Information Sciences 367–368 (2016) 1078–1093

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Random vector functional link network for short-term electricity load demand forecasting

Ye Ren [a], P.N. Suganthan [a,*], N. Srikanth [b], Gehan Amaratunga [c]

[a] School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Ave., 639798 Singapore
[b] Energy Research Institute @ NTU (ERI@N), #06-04 CleanTech One 1 CleanTech Loop, 637141 Singapore
[c] Department of Engineering, University of Cambridge, 9 JJ Thomson Ave., Cambridge CB3 0FA, UK

ARTICLE INFO

Article history:
Received 13 March 2015
Revised 8 November 2015
Accepted 29 November 2015
Available online 6 January 2016

Keywords:
Random weights
Random vector functional link
Neural network
Time series forecasting
Electricity load demand forecasting

ABSTRACT

Short-term electricity load forecasting plays an important role in the energy market as accurate forecasting is beneficial for power dispatching, unit commitment, fuel allocation and so on. This paper reviews a few single hidden layer network configurations with random weights (RWSLFN). The RWSLFN was extended to eight variants based on the presence or absence of input layer bias, hidden layer bias and direct input–output connections. In order to avoid mapping the weighted inputs into the saturation region of the enhancement nodes' activation function and to suppress the outliers in the input data, a quantile scaling algorithm to re-distribute the randomly weighted inputs is proposed. The eight variations of RWSLFN are assessed using six generic time series datasets and 12 load demand time series datasets. The result shows that the RWSLFNs with direct input–output connections (known as the random vector functional link network or RVFL network) have statistically significantly better performance than the RWSLFN configurations without direct input–output connections, possibly due to the fact that the direct input–output connections in the RVFL network emulate the time delayed finite impulse response (FIR) filter. However the RVFL network has simpler training and higher accuracy than the FIR based two stage neural network. The RVFL network is also compared with some reported forecasting methods. The RVFL network overall outperforms the non-ensemble methods, namely the persistence method, seasonal autoregressive integrated moving average (sARIMA), artificial neural network (ANN). In addition, the testing time of the RVFL network is the shortest while the training time is comparable to the other reported methods. Finally, possible future research directions are pointed out.

1. Introduction

Short-term electricity load demand forecasting targets to predict the future load demand ranging from a few minutes to 1 day ahead [2,19]. In a contemporary competitive energy market, as power systems become deregulated and involve distributed power sources, electricity demand becomes more complicated in both temporally and spatially. Since electricity load demand is closely related to the electricity price, load forecasting plays an important role in the energy market

* Corresponding author. Tel.: +65 6705404; fax.: +65 67933318.
E-mail addresses: re0003ye@ntu.edu.sg (Y. Ren), epnsugan@ntu.edu.sg (P.N. Suganthan), nsrikanth@ntu.edu.sg (N. Srikanth), gaja1@hermes.cam.ac.uk (G. Amaratunga).
URL: http://www.ntu.edu.sg/home/epnsugan (P.N. Suganthan)

http://dx.doi.org/10.1016/j.ins.2015.11.039
0020-0255/

[19]. From the energy generation point of view, load forecasting is beneficial for electric utility operations such as power dispatching, unit commitment, fuel allocation and network diagnosis [2]. The significance of forecasting error with respect to power system operational cost was stated in [2,19]: 1% increase in error can result in wastage of millions of dollars. Therefore, improving the forecasting accuracy can significantly reduce the power system operational cost.

Fig. 3a in Section 5.1 shows a small fraction of load demand time series (TS). From the plot, we can observe the non-linearity of the load demand TS because it is usually affected by a lot of exogenous factors such as weather, special occasions, economy, and so on [9], thereby making the modeling of load demand TS stochastic. However, it also reveals strong seasonal components such as daily, weekly, monthly or yearly repetitions. These properties, on the other hand, makes the modeling somewhat deterministic [13].

In the literature, there are numerous methods for load forecasting and they can be categorized into (i) statistical methods, (ii) machine learning methods and (iii) hybrid methods. Statistical methods use mathematical equations and statistical theories to model and forecast the TS: it is fast but with limitations (distributions, stationarity, linearity, etc.). The commonly used statistical methods are exponential smoothing [32] and autoregressive moving average (ARMA) and so on [33]. Machine learning methods use supervised learning to model a portion of historical data (training data). Once the model is optimized based on the training data, it can be applied to forecast the unknown future data (testing data). Some well-known machine learning methods are artificial neural networks (ANN) [19,23], support vector regression (SVR) [12,18,22], etc. Combining more than one forecasting method to form a single forecasting method is known as a hybrid method. The combination can be sequential or parallel. For a sequentially combined hybrid method, one forecasting method's output is another forecasting method's input, and the final output is the output of the last forecasting method. For a parallel combined hybrid method, the training data is either bootstrapped to multiple training data sets or decomposed to a collection of training data sets first. Subsequently, each data set is trained by an individual method. Finally, the trained forecasting methods are used to predict the future data and the outputs are aggregated to form the final prediction. Negative correlation learning [3,9] and wavelet [17,20,25,35,36] and Empirical Mode Decomposition (EMD) [14,16,30] based methods are in this category.

ANN is a popular method among the machine learning based load forecasting methods. There are several ANN based forecasting methods reported in the literature [15,28,34]. The usual way to train the ANN is through back propagation (BP). But in [31], a single hidden layer neural network with random weights (RWSLFN) was reported with random weight assignment between input and hidden layers and least square estimation on the output weights as the training method. In addition, we will discuss another type of ANN which is called random vector functional link network (RVFL network) [5,6,27]. Although it is similar to RWSLFN reported in [31], it has direct input–output connections between input and output neurons. This paper will compare these neural networks and their forecasting capability in short-term load demand forecasting.

The remaining of the paper is organized as follows: Section 2 reviews different variants of RWSLFN/RVFL network; Section 3 states the essentials of the experiment configurations, feature selection methods, parameter optimization procedure and error measures. Section 4 assesses the performance of different variants on six generic TS data; Section 5 discusses the performance of RWSLFN/RVFL network with eight variants on 12 load demand TS, and Section 6 concludes the paper and suggests possible future research directions.

2. Review of related neural network structures

ANN is one of the most popular algorithms in machine learning. ANN can be categorized according to the connectionism: feedforward and recurrent. The main difference between feedforward neural network (FNN) and recurrent neural network (RNN) is that the FNN does not have connections from the hidden layers or output layer back to the hidden layer. RNN not only has the connections that the FNN has but also has the connections from hidden layers or output layer back to the input layer. The recurrent connections are usually delayed so that the RNN can exhibit better dynamic temporal behavior. There are two typical RNN architectures: Elman RNN and Jordan RNN. If there exists connections from the hidden layer to the input layer, it is an Elman RNN [8]. If the feedback connections are from the output layer to the input layer, it is a Jordan RNN [8].

In the literature, FNN is more widely used. It usually has one input layer, one output layer and several hidden layers as shown in Fig. 1. The neurons in the adjacent layers are connected. But there is no interconnection of neurons within the same layer or across non-adjacent layers.

In the input layer, each neuron i_m, $m \in \{1, \ldots, M\}$ takes a feature of the input vector and passes to the hidden layer neurons. Each neuron in the nth hidden layer h_{n,k_n} $n \in \{1, \ldots, N\}$, $k_n \in \{1, \ldots, K_N\}$ is formed by a nonlinear weighted sum of the outputs of the input layer or the preceding hidden layer (except the last hidden layer):

$$h_{1,k_1} = f\left(\sum_{m=0}^{M} w_{m,k_1} i_m\right), \quad \forall k_1 \in \{1, \ldots, K_1\} \tag{1}$$

$$h_{n,k_n} = f\left(\sum_{k_{n-1}=0}^{K_{n-1}} w_{k_{n-1},k_n} h_{n-1,k_{n-1}}\right), \quad \forall k_n \in \{2, \ldots, K_n\} \tag{2}$$

where $f(\cdot)$ is a nonlinear activation function, $w_{0,k_n} = 1$, $k_n \in \{1, \ldots, K_n\}$ denotes the input layer and hidden layer biases, M is the number of input layer neurons, N is the number of hidden layers, K_n is number of nth hidden layer neurons,

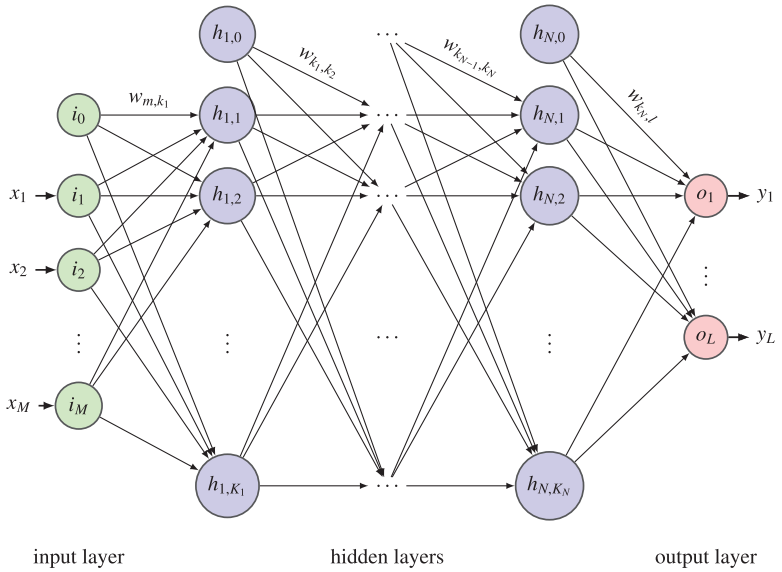

Fig. 1. Schematic diagram of an ANN, where M is the number of input layer neurons, N is the number of hidden layers, K_n is number of nth hidden layer neurons, L is the number of output neurons, w_{m,k_1} are the weights between the input and hidden layer neurons, and w_{k_{n-1},k_n} are the weights between the hidden layer neurons and $w_{k_N,l}$ is the weights between the hidden and output layer neurons, x is the input data and y is the output from ANN.

w_{m,k_1} are the weights between the input and hidden layer neurons, and w_{k_{n-1},k_n} are the weights between the hidden layer neurons.

The commonly used activation functions are logistic sigmoid *logsig* function and hyperbolic tangent *tanh* function:

$$logsig(x) = \frac{1}{1+e^{-x}} \tag{3}$$

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{4}$$

The output neuron is a weighted sum of all the outputs of the last hidden layer neurons:

$$o_l = \sum_{k_N=0}^{K_N} w_{k_N,l} h_{N,k_N}, \quad \forall l \in \{1,\ldots,L\} \tag{5}$$

where L is the number of output neurons, and $w_{k_N,l}$ are the weights between the hidden and output layer neurons.

