EMD-Att-LSTM: A Data-driven Strategy Combined with Deep Learning for Short-term Load Forecasting

Neeraj, Jimson Mathew, and Ranjan Kumar Behera

***Abstract*—Electric load forecasting is an efficient tool for system planning, and consequently, building sustainable power systems. However, achieving desirable performance is difficult owing to the irregular, nonstationary, nonlinear, and noisy nature of the observed data. Therefore, a new attention-based encoder-decoder model is proposed, called empirical mode decomposition-attention-long short-term memory (EMD-Att-LSTM). EMD is a data-driven technique used for the decomposition of complex series into subsequent simpler series. It explores the inherent properties of data to obtain the components such as trend and seasonality. Neural network architecture driven by deep learning uses the idea of a fine-grained attention mechanism, that is, considering the hidden state instead of the hidden state vectors, which can help reflect the significance and contributions of each hidden state dimension. In addition, it is useful for locating and concentrating the relevant temporary data, leading to a distinctly interpretable network. To evaluate the proposed model, we use the repository dataset of Australian energy market operator (AEMO). The proposed architecture provides superior empirical results compared with other advanced models. It is explored using the indices of root mean square error (RMSE) and mean absolute percentage error (MAPE).**

***Index Terms*— Short-term load forecasting, Australian energy market operator, long short-term memory (LSTM), empirical mode decomposition (EMD), attention mechanism.**

I. Introduction

In past years, many studies have been conducted on different techniques for time-series forecasting. The models established using different methods are grouped into linear, nonlinear, ensemble, and deep learning ones.

Linear functions are used in linear forecasting models. Some of the prominent techniques of linear models are exponential smoothing, linear regression (LR), autoregressive integrated moving average (ARIMA), Holt-Winters, and other derived techniques such as seasonal autoregressive moving average (SARIMA). ARIMA is the most widely-used model for linear time-series forecasting. Moreover, solving nonlinear problems using linear forecasting models is challenging as linear models use linear relationships between the actual and forecasted data.

Indeed, nonlinear forecasting models require a nonlinear function for predicting load demands. These models are effective in learning the complex load behaviors. Hence, they can be used for electric load forecasting. With the advances in computing resources and computing power, nonlinear models have garnered significant attention. Some widely-used nonlinear models are support vector machine (SVM) [1], artificial neural networks (ANNs) [2], fuzzy techniques [3], and genetic algorithm (GA) [4].

In particular, a data-driven approach is used in ANNs, and the data analysis is performed using limited prior knowledge regarding the relationship between input and output data. These models perform like the human brain as they learn dependencies and patterns from the existing data and predict future results. Some important ANN models are the generalized regression neural network [5] and multilayer perceptron [6].

Reference [7] demonstrates a hierarchical ANN approach for a 15-min-ahead forecasting. They implement five neural networks (NNs) to observe different periods of 24-hour duration for predicting a day-wise electric load. The results of these five NNs are subsequently combined using another ANN.

Reference [8] implements a technique based on wavelet transform and NN for 1-hour- to 24-hour-ahead load forecasting of North American data. In this technique, the load values are first decomposed into various components by applying the wavelet transform, and then the final forecasting is made using the NN-based approach. Reference [9] proposes a hybrid model comprising of the deep belief network (DBN) and LR models to forecast the future values of time series. After fitting the LR model to the original data, the LR model residuals are used as the extra nonlinear inputs for the DBN model.

Nonetheless, using these models has disadvantages such as the output getting stuck to local minima during optimization, hyper-parameter tunning, and appropriate kernel selection. Ensemble models are a combination of linear and nonlinear models; hence, they combine the advantages and overcome the limitations of linear and nonlinear models to predict results accurately. Most ensemble models combine classical statistical models, machine-learning models, and decomposition techniques. Reference [10] introduces an ensemble deep learning methodology to predict time series, which combines the results of different DBNs using the SVR model. Reference [11] presents an EMD-DBN-based method for

Manuscript received: August 20, 2020; revised: January 6, 2021; accepted: June 10, 2021. Date of CrossCheck: June 10, 2021. Date of online publication: September 17, 2021.

Neeraj (corresponding author), J. Mathew, and R. K. Behera are with Indian Institute of Technology, Patna, 801103, India (e-mail: neeraj.pcs17@iitp.ac.in; jimson@iitp.ac.in; rkb@iitp.ac.in).

