. In: Handbook of Research on Emerging Technologies for Electrical Power Planning, Analysis, and Optimization, pp. 12–36. IGI Global (2016)
5. Jabari, F., Masoumi, A., Mohammadi-ivatloo, B.: Long-term solar irradiance forecasting using feed-forward back-propagation neural network. In: 3rd International Conference of IEA, Tehran, Iran, 28th Feb–1st Mar 2017
6. Tajer, A.: Load forecasting via diversified state prediction in multi-area power networks. IEEE Transactions on Smart Grid **8**(6), 2675–2684 (2016)
7. Khan, G.M., Zafari, F.: Dynamic feedback neuro-evolutionary networks for forecasting the highly fluctuating electrical loads. Genet. Program Evolvable Mach. **17**, 391–408 (2016)
8. Mitchell, G., Bahadoorsingh, S., Ramsamooj, N., Sharma, C.: A comparison of artificial neural networks and support vector machines for short-term load forecasting using various load types. In: 2017 IEEE Manchester PowerTech, pp. 1–4 (2017)

9. Son, H., Kim, C.: Short-term forecasting of electricity demand for the residential sector using weather and social variables. Resour. Conserv. Recycl. **123**, 200–207 (2016)
10. Guo, H., Chen, Q., Xia, Q., Kang, C., Zhang, X.: A monthly electricity consumption forecasting method based on vector error correction model and self-adaptive screening method. Int. J. Electr. Power Energy Syst. **95**, 427–439 (2018)
11. Le Cam, M., Zmeureanu, R., Daoud, A.: Cascade-based short-term forecasting method of the electric demand of HVAC system. Energy **119**, 1098–1107 (2017)
12. Li, B., Zhang, J., He, Y., Wang, Y.: Short-term load-forecasting method based on wavelet decomposition with second-order gray neural network model combined with ADF test. IEEE Access **5**, 16324–16331 (2017)
13. Szoplik, J.: Forecasting of natural gas consumption with artificial neural networks. Energy **85**, 208–220 (2015)
14. Monjoly, S., André, M., Calif, R., Soubdhan, T.: Hourly forecasting of global solar radiation based on multiscale decomposition methods: a hybrid approach. Energy **119**, 288–298 (2017)
15. Petković, D., Nikolić, V., Mitić, V.V., Kocić, L.: Estimation of fractal representation of wind speed fluctuation by artificial neural network with different training algorothms. Flow Meas. Instrum. **54**, 172–176 (2017)
16. Xue, X.: Prediction of daily diffuse solar radiation using artificial neural networks. Int. J. Hydrogen Energy **42**, 28214–28221 (2017)
17. Dongmei, H., Shiqing, H., Xuhui, H., Xue, Z.: Prediction of wind loads on high-rise building using a BP neural network combined with POD. J. Wind Eng. Ind. Aerodyn. **170**, 1–17 (2017)
18. Ramasamy, P., Chandel, S., Yadav, A.K.: Wind speed prediction in the mountainous region of India using an artificial neural network model. Renew. Energy **80**, 338–347 (2015)
19. Yadav, A.K., Chandel, S.: Identification of relevant input variables for prediction of 1-minute time-step photovoltaic module power using artificial neural network and multiple linear regression models. Renew. Sustain. Energy Rev. **77**, 955–969 (2017)
20. da Silva, T.V., Monteiro, R.V.A., Moura, F.A.M., Albertini, M.R.M.C., Tamashiro, M.A., Guimaraes, G.C.: Performance analysis of neural network training algorithms and support vector machine for power generation forecast of photovoltaic panel. IEEE Latin Am. Trans. **15**, 1091–1100 (2017)
21. Lourakis, M., Argyros, A.: The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm. Technical Report 340. Institute of Computer Science-FORTH, Heraklion, Crete, Greece (2004)
22. Lera, G., Pinzolas, M.: Neighborhood based Levenberg-Marquardt algorithm for neural network training. IEEE Trans. Neural Networks **13**, 1200–1203 (2002)
23. Wilamowski, B.M., Yu, H.: Improved computation for Levenberg–Marquardt training. IEEE Trans. Neural Networks **21**, 930–937 (2010)
24. Giovanis, D.G., Papaioannou, I., Straub, D., Papadopoulos, V.: Bayesian updating with subset simulation using artificial neural networks. Comput. Methods Appl. Mech. Eng. **319**, 124–145 (2017)
25. Kumar, P., Merchant, S., Desai, U.B.: Improving performance in pulse radar detection using Bayesian regularization for neural network training. Digit. Signal Proc. **14**, 438–448 (2004)
26. Sun, Z., Chen, Y., Li, X., Qin, X., Wang, H.: A Bayesian regularized artificial neural network for adaptive optics forecasting. Opt. Commun. **382**, 519–527 (2017)
27. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In International Conference on Machine Learning, pp. 1058–1066 (2013)
28. MacKay, D.J.: Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks. Netw. Comput. Neural Syst. **6**, 469–505 (1995)
29. Williams, P.M.: Bayesian regularization and pruning using a Laplace prior. Neural Comput. **7**, 117–143 (1995)
30. Cheng, C., Chau, K., Sun, Y., Lin, J.: Long-term prediction of discharges in Manwan Reservoir using artificial neural network models. In: International Symposium on Neural Networks, pp. 1040–1045 (2005)

31. Møller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw. **6**, 525–533 (1993)
32. Sözen, A., Arcaklioğlu, E., Özalp, M.: A new approach to thermodynamic analysis of ejector–absorption cycle: artificial neural networks. Appl. Therm. Eng. **23**, 937–952 (2003)
33. Estevez, P.A., Vera, P., Saito, K.: Selecting the most influential nodes in social networks. In: 2007 International Joint Conference on Neural Networks, IJCNN 2007, pp. 2397–2402 (2007)
34. Sözen, A., Özalp, M., Arcaklioğlu, E.: Investigation of thermodynamic properties of refrigerant/absorbent couples using artificial neural networks. Chem. Eng. Process. **43**, 1253–1264 (2004)
35. Babaie-Kafaki, S.: Two modified scaled nonlinear conjugate gradient methods. J. Comput. Appl. Math. **261**, 172–182 (2014)