To determine the optimal output value, the weights w_{m,k_1}, $w_{k_N,l}$ and the weights between the hidden neurons should be determined and they are usually determined by the BP learning [23]. However, the BP method requires an iterative training process which is computationally demanding. Further, BP is also highly likely to be trapped in a local minimum.

2.1. Single hidden layer neural network

One of the most popular feedforward neural network topology is single hidden layer neural network (SLFN). The neurons between adjacent layers (input to hidden layer and hidden to output layer) are connected. There is no interconnection of neurons within the same layer or across non-adjacent layers (input to output layer).

2.2. SLFN with random weights

In 1992, Schmidt et al. [31] reported an SLFN with fixed random weights (RWSLFN) assigned to the input to hidden layer connections. The reported RWSLFN [31] has the same structure as a conventional SLFN. The activation function of the hidden neurons is a *logsig* function.

Training of a general SLFN is to minimize the squared output error by finding the optimal hidden layer weights $\mathbf{W}_h = \{w_{m,k}, b_{i,k}\}$ and output layer weights $\mathbf{W}_o = \{w_{k,l}, b_{h,l}\}$.

$$\min \varepsilon^2 = \sum_{i=1}^{N}(y_i - \hat{y}_i) = \sum_{i=1}^{N}\left(y_i - \sum_{k=1}^{K} w_{k,l} f\left(\sum_{m=1}^{M} w_{m,k} i_m + b_{i,k}\right) + b_{h,l}\right)^2 \tag{6}$$

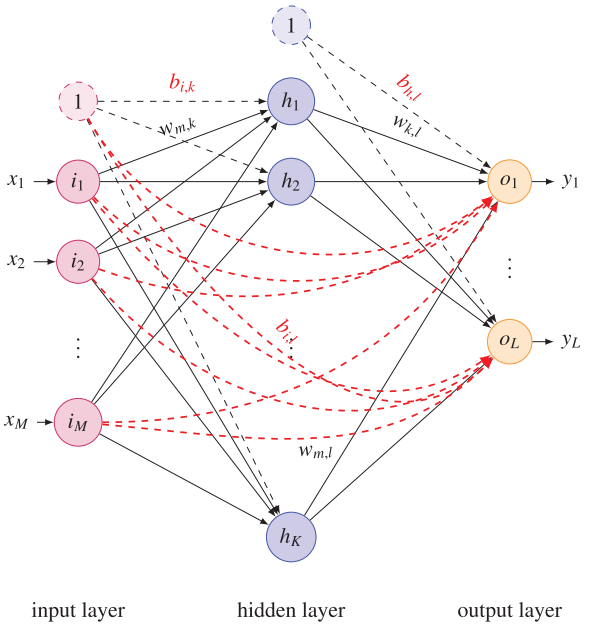

Fig. 2. Schematic diagram of an RVFL network, the dashed red arrows show the direct connections between the input neurons and the output neurons, the dashed black arrows show the connections from a bias neuron, $f(\cdot)$ is a *logsig* activation function. M is the number of input neurons and K is the number of hidden neurons. $w_{m,k}$ are the weights between the input and hidden layer neurons, $w_{k,l}$ are the weights between the hidden and output layer neurons and $b_{i,k}$ and $b_{h,j}$ are the input layer and hidden layer biases, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The difference between the reported RWSLFN and the conventional SLFN is that for the conventional SLFN, the optimal hidden layer and output layer weights are determined by the BP whereas in the reported RWSLFN, the hidden layer weights $\mathbf{W}_h$ are randomly sampled from a uniform distribution in $[-1, 1]$ and the output layer weights $\mathbf{W}_o$ are optimized by a least squares method.

The RWSLFN does not require iteratively updating the input layer to hidden layer weights during training, which significantly speeds up the training process. However, there are potential problems with the randomly assigned weights such as instability or overfitting of the network due to rank deficiency and non-linear mapping. Although the authors specified the distribution of the random numbers as uniformly distributed in $[-1, 1]$, the generalization of this random weight assignment is questionable.

2.3. SLFN with direct input–output connections (RVFL network)

In 1994, Pao et al. [27] proposed an RVFL network that combines the advantage of random weights and functional link [21] together. The RVFL network is an SLFN with direct connections from the input layer to the output layer as shown in Fig. 2. The dashed arrows show the direct connections from the input neurons to the output neurons.

The proposed RVFL network in [27] has a set of nodes called enhancement nodes, which are equivalent to the hidden layer in the conventional SLFN. Similar to [31], the weights from the input layer to the enhancement nodes are randomly assigned within a suitable range. In [27], the authors proposed to use a conjugated BP to tune the weights from the input layer to the output layer as well as the weights from the enhancement nodes to the output layer. However, the authors also stated that applying the least square method is possible if the matrix inversion is feasible [27]. For the interval of the random weights, the authors gave a brief rule of thumb: the value of the weights $\mathbf{W_K}$ should constrain the activation functions $f(\cdot)$ away from the saturation region most of the time [27].

In 1996, Chen [5] reported an RVFL network for function approximation and interpolation. The structure of the reported RVFL network is the same as [27] with direct connections from the input to the output layers. However, the activation function of the enhancement node in Chen's RVFL network is *tanh* whereas it is *logsig* in Pao's RVFL network. In addition, Chen's RVFL network has an additional activation function applied to the output layer when it is applied to solve function approximation problems [5] while for TS forecasting, this activation function in the output layer is not used [6].

Another novelty of Chen's RVFL network is that it has an online fast training capability. When adding a new training sample to the existing trained RVFL network model, the output weights $\mathbf{W}_o$ need not be re-trained, but are updated based on the previously calculated data and the new training sample. Similarly, when a new enhancement node is added, total retraining is avoided by updating the model with the existing data and the random weights associated with the newly added enhancement node. The details of the fast training procedures can be found in [5,6]. A requisite of this online training is

Table 1
RWSLFN/RVFL network with different configurations, $f(\cdot)$ is a *logsig* activation function. M is the number of input neurons and K is the number of hidden neurons. $w_{m,k}$ are the weights between the input and hidden layer neurons, $w_{k,l}$ are the weights between the hidden and output layer neurons and $b_{i,(k/l)}$ and $b_{h,j}$ are the input layer and hidden layer biases, respectively.

Method	Input layer bias	Hidden layer bias	Input–output connection	Formula
M1	√	√	√	$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m + b_{i,k})$ $o_l = \sum_{k=1}^{K} w_{k,l} h_k + b_{h,l} + \sum_{m=1}^{M} w_{m,l} i_m + b_{i,l}$
M2 [31]	√	√	×	$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m + b_{i,k})$ $o_l = \sum_{k=1}^{K} w_{k,l} h_k + b_{h,l}$
M3 [6,27]	√	×	√	$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m + b_{i,k})$ $o_l = \sum_{k=1}^{K} w_{k,l} h_k + \sum_{m=1}^{M} w_{m,l} i_m + b_{i,l}$
M4	√	×	×	$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m + b_{i,k})$ $o_l = \sum_{k=1}^{K} w_{k,l} h_k$
M5	×	√	√	$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m)$ $o_l = \sum_{k=1}^{K} w_{k,l} h_k + b_{h,l} + \sum_{m=1}^{M} w_{m,l} i_m$
M6	×	√	×	$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m)$ $o_l = \sum_{k=1}^{K} w_{k,l} h_k + b_{h,l}$
M7	×	×	√	$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m)$ $o_l = \sum_{k=1}^{K} w_{k,l} h_k + \sum_{m=1}^{M} w_{m,l} i_m$
M8	×	×	×	$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m)$ $o_l = \sum_{k=1}^{K} w_{k,l} h_k, \forall k \in \{1, \ldots, K\}, \forall l \in \{1, \ldots, L\}$

that when a new training sample or a new enhancement node is added to the existing RVFL network, the rank of the augmented extended input pattern matrix $[\mathbf{X}|f(\mathbf{W}^T\mathbf{X} + b)]$ should be greater than the extended input pattern matrix before augmentation.

The main difference between RWSLFN and the RVFL network is that the RWSLFN does not have the direct input–output connections. Without the linear estimations from the input to the output, the RWSLFN may be unstable due to the randomly generated input to hidden layer weights. In the following sections, the influence of the direct input–output connections will be assessed.

3. Experiment setup

3.1. Variations on RWSLFN/RVFL network

By varying the network components such as input layer bias, hidden layer bias and input– output connections, we can obtain eight different RWSLFN/RVFL network configurations. The eight RWSLFN/RVFL network configurations are shown in Table 1.

3.2. Time series cross validation

The time series cross validation (TS-CV) is used to determine the optimal parameters of the methods. The TS-CV process is slightly different from CV for classification and regression because the TS data are sequential and auto-correlated. The conventional k-fold CV randomly partitions the training data into k subsets, then $k-1$ subsets are used for training and the remaining subset is used for validation. The process repeats for k times, resulting k error measures. For each parameter combination, there are k error measures, and these error measures are averaged for that particular combination. The optimal parameter combination is therefore the one with the lowest error measure. For TS-CV, the TS is partitioned into $k + (k-1)$ sub series and a rolling window repeats the training and validation dataset selection for k times. After k-fold TS-CV, the optimal parameters can be selected according to the error measures. The TS-CV is used to select the optimal number of enhancement nodes for the RVFL network and optimal number of hidden neurons for RWSLFN in the experiment.

3.3. Error measures

In this paper, normalized root mean square error (nRMSE) is used to measure the performance. It is defined as:

$$\mathrm{nRMSE} = \frac{1}{y_{\max} - y_{\min}} \sqrt{\mathrm{E}[(\hat{y} - y)^2]} \tag{7}$$

where $\hat{y}$ is the predicted data, y is the desired target data and n is the number of data points in the TS.

3.4. Feature selection

The input data are based on the historical data. In this paper, we evaluated the forecasting methods on the datasets with three different feature sets. The first input dataset comprised historical data up to a lag defined by the user $L_{\max}$. It is denoted as 'full' feature set in the paper.

The second input dataset is a subset of the 'full' feature set but with Partial autocorrelation function (PACF) selected features. PACF calculated the cross-correlations among different lags in the historical data. It can be used to determine the p order of an ARMA model. In this paper, we employ PACF to score the input features based on the correlation coefficient with respect to the future data and select the input features whose scores exceed the confidence margin ($\pm\frac{1.96}{\sqrt{n}}$) This input dataset is denoted as 'PACF' feature set.