DOI: 10.35833/MPCE.2020.000626

short-term load forecasting (STLF), which uses two restricted Boltzmann machines (RBMs) and DBN to forecast the different intrinsic mode functions (IMFs) acquired from EMD.

Reference [12] proposes a methodology that combines variational mode decomposition (VMD) and extreme learning machine (ELM) for STLF. First, VMD is used to decompose the original time series and remove the skewness. Subsequently, the obtained modes are used for forecasting by ELM enhanced with the differential evolution algorithm.

Ensemble models can achieve high accuracies. However, the network architecture used by these models is completely specific and complex. In addition, they are not well-equipped to handle long-term dependency problems. These models require a defined approach to address the issue of convergence to local minima. Table I shows the advantages, disadvantages, and similarities or dissimilarities among the linear, nonlinear, and ensemble models. Additionally, it lists some sample models.

TABLE I
COMPARISON OF SEVERAL POPULAR TIME SERIES PREDICTION MODELS

Model	Example	Methodology	Advantages and Disadvantages
Linear models	Moving average (MA) & exponential smoothing (ES) [13]	MA and ES methods are applied for STLF. The data are collected for Universiti Teknologi PETRONAS (UTP), Malaysia, which are divided into two categories, i. e., semester on (SON) and semester off (SOF). The load values for the year 2010 are analyzed to forecast the load for the year 2011.	1) Advantages: ① a linear function is used to predict the future values of electric load series ② implementation is easy ③ less data are required for training 2) Disadvantages: ① the forecasting accuracy is not high ② it is insufficient to represent the non-linear behaviour of load series
	ARIMA [14]	ARIMA model is used to predict monthly load. Five-year historical data are used to forecast the sixth-year data. The model is a multiplicative combination of seasonal and non-seasonal patterns. The order for the model used is ARIMA(1, 1, 0) (1, 1, 0).	
	Box-Jenkins ARIMA [15]	A variant of ARIMA models, i.e., Box-Jenkins ARIMA, is used to forecast residential load in Greece. Fifty-year monthly and quarterly data are used for the analysis with a total of 180 samples. 156 samples are used to train the model and the remaining 24 observations are used to test the accuracy of the model.	
	ARIMA with an external/exogenous input (ARIMAX) [16]	ARIMAX model is used to forecast the power load for the commercial building. Occupancy data are used as an external feature to improve the performance of the model. The hourly data for 79 days are recorded, out of which for 5 days are for network login data and for 17 complete days are missing. The missing values are handled by imputting the average of the non-missing values for that hour and week.	
	SARIMA [17]	SARIMA model is used to forecast Ghana using average monthly load demand data. Ten-year data are considered for analysis, out of which for nine months are considered as training data, one year as validation, and the remaining as the test data to test the model. The order of the model is SARIMA (1, 1, 1) (0, 1, 2).	
Non-linear models	SVR [18]	SVR model is developed with immune algorithm (IA) to forecast the annual power load in Taiwan, China. Data from 1981 to 2000 are used to train and test the model. The optimal parameters required to apply the SVR model are estimated using simulated annealing approach.	1) Advantages: ① a non-linear function is used to predict the future values ② Iit is more generalized, adaptable, and responsive ③ the input and output patterns can be mapped accurately 2) Disadvantages: ① it is easy to get stuck into local minima ② it requires constraints, such as parameter tunning ③ Kernel selection is a complex process
	Least square SVR (LS-SVR) [19]	The LS-SVR model for load forecasting of a commercial space in Guangzhou, China. The hourly climate data and building cooling load for five months are considered to train the model. Compared with ANN, the results show that LS-SVR performs superior using the mean absolute relative error (MARE).	
	ANN [20]	A feedforward ANN is developed to forecast cooling loads for three educational buildings in Singapore. The daily energy data of the last two years are used for the analysis. The data are divided into several classes to reduce the variation in the data. The energy data for the previous five days are taken as the output to forecast the next day.	
	Multi-layer perceptron (MLP) and radial basis function network (RBFN) [21]	The combination of the MLP and RBFN is used for hourly air temperature forecasting. Four models are developed with the 24-hour time series of air temperature in a day as the output, and the output layer consists of 24 neurons.	
	Fuzzy neural networks [22]	A linguistic out-sample approach for fuzzy time series is used to forecast the daily power load in Malaysia. The weights of the fuzzy logical relationship (FLR) and the index number of close relationships in the fuzzy logical group are used. Daily electric load data for eight months are considered for the analysis.	
Ensemble models	Differential-EMD (DEMD)-SVR-SR [23]	An integrated DEMD approach is proposed to disintegrate the original load series into different modes. The residual is then forecasted using the auto-regressive (AR) model, while IMFs with non-linear SVR for better forecasting accuracy.	1) Advantages: ① the combination of linear and non-linear models is used to predict the future values ② these models are more robust and efficient ③ the forecasting accuracy is high 2) Disadvantages: ① a proper network composition is required ② the complexity is high ③ it is easy to get stuck into local minima
	EMD-DBN [11]	An EMD-DBN-based approach is proposed for STLF, where two restricted RBMs and DBN are used to accurately predict the individual IMFs obtained from EMD.	
	Ensemble DBN (EDBN)-SVR [10]	An EDBN is implemented for time-series forecasting. The outputs from various DBNs are aggregated using the SVR model.	
	SSA-LSTM [24]	SSA is used to eliminate the noisy components of a skewed load series. LSTM model uses the outcome of SSA to forecast the final load.	
	SSA-SVM-ARIMA-cuckoo search (CS) [25]	SSA is used to decompose the original series to identify and extract interpretable components from the original series. The linear part of the data is modeled using ARIMA and nonlinear part using the SVR optimized using the CS algorithm.	