Another feature set is formed according to a seasonal auto-regression (sAR) model: $AR(p) \times (P)[s]$. An $AR(p) \times (P)[s]$ which is to model the future load demand $y(t+h)$ as:

$$\begin{aligned}\hat{y}(t+h) = {} & \phi_1 y(t) + \cdots + \phi_p y(t-p+1) + \Phi_1\phi_1 y(t-s) + \cdots + \Phi_1\phi_p y(t-p+1-s) \\ & + \cdots + \Phi_P\phi_1 y(t-Ps) + \cdots + \Phi_P\phi_p y(t-p+1-Ps)\end{aligned} \tag{8}$$

where p and P are the number of lags, ϕ and Φ are coefficients and s is the period. However, if the TS does not have seasonal components, this sAR feature selection is not applicable.

3.5. Range determination of the random weights

The weights of the input to hidden layer connections are randomly assigned, and the hidden neuron then maps the input vectors to a non-linear space based on a non-linear activation function (in this paper, *logsig* is used). However, there is little research on the range of the random weights. In [27], the authors stated that the values of the weights should constrain the sigmoid activation function away from the saturation region most of the time. But this only served as a rule of thumb. And in [31], the authors used uniformly distributed random numbers from $[-1, 1]$ as the random weights.

Zhang and Suganthan [37] proposed a method to find the optimal interval $[-S, +S]$ for the random weights by a range scaling algorithm. The method iterated on S while using CV to determine the best S value (corresponds to the smallest error). The authors found that for classification cases, $S = 1$ may not always lead to optimal performance.

For TS forecasting, we also applied the method reported in [37], but the performance with five-fold CV did not give the same conclusion. Instead, it strongly supported to remain at $S = 1$. The detailed experiment is shown in Section 5.2. In order to find an alternative method to scale the range of the random weights, we propose a posteriori method called quantile scaling.

3.6. Re-distribution of the randomly weighted inputs

The purpose of the range scaling method discussed in the previous section is to avoid enhancement node saturation and to suppress the outliers in the input data. To achieve the same goal, we would like to constrain the weighted input data to the enhancement node so that majority of them fall into the non-linear region of the activation function of the enhancement node.

In this paper, we propose to apply a scaling function based on the quantiles of the weighted input data and the corresponding quantiles of the sigmoid function to re-distribute the randomly weighted input data. Since the proposed quantile scaling is implemented after random weights range determination, it is independent of the range of the random weights (a posteriori to the range determination). Another advantage is that the proposed quantile scaling does not require iterative training. The details of the proposed quantile scaling algorithm is as follows:

1. For each enhancement node, form an input vector: $\underset{N\times 1}{EN_k} = \{\sum_{m=1}^{M} w_{m,k} i_m^{(n)} + b_{i,k} |\ \forall n \in \{1 \ldots N\},\ \forall k \in \{1 \ldots K\}\}$, where N is the number of input (training) samples, M is the number of input features, $i_m^{(n)}$ is the mth feature of the nth input sample, $b_{i,k}$ is the input layer bias.
2. Obtain the j and l quantiles of the vector EN_k: Q_j and Q_l. In this paper, we set $j = 0.05$ and $l = 0.95$.
3. Obtain the j and l quantiles of the *logsig* function: $S_j = -ln(\frac{1}{j}-1)$, $S_l = -ln(\frac{1}{l}-1)$.
4. Scale EN_k according to: $\frac{(EN_k - Q_j)\times(S_i - S_j)}{Q_l - Q_j} + S_j$.

4. Assessment on generic time series data

In order to assess the performances of the eight configurations of RWSLFN/RVFL network to obtain generalized conclusions, we chose six generic TS datasets [26]. The statistics of the six datasets are summarized in Table 2.

There were ten assessments implemented for each dataset and these assessments were on 1–10 steps ahead forecasting. The performances were measured by nRMSE. For each TS, the first 70% was used for training and the remaining 30% was used for testing. The feature set is 'full' with length being 1/20 of the total length. The optimal number of enhancement nodes were determined by a five-fold TS-CV and the experiments were run for 30 repetitions. The input data were

Table 2
Summary of the six generic TS datasets [26], '1st Q' and '3rd Q' are the 1st quartile (25th percentile) and 3rd quartile (75th percentile), respectively, 'Sd' is the standard deviation.

Name	Length	Min	1st Q	Median	Mean	3rd Q	Max	Sd
Atmosfera	365	5.60	13.00	15.40	15.35	18.2	21.6	3.16
Bebida	187	49.63	77.44	91.97	92.66	109.8	134.6	20.35
Consumo	154	75.39	102.10	116.40	120.90	130.5	232.0	27.24
Fortaleza	149	468.00	1089.00	1399.00	1445.00	1752.0	2836.0	496.95
Ipi	187	65.81	91.21	103.30	103.60	115.4	148.3	18.28
Lavras	384	0.00	28.55	92.25	127.60	200.3	718.0	122.06

Table 3
Wilcoxon signed rank test statistics of the RWSLFNs with vs. without input layer bias on six generic TS data, alternative hypothesis H_a is 'two sided'. If $p < 0.5$ for Mx vs. My, Mx is better. If $p < 0.05$ (in bold) for Mx vs. My, Mx is statistically significantly better. If $p > 0.95$ (underlined) for Mx vs. My, My is statistically significantly better, $x, y \in \{1, \ldots, 8\}$. M1 ... M8 is defined in Table 1.

Comparison	Horizon (step)									
	1	2	3	4	5	6	7	8	9	10
M1 vs. M5	0.383	0.083	0.645	0.46	0.934	0.868	0.9	0.998	0.676	0.099
M2 vs. M6	0.309	0.76	0.493	0.064	0.131	**0.034**	**0.024**	0.37	0.972	0.335
M3 vs. M7	0.109	**0.048**	0.729	0.156	0.927	0.874	0.879	0.987	0.847	0.177
M4 vs. M8	0.373	0.76	0.417	0.124	0.135	**0.033**	**0.049**	0.424	0.989	0.398

Table 4
Wilcoxon signed rank test statistics of the RWSLFNs with vs. without hidden layer bias on six generic TS data, alternative hypothesis H_a is 'two sided'. If $p < 0.5$ for Mx vs. My, Mx is better. If $p < 0.05$ (in bold) for Mx vs. My, Mx is statistically significantly better. If $p > 0.95$ (underlined) for Mx vs. My, My is statistically significantly better, $x, y \in \{1, \ldots, 8\}$. M1 ... M8 is defined in Table 1.

Comparison	Horizon (step)									
	1	2	3	4	5	6	7	8	9	10
M1 vs. M3	0.267	0.098	**0.018**	0.727	0.914	0.194	0.114	0.259	**0.002**	**0.001**
M2 vs. M4	**0.002**	0.287	0.057	0.537	0.071	0.379	0.229	**0.001**	**0.014**	**0.076**
M5 vs. M7	**0.078**	**0.077**	0.184	**0.032**	0.111	0.194	0.132	**0.021**	**0.006**	0.201
M6 vs. M8	0.536	0.965	0.609	0.663	0.998	0.9	0.5	0.133	0.768	0.487

Table 5
Wilcoxon signed rank test statistics of the RWSLFNs with vs. without direct input–output connections on six generic TS data, alternative hypothesis H_a is 'two sided'. If $p < 0.05$ for Mx vs. My, Mx is statistically significantly better, otherwise, they are statistically comparable (underlined), $x, y \in \{1, \ldots, 8\}$. M1 ... M8 is defined in Table 1.

Comparison	Horizon (step)									
	1	2	3	4	5	6	7	8	9	10
M1 vs. M2	3.1e−06	1.6e−05	1.2e−07	3.7e−05	0.0082	0.00017	0.02	0.5	0.0011	0.011
M3 vs. M4	2.4e−06	0.0022	1.3e−06	3.3e−06	0.015	0.00031	0.056	0.68	0.006	0.056
M5 vs. M6	9.8e−05	0.037	1.2e−08	1.5e−05	3.6e−06	3e−06	3.7e−06	6.8e−05	0.0089	0.3
M7 vs. M8	0.0017	0.18	4.2e−08	0.00015	1.9e−06	9.9e−08	1.8e−05	0.0018	0.028	0.35

normalized to [0, 1] interval. The randomization uses a uniform distribution in $[-1, 1]$ following [31]. The random seeds are generated by a random seed array function '.Random.seed[1:30]' in 'R' [29]. Wilcoxon signed rank tests were employed to test the statistical difference and the p values were recorded in Tables 3–5. From the statistical tests, we can conclude that the input layer bias and hidden layer bias insignificantly affected the performance on TS forecasting because the p values hardly fall below 0.05 whereas the direct input–output connections significantly improved the performance on TS forecasting because the p values were much smaller than 0.05 in most of the cases as seen in Table 5.

5. Results and discussions on load demand forecasting

In this section, eight different variations of RWSLFN/RVFL network were applied to model and forecast the electricity load demand. Then the best of the eight configurations was compared with several reported statistical and machine learning methods for load demand forecasting.

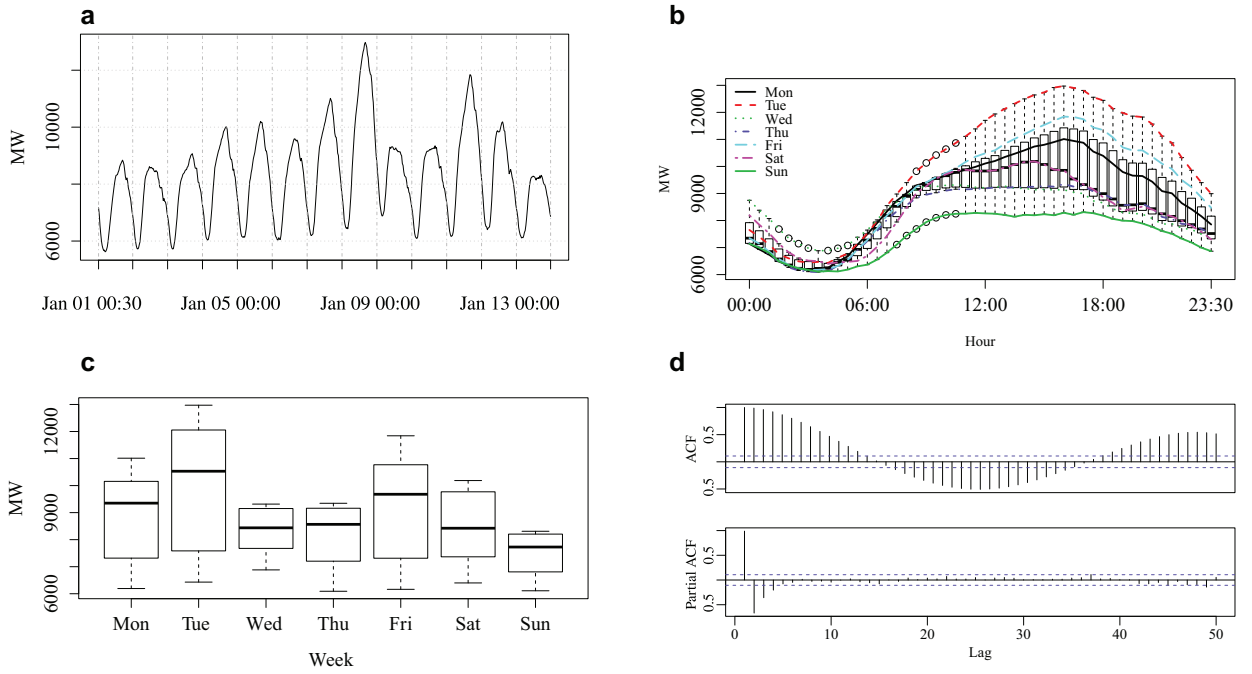

Fig. 3. Statistical characteristics of an AEMO load demand TS from 2013/01/01 to 2013/01/13, (a) plots of hourly load demand (b) boxplot of hourly load demand, (c) boxplot of daily load demand and (d) ACF and PACF of the time series.

5.1. Datasets

The data used for the experiments were retrieved from Australian Energy Market Operator (AEMO) [1]. The AEMO dataset contains half-hourly electricity load data (unit is MW) during 2013 recorded in New South Wales, Australia. A sample TS from 2013/01/07 to 2013/01/13 (a week) was used to show general statistical characteristics of the TS. Fig. 3b shows the half-hourly load demand for each day in that week together with the box-plot of the half-hourly load demand within the week. Fig. 3c shows the box-plot of the daily load demand of that week; and Fig. 3d shows the autocorrelation function (ACF) plot and PACF plot of the week. As shown in the hourly plot and the ACF, the load demand time series shows a significant seasonal characteristic with a period of 24 h. Therefore, we need to consider this seasonal component in the forecasting model.

In the experiments, the AEMO dataset was partitioned into 12 smaller monthly datasets and targeted for 1–24 h ahead forecasting. For each monthly dataset, the first 50% data were used for training and the remaining 50% was used for testing. The 'full' feature set contains 2 days' historical value. To average out the possible bias and variance introduced by randomization, each forecasting was repeated for 30 times. The code was written in 'R' [29]. The random seeds were generated by a random seed array function '.Random.seed[1:30]', and the random distribution was uniform in $[-1, 1]$.

5.2. Impact of the proposed quantile scaling on the load forecasting accuracy

As discussed in Section 3.5, there is a random weight range scaling method reported in [37] for classification. We use the same setup for TS forecasting: the random weights have a uniform distribution with range in $[-S, +S]$ for $S \in \{2^t | t = -5, -4.5, \ldots, 5\}$. Five-fold CV was used to determine the optimal enhancement node number and the optimal S value. To analyze the effect of S, the optimal enhancement node number was first obtained by find the minimum mean square error (MSE) averaged over five folds for each possibility of enhancement node number. Then the MSE is ranked with respect to the range of S to obtain the optimal S value. A MSE vs. S plot on M3 configuration for 1 h ahead forecasting is shown in Fig. 4. We can see that for 12 load demand data, the optimal S is at 1. This observation shows that random weights range other than $[-1, 1]$ can degrade the performance.

We then analyzed the characteristics of the randomly weighted data, which are the inputs to the enhancement node. For the load demand data, the inputs to the enhancement nodes make the *logsig* mapping skewed and the distribution has a peak near 1 as shown in Fig. 5.

In order to overcome the skewness problem, we proposed a quantile scaling algorithm to re-shape the distribution. The details of the quantile scaling algorithm is discussed in Section 3.6. We can see from Fig. 5 that the density of the scaled input vector after quantile scaling is more flat (evenly distributed). The performance of the RWSLFN/RVFL network with vs. without scaling is tested by a Wilcoxon signed rank test as shown in Table 6 (with 'full' feature set). It is shown that the

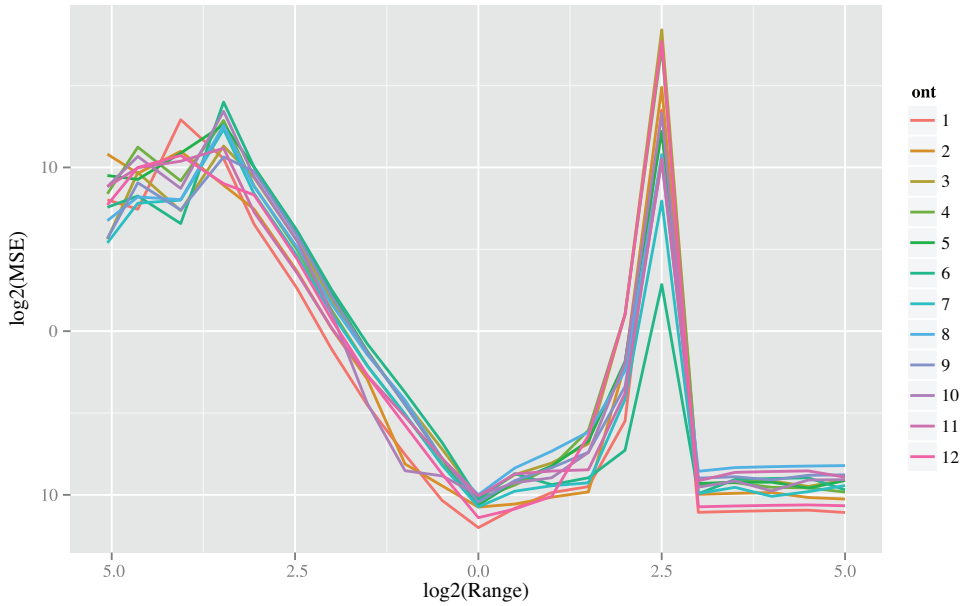

Fig. 4. Mean square error (log scaled) associated with Optimal Enhancement Node Number (log_2 scaled) vs. Range of Random Weights on M3 Configuration with 'full' feature set.

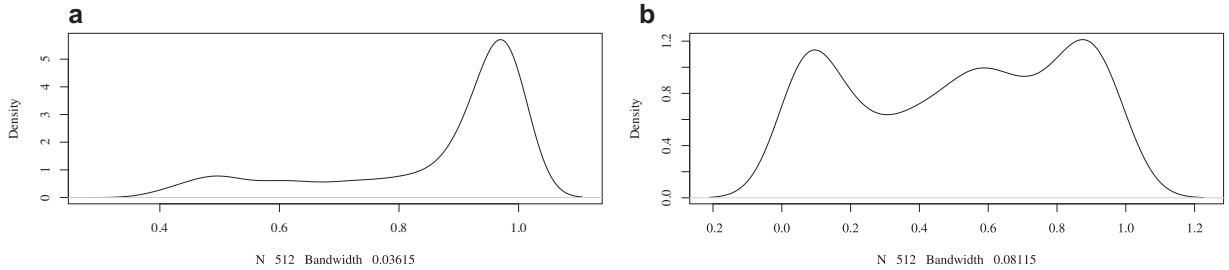

Fig. 5. The density plots of the weighted input vector to an enhancement node: (a) un-scaled and (b) scaled according to the proposed quantile scaling algorithm.

Table 6
p values of Wilcoxon signed rank tests on the eight configurations of RWSLFN/RVFL network with vs. without the proposed quantile scaling of the inputs to the enhancement nodes. If $p < 0.5$, quantile scaling is better. If $p < 0.05$ (in bold) quantile scaling is statistically significantly better. If $p > 0.95$ (underlined) without quantile scaling is statistically significantly better, M1 ... M8 is defined in Table 1.

Dataset	Horizon												
	1	2	4	6	8	10	12	14	16	18	20	22	24
M1	0.470	0.245	**0.050**	0.098	0.895	0.920	0.519	0.834	1.000	0.239	0.509	0.432	0.432
M2	**0.000**	**0.000**	**0.002**	0.157	0.209	0.058	0.135	0.239	0.214	0.192	**0.044**	**0.008**	**0.008**
M3	0.470	0.245	**0.050**	0.098	0.895	0.920	0.519	0.834	1.000	0.239	0.509	0.432	0.432
M4	**0.000**	**0.000**	**0.010**	0.592	0.379	0.192	0.245	0.614	0.871	0.774	0.135	**0.018**	**0.013**
M5	0.592	0.300	0.539	0.739	0.603	0.239	0.104	0.157	0.245	0.139	0.088	0.062	**0.041**
M6	**0.000**	**0.000**	**0.027**	0.062	**0.048**	**0.025**	**0.036**	**0.005**	**0.004**	**0.048**	**0.006**	**0.001**	**0.005**
M7	**0.017**	**0.025**	0.119	0.432	0.214	0.062	0.041	0.067	0.095	**0.021**	**0.008**	**0.002**	**0.001**
M8	**0.000**	**0.000**	**0.004**	**0.046**	**0.036**	**0.013**	0.056	0.499	0.252	0.423	0.286	0.171	0.171

proposed quantile scaling improved the performance of the RWSLFN/RVFL network for 1, 2, 4, 20, 22 and 24 h ahead while the improvement is marginal for 6–18 h ahead. Similar test statistics can be obtained with PACF and sAR selected feature sets. The proposed quantile scaling is applied in the subsequent experiments.

5.3. Enhancement node optimization

The number of enhancement nodes is optimized by a five-fold TS-CV algorithm. The median of the optimal number of enhancement nodes over the 30 trials is shown in Table 7. We can observe that there are much more enhancement nodes

Table 7
Optimal number of enhancement nodes for different RWSLFN configurations and feature sets over 12 months, median over 30 trials. M1 ... M8 is defined in Table 1.