Furthermore, deep learning models can capture the nonlinear characteristics of time-series data. These models have a unique ability of capturing the hidden features in the time series. Recurrent neural networks (RNNs) are considered to be very powerful in managing the sequence dependence of time-series data. The LSTM network [26], [27] is a particular type of modified RNN widely used in deep learning to successfully model a large amount of data. LSTMs are formulated explicitly to solve the problem of long-term dependency encountered by RNNs. By default, LSTM networks can remember data for a long time. In this paper, we present a novel attention mechanism using a sequence-to-sequence (Seq2Seq) model combined with EMD as a data pre-processing technique [28]. The Seq2Seq model is a deep neural network model based on LSTM units. EMD is an empirical method used in time-series analysis to characterize the instantaneous frequency data from a nonlinear time series; thus, it improves the forecasting accuracy. In addition, the EMD decomposes a varying time signal into several IMFs and a residual component, which corresponds to the trend and seasonality.

The remainder of this paper is organized as follows. Section II contains the theoretical background of the methodologies, data description, and a summary of methods. Section III elaborates on the comparative studies on the experimental results. Section IV draws the conclusions and presents the outlook of future work.

II. Preliminaries, Data Analysis, and Methodology

A. LSTM Network

LSTM is a particular class of RNN models with a particular capability to remember long-term temporal dependencies. The default property of these networks is to remember information for long periods. In addition, these models have a chain-like structure called cell state with repeating loops. The repeating loops help the network retain relevant information for short periods.

For an input sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$ ($\boldsymbol{x}_t \in \mathbf{R}^n$), the LSTM network calculates $\boldsymbol{h}_t \in \mathbf{R}^m$ for each time step t. The recurrent function of the LSTM cell can be defined as:

$$(\boldsymbol{h}_t, \boldsymbol{c}_t) = F(\boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{x}_t) \tag{1}$$

It can be expressed as:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{h}_{t-1} \quad \boldsymbol{x}_t] + \boldsymbol{b}_f) \tag{2}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{h}_{t-1} \quad \boldsymbol{x}_t] + \boldsymbol{b}_i) \tag{3}$$

$$\tilde{\boldsymbol{c}} = \tanh(\boldsymbol{W}_c[\boldsymbol{h}_{t-1} \quad \boldsymbol{x}_t] + \boldsymbol{b}_c) \tag{4}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}} \tag{5}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1} \quad \boldsymbol{x}_t] + \boldsymbol{b}_0) \tag{6}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{7}$$

where $\boldsymbol{f}_t$, $\boldsymbol{i}_t$, $\tilde{\boldsymbol{c}}$, $\boldsymbol{c}_t$, $\boldsymbol{o}_t \in \mathbf{R}^m$; $\boldsymbol{W}_f$, $\boldsymbol{W}_i$, $\boldsymbol{W}_c$, $\boldsymbol{W}_o \in \mathbf{R}^{m \times n}$; $\boldsymbol{c}_t$ is the context vector; $\boldsymbol{b}_f$, $\boldsymbol{b}_i$, $\boldsymbol{b}_c$, and $\boldsymbol{b}_0$ are the bias vectors; and $\odot$ represents the Hadamard product.