Dataset	Full								PACF								sAR							
	M1	M2	M3	M4	M5	M6	M7	M8	M1	M2	M3	M4	M5	M6	M7	M8	M1	M2	M3	M4	M5	M6	M7	M8
1	3	73	3	70	3	68	4	69	4	12	4	13	5	9	5	11	4	6	4	8	4	5	4	6
2	5	82	5	84	5	81	2	82	76	78	76	82	46	57	46	53	3	5	3	6	2	5	2	6
3	14	75	14	74	20	70	26	74	74	36	82	48	18	6	50	9	2	2	2	3	1	2	2	3
4	2	82	2	78	2	86	1	83	1	6	1	7	1	7	1	8	7	2	5	3	6	7	2	8
5	31	90	31	90	30	87	31	88	52	53	52	60	60	54	60	50	1	3	1	4	2	3	3	5
6	9	90	9	92	18	91	2	91	78	70	78	68	76	78	75	82	1	2	1	3	1	2	2	3
7	41	92	41	93	44	88	47	88	22	29	22	30	21	32	20	32	2	7	2	8	1	6	2	8
8	1	93	1	92	1	93	2	92	51	54	51	55	42	52	44	51	14	15	14	15	18	20	16	20
9	5	80	5	82	4	75	4	78	30	42	30	42	35	49	40	48	1	4	1	6	1	4	1	6
10	4	78	4	76	2	82	2	82	52	54	52	45	4	10	3	14	3	4	3	5	4	4	1	5
11	9	82	9	82	8	81	10	82	73	77	73	78	68	74	71	78	8	14	8	15	10	13	11	14
12	3	81	3	79	3	81	4	84	4	8	4	9	3	9	4	10	6	12	6	12	5	12	6	12

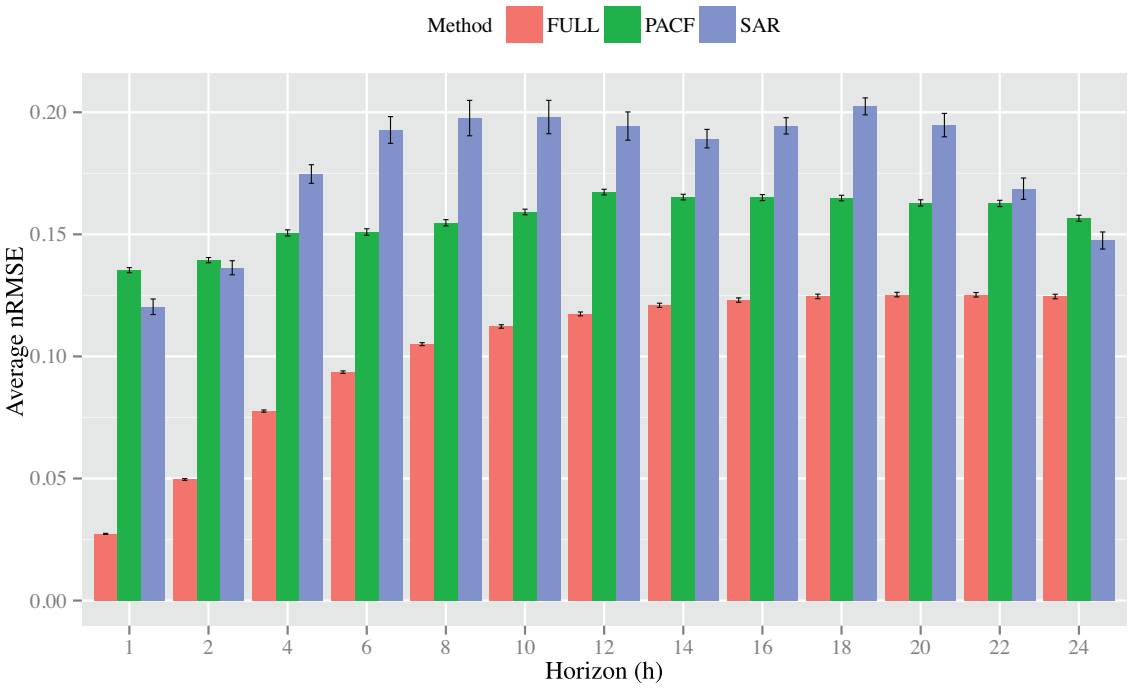

Fig. 6. Barplot of nRMSE vs. forecasting horizons of three different feature sets on M3 RWSLFN configuration.

required for configurations without direct input–output connections (M2, M4, M6 and M8) to obtain the optimal CV error than those with direct input–output connections (M1, M3, M5 and M7) for full feature set. The effects of input layer bias and hidden layer bias are in-significant on the optimal number of enhancement nodes. However, the optimal number of enhancement nodes of the RWSLFNs with the PACF or sAR feature sets is less distinguishable.

5.4. Comparison on feature sets

To evaluate the performance with respect to the feature sets, we plot the average nRMSE and the standard error for 1–24 h ahead forecasting on M3 configuration. The plot is shown in Fig. 6. We can see that the M3 with full feature set has the smallest nRMSE for all forecasting horizons, and the nRMSE is increasing with respect to the forecasting horizon. For PACF based feature selection and sAR based feature selection, the nRMSE vs. forecasting horizon is not monotonic, and it is always larger than the nRMSE with full feature set. Similar observations are made on other variants. Therefore, we can conclude that PACF or sAR feature selection on load forecasting degrades the performance. In the following comparisons, we will use RWSLFNs with full feature set.

Table 8
Average nRMSE of different RWSLFN configurations on load demand forecasting. M1 ... M8 is defined in Table 1.

	Dataset	M1	M2	M3	M4	M5	M6	M7	M8
Horizon = 1 h	1	0.018 ± 0	0.038 ± 0.005	0.018 ± 0	0.039 ± 0.006	0.018 ± 0	0.039 ± 0.004	0.018 ± 0	0.038 ± 0.005
	2	0.025 ± 0.003	0.046 ± 0.004	0.025 ± 0.003	0.046 ± 0.005	0.024 ± 0.001	0.044 ± 0.004	0.024 ± 0.002	0.045 ± 0.006
	3	0.028 ± 0.004	0.049 ± 0.01	0.028 ± 0.004	0.05 ± 0.007	0.028 ± 0.004	0.049 ± 0.008	0.029 ± 0.003	0.05 ± 0.006
	4	0.026 ± 0	0.044 ± 0.005	0.026 ± 0	0.043 ± 0.004	0.026 ± 0	0.042 ± 0.003	0.026 ± 0	0.042 ± 0.005
	5	0.041 ± 0.009	0.063 ± 0.008	0.041 ± 0.009	0.063 ± 0.008	0.04 ± 0.007	0.063 ± 0.011	0.04 ± 0.009	0.064 ± 0.01
	6	0.03 ± 0.004	0.052 ± 0.008	0.03 ± 0.004	0.052 ± 0.008	0.032 ± 0.006	0.051 ± 0.006	0.03 ± 0.004	0.051 ± 0.004
	7	0.032 ± 0.004	0.056 ± 0.007	0.032 ± 0.004	0.057 ± 0.007	0.033 ± 0.005	0.056 ± 0.007	0.032 ± 0.004	0.055 ± 0.006
	8	0.027 ± 0.001	0.049 ± 0.006	0.027 ± 0.001	0.049 ± 0.006	0.027 ± 0.001	0.048 ± 0.005	0.027 ± 0.001	0.048 ± 0.005
	9	0.027 ± 0	0.047 ± 0.005	0.027 ± 0	0.047 ± 0.005	0.027 ± 0.001	0.049 ± 0.008	0.027 ± 0	0.047 ± 0.004
	10	0.025 ± 0	0.051 ± 0.007	0.025 ± 0	0.049 ± 0.007	0.025 ± 0	0.05 ± 0.005	0.025 ± 0	0.051 ± 0.006
	11	0.026 ± 0.001	0.052 ± 0.007	0.026 ± 0.001	0.051 ± 0.007	0.027 ± 0.002	0.049 ± 0.007	0.026 ± 0.001	0.048 ± 0.007
	12	0.023 ± 0.002	0.05 ± 0.005	0.023 ± 0.002	0.05 ± 0.006	0.023 ± 0.001	0.05 ± 0.005	0.023 ± 0.001	0.049 ± 0.005
Horizon = 12 h	1	0.117 ± 0.005	0.141 ± 0.018	0.117 ± 0.005	0.142 ± 0.018	0.118 ± 0.007	0.142 ± 0.016	0.118 ± 0.005	0.14 ± 0.013
	2	0.113 ± 0.01	0.12 ± 0.006	0.113 ± 0.01	0.119 ± 0.006	0.112 ± 0.004	0.119 ± 0.006	0.111 ± 0.007	0.118 ± 0.006
	3	0.156 ± 0.024	0.161 ± 0.02	0.156 ± 0.024	0.163 ± 0.02	0.158 ± 0.022	0.164 ± 0.014	0.158 ± 0.022	0.163 ± 0.013
	4	0.086 ± 0.003	0.081 ± 0.004	0.086 ± 0.003	0.081 ± 0.004	0.086 ± 0.003	0.081 ± 0.005	0.089 ± 0.005	0.08 ± 0.005
	5	0.137 ± 0.02	0.109 ± 0.006	0.137 ± 0.02	0.11 ± 0.006	0.138 ± 0.022	0.112 ± 0.009	0.131 ± 0.016	0.113 ± 0.009
	6	0.109 ± 0.022	0.1 ± 0.007	0.109 ± 0.022	0.1 ± 0.006	0.111 ± 0.023	0.101 ± 0.007	0.108 ± 0.022	0.1 ± 0.007
	7	0.122 ± 0.016	0.129 ± 0.018	0.122 ± 0.016	0.123 ± 0.017	0.115 ± 0.015	0.128 ± 0.014	0.119 ± 0.019	0.123 ± 0.013
	8	0.102 ± 0.009	0.106 ± 0.008	0.102 ± 0.009	0.105 ± 0.008	0.099 ± 0.004	0.105 ± 0.007	0.098 ± 0.004	0.103 ± 0.005
	9	0.1 ± 0.003	0.096 ± 0.006	0.1 ± 0.003	0.096 ± 0.006	0.1 ± 0.004	0.097 ± 0.005	0.1 ± 0.005	0.097 ± 0.005
	10	0.115 ± 0.003	0.116 ± 0.005	0.115 ± 0.003	0.117 ± 0.007	0.115 ± 0.004	0.115 ± 0.004	0.116 ± 0.003	0.113 ± 0.005
	11	0.103 ± 0.006	0.109 ± 0.007	0.103 ± 0.006	0.108 ± 0.008	0.103 ± 0.005	0.106 ± 0.005	0.104 ± 0.006	0.105 ± 0.005
	12	0.149 ± 0.009	0.195 ± 0.013	0.149 ± 0.009	0.19 ± 0.015	0.148 ± 0.01	0.186 ± 0.013	0.145 ± 0.003	0.187 ± 0.015
Horizon = 24 h	1	0.12 ± 0.005	0.164 ± 0.034	0.12 ± 0.005	0.166 ± 0.032	0.122 ± 0.013	0.164 ± 0.028	0.119 ± 0.005	0.16 ± 0.024
	2	0.109 ± 0.014	0.131 ± 0.013	0.109 ± 0.014	0.13 ± 0.011	0.105 ± 0.006	0.135 ± 0.012	0.104 ± 0.005	0.131 ± 0.012
	3	0.173 ± 0.01	0.17 ± 0.012	0.173 ± 0.01	0.169 ± 0.011	0.173 ± 0.011	0.169 ± 0.011	0.174 ± 0.01	0.168 ± 0.01
	4	0.085 ± 0.002	0.085 ± 0.005	0.085 ± 0.002	0.085 ± 0.004	0.085 ± 0.002	0.085 ± 0.004	0.087 ± 0.003	0.084 ± 0.004
	5	0.153 ± 0.024	0.138 ± 0.012	0.153 ± 0.024	0.139 ± 0.014	0.149 ± 0.024	0.146 ± 0.012	0.14 ± 0.018	0.147 ± 0.013
	6	0.128 ± 0.012	0.129 ± 0.006	0.128 ± 0.012	0.128 ± 0.006	0.132 ± 0.015	0.129 ± 0.005	0.13 ± 0.014	0.128 ± 0.005
	7	0.119 ± 0.017	0.132 ± 0.012	0.119 ± 0.017	0.131 ± 0.011	0.12 ± 0.017	0.133 ± 0.012	0.115 ± 0.013	0.13 ± 0.012
	8	0.114 ± 0.013	0.139 ± 0.009	0.114 ± 0.013	0.138 ± 0.009	0.111 ± 0.008	0.134 ± 0.006	0.111 ± 0.01	0.133 ± 0.007
	9	0.104 ± 0.004	0.101 ± 0.006	0.104 ± 0.004	0.102 ± 0.006	0.105 ± 0.002	0.103 ± 0.007	0.104 ± 0.005	0.103 ± 0.006
	10	0.118 ± 0.004	0.124 ± 0.005	0.118 ± 0.004	0.125 ± 0.005	0.118 ± 0.003	0.122 ± 0.005	0.121 ± 0.004	0.122 ± 0.005
	11	0.11 ± 0.005	0.116 ± 0.005	0.11 ± 0.005	0.116 ± 0.005	0.11 ± 0.006	0.118 ± 0.006	0.111 ± 0.007	0.117 ± 0.005
	12	0.162 ± 0.022	0.255 ± 0.021	0.162 ± 0.022	0.248 ± 0.022	0.163 ± 0.028	0.243 ± 0.022	0.154 ± 0.009	0.242 ± 0.023