B. Basic Attention Mechanism

Reference [29] proposes a basic attention mechanism, which calculates the weighted sum of the encoder RNN output and uses it to generate a context vector. Provided an input $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$, it stores all the encoded data $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T]$, where the dimension of $\boldsymbol{H}$ is $n \times m$, $m = T$; and n is the size of the RNN unit. The attention mechanism, which is a feedforward neural network, accepts the previous decoder hidden state $\boldsymbol{h}_t$ and one of the cell state vectors $\boldsymbol{d}_{t-1}$ as input and then outputs a relevant score $\boldsymbol{e}_t$. The mechanism begins with computing $\boldsymbol{e}_t$ ($t = 1, 2, \ldots, T$) using the score function $f_{att}(\cdot)$ as:

$$\boldsymbol{e}_t = f_{att}(\boldsymbol{h}_t, \boldsymbol{d}_{t-1}) \tag{8}$$

The attention score is α_t ($t = 1, 2, \ldots, T$), which is calculated using the softmax function as:

$$softmax(\alpha_t) = \frac{\exp(\boldsymbol{e}_t)}{\sum_t \exp(\boldsymbol{e}_t)} \tag{9}$$

The context vector $\boldsymbol{C}_t$ ($t = 1, 2, \ldots, T$) is the weighted sum of all encoded data $\boldsymbol{h}_t$, which can be expressed as:

$$\boldsymbol{C}_t = \sum_{t=1}^{T} \alpha_t \boldsymbol{h}_t \tag{10}$$

The computed value of $\boldsymbol{C}_t$ is used to predict the output. In the training process, $\boldsymbol{C}_t$ is one of the decoder inputs along with $\boldsymbol{d}_{t-1}$ and $\hat{\boldsymbol{y}}_{t-1}$, and the output is $\hat{\boldsymbol{y}}_t$. In the testing process, the output from previous step $\hat{\boldsymbol{y}}_{t-1}$, along with $\boldsymbol{d}_{t-1}$ and $\boldsymbol{C}_t$, are used as the input.

C. EMD

It is crucial to explore the inherent properties of time-series data to acquire the trend and seasonality components. In 1998, the EMD [28] was first introduced to decompose a signal into several IMFs and a residue using the Hilbert transform without taking any base function or filter function. It was observed that EMD works very well with nonlinear and nonstationary time-series data. Each IMF corresponds to a particular frequency band obtained from the original data. The more complex the data composition is, the more IMFs are obtained.

The original series $S(t)$ can be reconstructed from the decomposition by having a linear addition of all the IMFs IMF_i and the residual Res as:

$$S(t) = \sum_{i=1}^{n} IMF_i + Res \tag{11}$$

For the EMD, each IMF has to meet the following two criteria:

1) In the entire information set, the numbers of extrema and zero crossings should either be equal or different at most by one.

2) The average of the envelope outlined by the native maxima and minima should be zero.

The calculation of IMFs is an iterative process and continues until the number of extreme points is less than two, which results in a residual series of decomposition.

The procedure of EMD is shown in Algorithm 1.

Algorithm 1: the procedure of EMD

Step 1: consider the original series as $S(t)$ and take $x(t)=S(t)$ as initialization that should either be equal or different at most by one.
Step 2: find the extreme points of the series, which are termed as local maxima e_{max} and local minima e_{min}.
Step 3: use the extreme points and apply a cubic spline interpolation to find the surrounding envelope.
Step 4: calculate the average of points on the surrounding envelope as: $m_i(t)=(e_{max,i}+e_{min,i})/2$.
Step 5: compute the difference $h_i(t)=x(t)-m_i(t)$.
Step 6: if $h_i(t)$ satisfies the two criteria mentioned above for an IMF, then it is accepted as IMF_i, otherwise repeat *Steps* 2-5 until valid IMFs are found.
Step 7: update $r(t)=x(t)-IMF_i$. Repeat *Steps* 2-6 until the number of extreme points is less than two, which results in residual series and decomposition stops.

Finally, the original time-series signal is decomposed as:

$$x(t)=\sum_{i=1}^{n}c_i+r \tag{12}$$

where c_i is the i^{th} IMF extracted in the i^{th} decomposition process; and r is the final residue.

D. Data Description

For comparison, five publicly-available time-series datasets of load demand from the AEMO repository [30] are used. The datasets for 2013 are collected with a half-hour sampling frequency, which provides 17520 samples for each state (New South Wales (NSW), South Australia (SA), Tasmania (TAS), Queensland (QLD), and Victoria (VIC)).