Table 9
Wilcoxon signed rank test statistics of the RWSLFNs with vs. without input layer bias on load demand forecasting, alternative hypothesis H_a is 'two sided'. If $p < 0.5$ for Mx vs. My, Mx is better. If $p < 0.05$ (in bold) for Mx vs. My, Mx is statistically significantly better. If $p > 0.95$ (underlined) for Mx vs. My, My is statistically significantly better, $x, y \in \{1, \ldots, 8\}$. M1 ... M8 is defined in Table 1.

Comparison	Horizon (h)												
	1	2	4	6	8	10	12	14	16	18	20	22	24
M1 vs. M5	0.351	0.274	0.531	0.144	0.065	0.184	0.337	0.351	0.204	0.456	0.513	0.862	0.329
M2 vs. M6	0.421	0.172	0.974	0.492	0.418	0.116	0.255	0.368	0.248	0.443	0.234	0.059	0.173
M3 vs. M7	0.977	0.517	0.873	0.955	0.05	0.054	0.067	**0.008**	**0.008**	**0.033**	**0.032**	0.126	0.112
M4 vs. M8	0.339	0.646	0.98	0.907	0.711	0.163	0.522	0.6	0.545	0.6	0.433	0.068	0.315

5.5. Performance evaluation among RWSLFNs

The nRMSE (selected forecasting horizons: 1, 12 and 24 h ahead) of the eight RWSLFNs averaged over 30 trials are tabulated in Table 8. The comparisons among different RWSLFNs based on the three aspects for the different configurations: input layer bias, hidden layer bias and direct input–output connections.

By comparing the RWSLFNs with and without input layer bias, we applied the paired Wilcoxon signed rank tests on the four pairs: M1 vs. M5, M2 vs. M6, M3 vs. M7 and M4 vs. M8. The p values are shown in Table 9 with the alternative hypothesis being 'two-sided'. As shown, there is no significant performance different between the pairs. We can conclude that the input layer bias did not influence the performance for the load demand forecasting. However, as discussed in [5], input bias is necessary for functional approximation due to the addition of even product terms. Hence, we recommend to keep the input layer bias in the RWSLFN.

Next, the paired Wilcoxon signed rank tests were applied on four pairs: M1 vs. M3, M2 vs. M4, M5 vs. M7 and M6 vs. M8, for assessing the influence of hidden layer bias. The p values are tabulated in Table 10 with the alternative hypothesis being

Table 10
Wilcoxon signed rank test statistics of the RWSLFNs with vs. without hidden layer bias on load demand forecasting, alternative hypothesis H_a is 'two sided'. If $p < 0.5$ for Mx vs. My, Mx is better. If $p < 0.05$ (in bold) for Mx vs. My, Mx is statistically significantly better. If $p > 0.95$ (underlined) for Mx vs. My, My is statistically significantly better, $x, y \in \{1, \ldots, 8\}$. M1 ... M8 is defined in Table 1.

Comparison	Horizon (h)												
	1	2	4	6	8	10	12	14	16	18	20	22	24
M1 vs. M3	0.376	0.174	0.249	0.772	0.387	0.186	0.827	0.671	0.619	0.291	0.386	0.968	0.061
M2 vs. M4	**0.032**	0.236	0.609	0.468	0.176	0.069	**0.013**	**0.02**	**0.023**	0.499	0.352	0.181	0.357
M5 vs. M7	0.107	**0.036**	0.86	0.407	0.582	0.22	0.122	**0.007**	**0.008**	0.042	**0.037**	0.09	0.237
M6 vs. M8	0.244	0.977	0.934	0.083	0.254	**0.041**	0.364	0.28	0.158	0.249	0.101	0.063	0.963

Table 11
Wilcoxon signed rank test statistics of the RWSLFNs with vs. without direct input–output connections on load demand forecasting, alternative hypothesis H_a is 'two sided'. If $p < 0.05$ for Mx vs. My, Mx is statistically significantly better, otherwise, they are statistically comparable (underlined), $x, y \in \{1, \ldots, 8\}$. M1 ... M8 is defined in Table 1.

Comparison	Horizon (h)												
	1	2	4	6	8	10	12	14	16	18	20	22	24
M1 vs. M2	3.4e−57	1.2e−29	0.00011	0.13	0.16	0.062	0.025	0.0018	0.00065	0.00026	2.5e−05	3.3e−06	5.7e−09
M3 vs. M4	1.8e−57	3.3e−31	9.2e−05	0.18	0.22	0.14	0.056	0.0037	0.0011	0.00058	4.8e−05	3.9e−06	7e−10
M5 vs. M6	4.2e−56	1.5e−32	0.0019	0.22	0.76	0.86	0.61	0.23	0.12	0.017	0.00023	8.3e−05	4.4e−06
M7 vs. M8	1e−57	2.4e−39	6.3e−06	0.046	0.49	0.72	0.57	0.48	0.32	0.083	0.00057	0.00012	3.3e−08

Table 12
Wilcoxon signed rank test statistics of RVFL network vs. two-stage method on load demand forecasting over the 1–24 h ahead forecasting horizons, alternative hypothesis H_a is 'less than', $p < 0.05$ means significant outperformance.

1	2	4	6	8	10	12	14	16	18	20	22	24
1.1e−19	3.5e−20	1.2e−09	0.0071	0.048	0.064	0.035	0.008	0.0015	7.5e−05	5.4e−09	6.2e−14	1.4e−17

'two sided'. It is shown that for M2 vs. M4 and M5 vs. M7, the performance differences are significant on four forecasting horizons whereas for M1 vs. M3 or M6 vs. M8, the differences are insignificant (none and one case, respectively). We can conclude that the hidden layer bias has little impact on the performance of load demand forecasting.

The influence of direct input–output connections was assessed by comparing the performances of M1 vs. M2, M3 vs. M4, M5 vs. M6 and M7 vs. M8. The alternative hypothesis of the paired Wilcoxon signed rank tests is 'less than'. It is shown in Table 11 that the direct input–output connections significantly improved the performance of RWSLFN for 1–24 h ahead forecasting except for some cases (6–10 h ahead for M1 vs. M2, 6–12 h ahead for M3 vs. M4, 6–16 h ahead for M5–M6 and 8–18 h ahead for M7 vs. M8).

Based on the above-mentioned observations and implications, we can see that the RVFL network [27] has the overall best performance: with input layer bias, without hidden layer bias and with direct input–output connections. We will focus on analyzing the RVFL network.

In the RVFL network, the outputs from the enhancement nodes (non-linear) and the direct input–output connections (linear) are optimized together by a least square estimation. To examine the effectiveness of the optimization strategy of the RVFL network, a two-stage method is constructed and compared with it. The two-stage method is to first train a time delayed finite impulse response (FIR) filter for the linear output part and then train a RWSLFN without direct input–output connections for the non-linear output part. Finally the outputs from the two parts are aggregated by a least square estimation.

The Wilcoxon signed rank test statistics are shown in Table 12. It is shown that the RVFL network outperformed the two-stage method for all forecasting horizons except for 12 h ahead forecasting. The possible reason for the outperformance is that the two stage optimization has introduced independent error from each stage, and the error cannot be attenuated during the final aggregation. On the other hand, in the single stage optimization (RVFL network), the least square estimation takes into consideration both outputs with a single minimization goal, which eliminates the possibility of error introduced between the overhead. Therefore, based on simplicity and accuracy, the RVFL network is preferable to the two-stage method.

5.6. Performance evaluation of the RVFL network against reported methods

The RVFL network [27] was compared against some commonly used methods: persistence, seasonal autoregressive integrated moving average (sARIMA), support vector regression (SVR) [10,12,18,22], ANN [23] and random forests (RF) [4,7,11,24]. The SVR is an ϵ-SVR with Radial basis function (RBF) kernel; the ANN is an SLFN with *logsig* activation function; and the RF is with 500 trees. All of them were trained with the five-fold TS-CV. The optimal parameters of the methods are shown in Table 13. The nRMSE of the five methods and the RVFL methods on 1, 12 and 24 h ahead forecasting are shown in Table 14.