Load demand depends on many external factors, such as the time scale. Figure 1 shows the monthly load demand patterns. Figure 2 shows the box-plot of the monthly load demand in 2013.

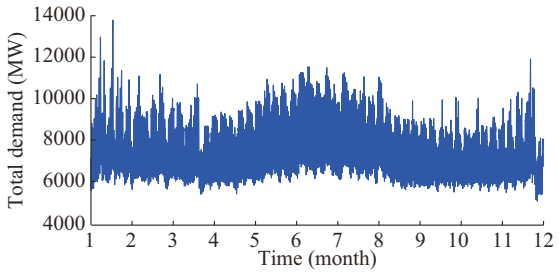

Fig. 1. Original series of monthly load demand.

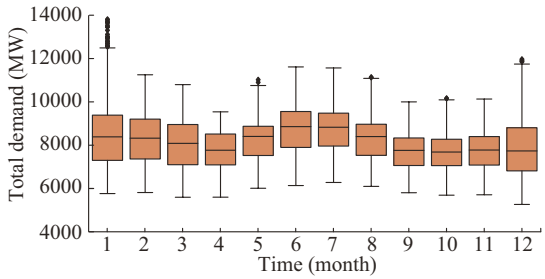

Fig. 2. Box-plot of monthly load demand.

Figure 2 shows that the median of load demand decreases in the first quarter, increases in the second, decreases in the third, and increases in the fourth. It suggests that the demand data depends on the time-of-year effect because of the different working conditions of air conditioning and other similar appliances in winter and summer. The datasets can be categorized into four different seasons based on the time-of-year effect: autumn (March, April, and May), winter (June, July, and August), spring (September, October, and November), and summer (December, January, and February). To examine the seasonal component of the data, one month from each season is considered, e. g., January, April, July, and October are used. In this study, 80% (1152 samples) of total samples are used as the training set, 10% (168 samples) as the validation set, and the rest (168 samples) as the test set.

As a result, a cyclic pattern is identified in the data. Figure 3 presents the cyclic pattern in half-hourly and hourly data readings, and indicates that a daily cycle is repeated at a regular time interval. Figure 4 shows the auto-correlation plot for January in 2013 in NSW and a zoomed graph showing auto-correlation among the first 96 lags. The correlation between the lags decreases as the sample size increases. In addition, a strong correlation among the first 96 lags is observed, and it further decreases in the subsequent lags. Consequently, we use grid search on various lag values to select the desired features (time steps) for training the model. Table II shows root mean square error (RMSE) values corresponding to different lag values (time steps) for the NSW data. The lag value of 96 provides the lowest RMSE value. The analysis confirms that the forecasting load depends on the load of the current and previous days. Therefore, the number of time steps in the model is $48\times2=96$.

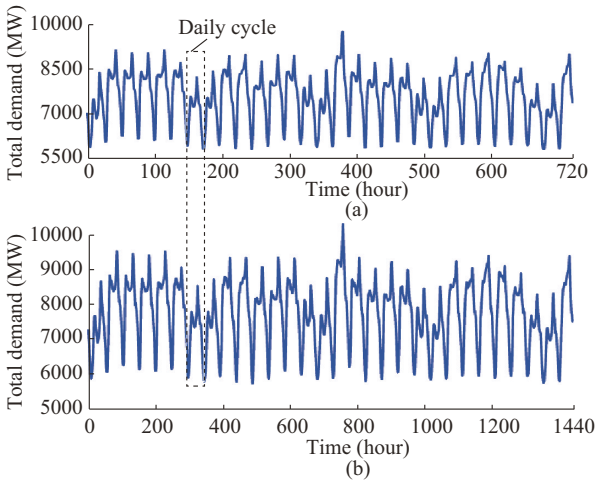

Fig. 3. Cyclic patterns of example time-series. (a) Hourly data. (b) Half-hourly data.

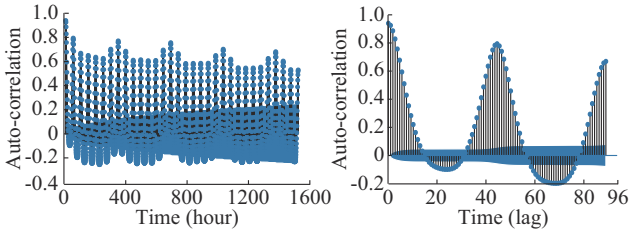

Fig. 4. Auto-correlation plot for January in 2013 and zoomed graph showing auto-correlation among the first 96 lags. (a) Auto-correlation plot. (b) Zoomed graph.