Table 13
Optimal parameters of sARIMA, ANN, SVR and RF.

Dataset	$sARIMA(p, d, q) \times (P, D, Q)[s]$	ϕ_p	θ_q	Φ_P	Θ_Q	b
1	(1, 1, 1) × (1, 0, 1)[48]	0.886	−0.353	0.935	−0.438	
2	(3, 0, 3) × (1, 0, 1)[48]	0.835,0.751,-0.617	0.707,0.119 0.269	1	−0.966	0.365
3	(1, 1, 0) × (0, 1, 1)[48]	0.55			−0.687	
4	(1, 1, 1) × (1, 1, 0)[48]	−0.169	0.416	−0.12		
5	(2, 0, 2) × (1, 0, 1)[48]	1.531,−0.549	0.011,0.207	0.959	−0.399	
6	(2, 1, 1) × (1, 1, 1)[48]	−0.133,0.646	0.797	−0.944	0.786	
7	(4, 0, 4) × (1, 0, 1)[48]	1.287,−1.007,1.339,−0.651	−0.059, 1.087,−0.359,0.291	0.99	−0.397	
8	(1, 1, 0) × (1, 0, 1)[48]	0.665		0.907	−0.29	
9	(3, 1, 1) × (1, 1, 1)[48]	1.219,−0.273,−0.145	−0.707	0.359	−0.72	
10	(2, 1, 0) × (1, 1, 1)[48]	0.333,0.209		−0.383	−0.279	
11	(3, 1, 5) × (1, 1, 1)[48]	0.05,−0.803,−0.147	0.389,1.368, 0.911,0.51,0.392	−0.097	−1	
12	(5, 1, 0) × (1, 1, 1)[48]	0.36,0.279,0.106,−0.2,−0.003		0.027	−0.321	

Dataset	1	2	3	4	5	6	7	8	9	10	11	12
ANN n_h =	96	120	144	144	120	120	120	96	168	192	120	144
SVR C =	100	1	1	1	1	100	1	1	1	0.1	1	1
ϵ =	0.1	0.1	0.01	0.1	0.1	0.01	0.1	0.1	0.1	0.01	0.1	0.01

RF $mtry = 32$, $ntree = 500$

$(p, d, q) \times (P, D, Q)[s]$ are the orders of sARIMA, ϕ_p, θ_q, Φ_P, Θ_Q, $b(intercept)$ are the coefficients of sARIMA, n_h is the optimal number of hidden neurons of ANN, C and ϵ are the optimal cost factor and insensitive parameter of SVR, *mtry* and *ntree* are the number of variables randomly sampled at each split and number of trees to grow of RF, respectively.

Table 14
nRMSE of persistence (Per.) sARIMA, ANN, SVR, RF and RVFL network on load demand forecasting.

	Horizon = 1 h						Horizon = 12 h						Horizon = 24 h					
	Per.	sARIMA	ANN	SVR	RF	RVFL	Per.	sARIMA	ANN	SVR	RF	RVF	Per.	sARIMA	ANN	SVR	RF	RVFL
1	0.16	0.218	0.134	0.056	0.044	0.018	0.16	0.245	0.138	0.114	0.108	0.117	0.161	0.265	0.34	0.15	0.118	0.12
2	0.121	0.19	0.042	0.063	0.058	0.025	0.121	0.153	0.195	0.128	0.114	0.113	0.124	0.14	0.188	0.132	0.113	0.109
3	0.118	0.172	0.088	0.069	0.047	0.028	0.118	0.194	0.146	0.215	0.132	0.156	0.115	0.246	0.256	0.165	0.177	0.173
4	0.113	0.178	0.049	0.04	0.045	0.026	0.108	0.18	0.083	0.114	0.076	0.086	0.098	0.204	0.082	0.074	0.075	0.085
5	0.103	0.181	0.097	0.09	0.083	0.041	0.103	0.133	0.128	0.135	0.104	0.137	0.104	0.136	0.196	0.126	0.118	0.153
6	0.101	0.194	0.059	0.073	0.07	0.03	0.099	0.189	0.152	0.103	0.098	0.109	0.098	0.208	0.167	0.145	0.116	0.128
7	0.095	0.195	0.081	0.049	0.05	0.032	0.091	0.134	0.087	0.096	0.081	0.122	0.091	0.133	0.113	0.1	0.086	0.119
8	0.121	0.216	0.07	0.067	0.053	0.027	0.121	0.234	0.13	0.115	0.097	0.102	0.123	0.256	0.154	0.152	0.116	0.114
9	0.122	0.2	0.076	0.046	0.056	0.027	0.127	0.204	0.129	0.115	0.096	0.1	0.135	0.222	0.117	0.092	0.097	0.104
10	0.139	0.217	0.049	0.064	0.04	0.025	0.139	0.233	0.237	0.12	0.108	0.115	0.139	0.283	0.202	0.129	0.114	0.118
11	0.12	0.194	0.098	0.054	0.044	0.026	0.126	0.21	0.175	0.105	0.098	0.103	0.131	0.245	0.234	0.108	0.115	0.11
12	0.152	0.188	0.17	0.073	0.059	0.023	0.152	0.199	0.205	0.14	0.147	0.149	0.149	0.235	0.207	0.188	0.151	0.162

The Friedman ranked sum test showed that there are significant performance differences among the six different methods. A post-hoc Nemenyi test was implemented to distinguish them. The plot of the Nemenyi test is shown in Fig. 7. It is shown that RF, SVR and RVFL network are among the better performing methods whereas ANN and sARIMA are among the poorer performed methods. For certain cases, the persistence method also have good accuracy because that the load demand TS is highly periodic. A Wilcoxon signed rank tests are plotted in Fig. 8 with an additional normalized mean absolute error (nMAE) error metric shown in Eq. (9).

$$\mathrm{nMAE} = \frac{1}{y_{\max} - y_{\min}} \mathrm{E}[|\hat{y} - y|] \tag{9}$$

From the plot we can see that the RVFL network significantly outperformed sARIMA and ANN and had slightly better performance as SVR (outperformance on 2, 4, and 6 h ahead forecasting). The RF had several cases that outperformed the RVFL network but on the other hand the RVFL network also outperformed the RF method on several forecasting horizons. Results again showed the relatively high accuracy of the persistence method compared with sARIMA and ANN due to the characteristics of the load demand TS.

The overall rank of the methods on the load demand forecasting across the 1–24 h forecasting horizons are tabulated in Table 15. It is shown that the RVFL network has the second best performance ranking and it is the best performing method among the non-ensemble methods. If we consider that the critical distance (0.51), the RVFL network and SVR have comparable performances with 95% confident. The RF has the significantly best ranking but it is an ensemble method.

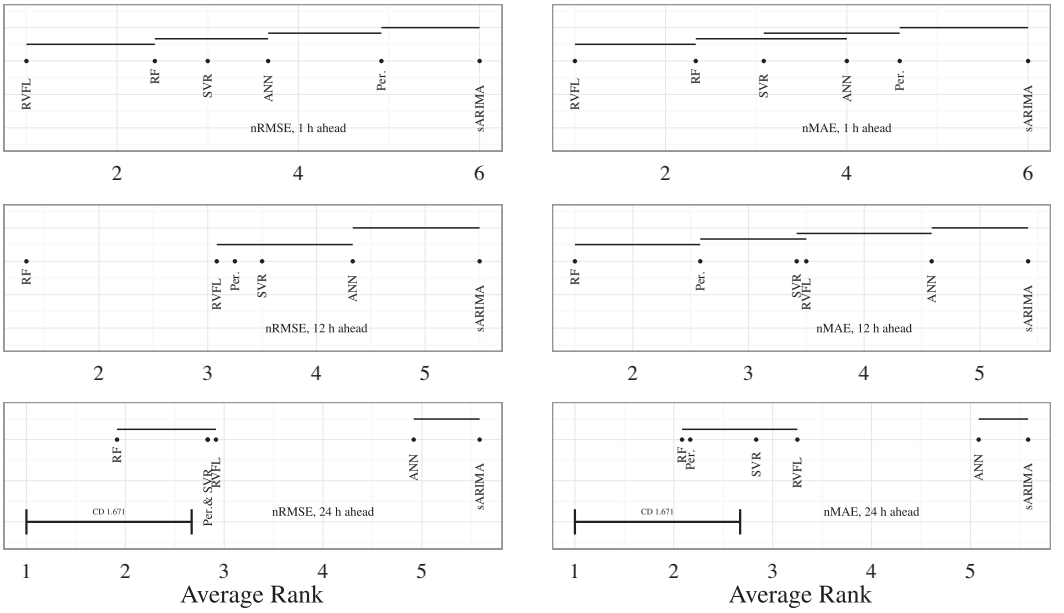

Fig. 7. Nemenyi test based on nRMSE/nMAE among the methods across the 1, 12 and 24 h ahead forecasting horizons: similar performances (95% confidence) are linked by a segment whose length is shorter than the critical distance (CD=1.671). 'Per.' stands for the persistence method.

Fig. 8. Wilcoxon signed rank test based on nRMSE/nMAE among the methods across the 1–24 h ahead forecasting horizons: significantly outperforming (*y* axis vs. *x* axis) cases (95% confidence) are marked on the figure.

Table 15
Overall rank of the forecasting methods on the 12 load demand TS, with respect to the error measures, the critical distance CD=0.51.

Error Measure	RF	RVFL	SVR	Per.	ANN	sARIMA
nRMSE	1.89	2.64	2.85	3.57	4.14	5.89
nMAE	1.91	2.86	2.94	2.99	4.38	5.89

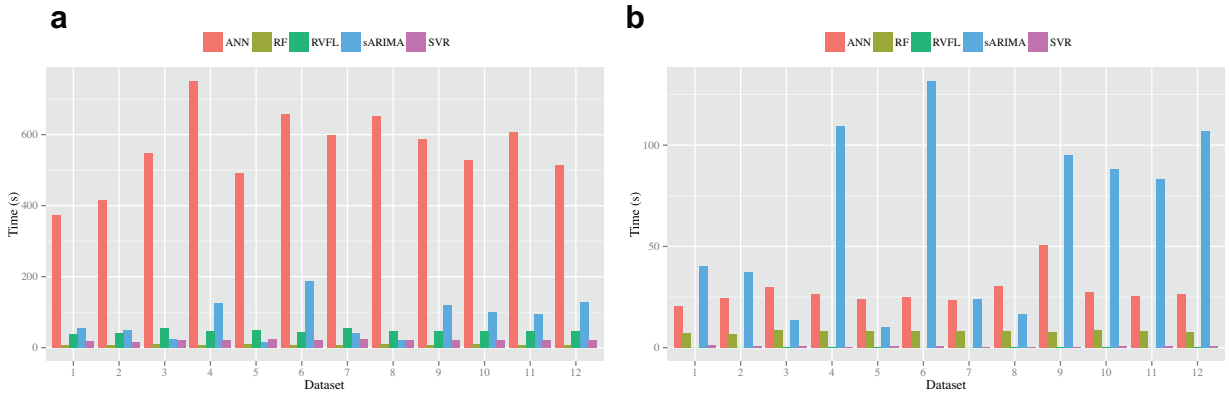

Fig. 9. (a) Training time and (b) testing time of the forecasting methods on 12 datasets for 1 h ahead forecasting.

5.7. Computation time comparison

It is worthwhile to note that the computational speed of the RVFL network is superior than ANN and SVR because the ANN and SVR require repetitive parameter tuning such as grid search on c and ϵ for SVR and the BP to iteratively tune the weights for ANN. In the testing phase the computation time of the RVFL network is also shorter because it only requires non-iterative matrix manipulation. The training and testing times of all the methods are plotted in Fig. 9. It is shown that the RVFL network has a shorter training time than sARIMA and ANN, and a comparable training time as SVR and RF. However, the RVFL network has the shortest testing time among the six methods. We can conclude that the RVFL network has better performance by considering both accuracy and computation time.

6. Conclusion

Random weight single hidden layer neural networks (RWSLFN) have been employed for short term time series forecasting. Eight configurations of RWSLFN have been formulated and compared on six generic time series datasets as well as twelve load demand datasets. From the comparison results, the following conclusions are made:

- The quantile scaling algorithm has improved the performance for 1–4 h and 18–24 h ahead forecasting horizons.
- Feature selection based on partial auto correlation function or seasonal auto-regression has consistently degraded the performance on the seasonal time series.
- The direct input–output connections have boosted the RWSLFN to significantly better performance implying RWSLFN with direct input–output connections (known as random vector functional link network or RVFL network) is the preferred for time series forecasting. Hence, removing direct links in RVFL is detrimental as these direct links emulate the finite impulse response filter (FIR).
- Empirical studies in this paper and in [37] showed no clear overall advantage of input layer and hidden layer biases. However, as discussed in [5], the input layer biases are necessary for the neural networks to function properly as a universal approximator. Hence, we recommend retaining biases as the selection choices as they may be beneficial for some forecasting problems.
- Compared with reported non-ensemble forecasting methods such as the persistence method, seasonal ARIMA and artificial neural networks, the RVFL network has significantly better performance. The RVFL network has comparable performance as SVR but it is underperformed by random forest, which is an ensemble method.
- The computation time (including training and testing) of the RVFL network is the shortest compared with the reported methods.

Regarding future research directions, ensemble methods such as bagging and boosting can be applied together with the RVFL network. As ensemble methods usually require training using multiple datasets in parallel (such as bagging) or in sequence (such as boosting), the computation time is usually very long. With the advantage of the RVFL network, the training time can be significantly reduced. Hence, thorough study on Ensemble RVFL network is a promising future research direction. Future research may also investigate other types of TS such as wind power (without seasonal component), financial data (strong exogenous components/external factors) and additional non-linear, non-stationary TS.

Acknowledgments

This work was supported by the Singapore National Research Foundation (NRF) under its Campus for Research Excellence And Technological Enterprise (CREATE) program, and Cambridge Advanced Research Centre in Energy Efficiency in Singapore (CARES), C4T project.

Authors thank the managing Guest Editor Associate Professor Dianhui Wang for suggesting us to investigate the scaling of randomization.

The author Ye Ren would also like to thank the Clean Energy Program Office (CEPO) for scholarship.

References

[1] Australian Energy Market Operator, 2015.
[2] H.K. Alfares, M. Nazeeruddin, Electric load forecasting: literature survey and classification of methods, Int. J. Syst. Sci. 33 (1) (2002) 23–34.
[3] M. Alhamdoosh, D. Wang, Fast decorrelated neural network ensembles with random weights, Inf. Sci. 264 (2014) 104–117.
[4] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[5] C.L.P. Chen, A rapid supervised learning neural network for function interpolation and approximation, IEEE Trans. Neural Netw. 7 (5) (1996) 1220–1230.
[6] C.L.P. Chen, J.Z. Wan, A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 29 (1) (1999) 62–72.
[7] Y.-Y. Cheng, P. Chan, Z.-W. Qiu, Random forest based ensemble system for short term load forecasting, in: Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC2012), vol. 1, 2012, pp. 52–56, doi:10.1109/ICMLC.2012.6358885.
[8] H. Cruse, Neural Networks as Cybernetic Systems, Thieme Stuttgart, 1996.
[9] M. De Felice, X. Yao, Short-term load forecasting with neural network ensembles: a comparative study [application notes], IEEE Comput. Intell. Mag. 6 (3) (2011) 47–56.
[10] H. Drucker, C.J. Burges, L. Kaufman, A. Smola, V. Vapnik, Support vector regression machines, Adv. Neural Inf. Process. Syst. 9 (1997) 155–161.
[11] G. Dudek, Short-term load forecasting using random forests, in: D. Filev, J. Jabłkowski, J. Kacprzyk, M. Krawczak, I. Popchev, L. Rutkowski, V. Sgurev, E. Sotirova, P. Szynkarczyk, S. Zadrozny (Eds.), Intelligent Systems'2014, Advances in Intelligent Systems and Computing, vol. 323, Springer International Publishing, 2015, pp. 821–828, doi:10.1007/978-3-319-11310-4_71.
[12] E.E. Elattar, J. Goulermas, Q.H. Wu, Electric load forecasting based on locally weighted support vector regression, IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 40 (2010) 438–447.
[13] I. Esener, T. Yuksel, M. Kurban, Artificial intelligence based hybrid structures for short-term load forecasting without temperature data, in: Proceedings of the International Conference on Machine Learning and Applications (ICMLA2012), FL, US, vol. 2, 2012, pp. 457–462.
[14] G.F. Fan, S. Qing, H. Wang, W.C. Hong, H.J. Li, Support vector regression model based on empirical mode decomposition and auto regression for electric load forecasting, Energies 6 (2013) 1887–1901.
[15] R. Ghazali, A. Hussain, D. Al-Jumeily, P. Lisboa, Time series prediction using dynamic ridge polynomial neural networks, in: Proceedings of the International Conference on Developments in eSystems Engineering (DESE2009), 2009, pp. 354–363.
[16] L. Ghelardoni, A. Ghio, D. Anguita, Energy load forecasting using empirical mode decomposition and support vector regression, IEEE Trans. Smart Grid 4 (1) (2013) 549–556.
[17] C. Guan, P.B. Luh, L.D. Michel, Y. Wang, P.B. Friedland, Very short-term load forecasting: wavelet neural networks with data pre-filtering, IEEE Trans. Power Syst. 28 (1) (2013) 30–31.
[18] Y.-C. Guo, D. xiao Niu, Y.-X. Chen, Support vector machine model in electricity load forecasting, in: Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC2006), 2006, pp. 2892–2896, doi:10.1109/ICMLC.2006.259076.
[19] H.S. Hippert, C.E. Pedreira, R.C. Souza, Neural networks for short-term load forecasting: a review and evaluation, IEEE Trans. Power Syst. 16 (2001) 44–55.
[20] R.-A. Hooshmand, H. Amooshahi, M. Parastegari, A hybrid intelligent algorithm based short-term load forecasting approach, Int. J. Electr. Power Energy Syst. 45 (2013) 313–324.
[21] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[22] A. Kavousi-Fard, H. Samet, F. Marzbani, A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting, Expert Syst. Appl. 41 (2014) 6047–6056.
[23] C.-N. Ko, C.-M. Lee, Short-term load forecasting using SVR (support vector regression)-based radial basis function neural network with dual extended Kalman filter, Energy 49 (2013) 413–422.
[24] A. Lahouar, J.B.H. Slama, Day-ahead load forecast using random forest and expert input selection, Energy Convers. Manage. 103 (2015) 1040–1051, doi:10.1016/j.enconman.2015.07.041.
[25] M. Moazzami, A. Khodabakhshian, R. Hooshmand, A new hybrid day-ahead peak load forecasting method for Iran's national grid, Appl. Energy 101 (2013) 489–501.
[26] P.A. Morettin, C. Toloi, Análise de séries temporais, Blucher, 2006.
[27] Y.-H. Pao, G.-H. Park, D.J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.
[28] D.-C. Park, A time series data prediction scheme using bilinear recurrent neural network, in: Proceedings of the IEEE International Conference on Information Science and Applications (ICISA2010), Seoul, 2010, pp. 1–7.
[29] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2015.
[30] Y. Ren, P.N. Suganthan, N. Srikanth, A comparative study of empirical mode decomposition-based short-term wind speed forecasting methods, IEEE Trans. Sustainable Energy 6 (1) (2015) 236–244.
[31] W.F. Schmidt, M.A. Kraaijveld, R.P. Duin, Feedforward neural networks with random weights, in: Proceedings of the IAPR International Conference on Pattern Recognition Conference B: Pattern Recognition Methodology and Systems, 1992, pp. 1–4.
[32] J.W. Taylor, Short-term load forecasting with exponentially weighted methods, IEEE Trans. Power Syst. 27 (1) (2012) 458–464.
[33] J.W. Taylor, P.E. McSharry, Short-term load forecasting methods: an evaluation based on European data, IEEE Trans. Power Syst. 22 (4) (2007) 2213–2219.
[34] L. Wang, X. Fu, Data Mining with Computational Intelligence, Springer, Berlin, 2005.
[35] L.P. Wang, S. Gupta, Neural networks and wavelet de-noising for stock trading and prediction, in: W. Pedrycz, S.M. Chen (Eds.), Time Series Analysis, Modeling and Applications, Springer, 2013, pp. 229–247.
[36] L.P. Wang, K. Teo, Z. Lin, Predicting time series with wavelet packet neural networks, in: Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN2001), 2001, pp. 1593–1597.
[37] L. Zhang, P.N. Suganthan, A comprehensive evaluation of random vector functional link networks, Inf. Sci. (2015), doi:10.1016/j.ins.2015.09.025.