TABLE II
RMSE VALUES CORRESPONDING TO DIFFERENT TIME STEPS FOR NSW DATA

Time step	RMSE			
	January	April	July	October
96	556.56	287.24	263.15	158.97
148	567.11	291.67	271.07	159.09
240	567.45	288.45	271.86	163.22
288	581.24	298.90	288.58	172.01
336	564.22	285.32	266.44	159.78
384	603.98	312.45	291.81	188.29

The forecasting process is mainly divided into two phases. In the first phase, EMD is used to gain the underlying multi-scale dynamics of the load time series such as trend and seasonality. In the second phase, an attention-based deep learning model is used to predict the decomposed time series.

The flow chart of the procedure for the proposed EMD model is shown in Fig. 5. After grouping the data into stable and unstable IMFs and residual terms, the fine-grained attention-based model predicts each group. Figure 6 shows the training procedure of the proposed model, where *Result* is the corresponding forecasting result.

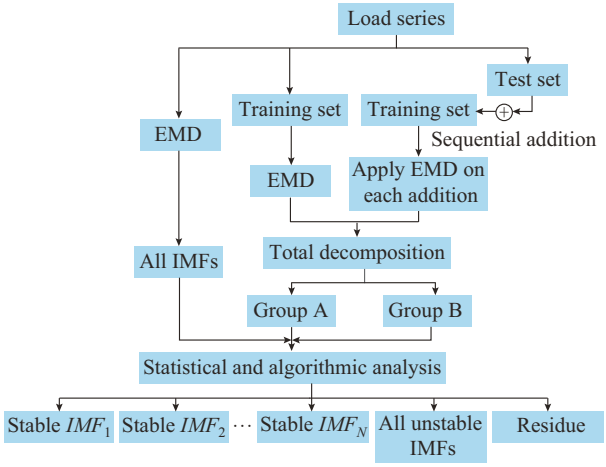

Fig. 5. Flow chart of procedure for EMD model.

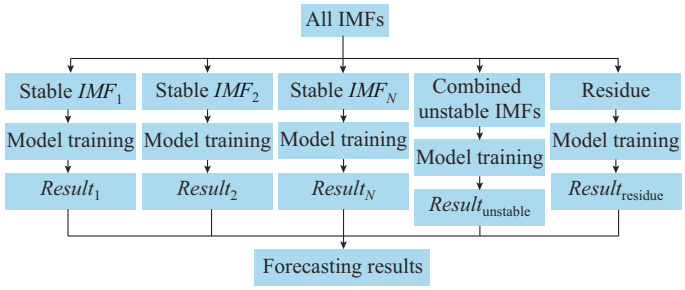

Fig. 6. Flow chart of training procedure for proposed model.

A divide-and-conquer algorithm recursively divides the problem into sub-problems for simplification and directly solve simple sub-problems. Subsequently, sub-problem solutions are combined to find the solution of the original problem.

In the proposed method, the load demand data are decomposed into several IMFs and one residue using the EMD method. Even though the EMD decomposes complex series into subsequent simpler series, it is necessary to thoroughly check the decomposition stability. This information helps the training process as stable information and leads to a more precise estimation. To perform our analysis, we consider the season-wise data. The main idea is to check whether the data remain stable after adding more samples. The insights into our approach can be explained as follows.

Step 1: season-wise data are divided into the training and testing sets.

Step 2: the EMD is performed on the training set to obtain the respective IMFs and residual series.

Step 3: data from the test set are added one by one, and decomposition by EMD is carried out every time.

Step 4: *Step 3* is repeated until the entire test set is added sequentially.

Step 5: after obtaining total decompositions, we divide them into two groups: Group A and Group B, representing decompositions with the most frequent count of IMFs (in our case it is 9) and the remaining decompositions, respectively.

Step 6: a statistical approach such as Pearson correlation, and a comparison algorithm such as the lower bound (LB) [31] are used to check the stability of each IMF in the groups.

Step 7: based on the results, the entire decomposition set is divided into three sets: stable and unstable IMFs, and residue.

Step 8: using the algorithm explained in Algorithms 2-4, the attention-based LSTM is trained to obtain the forecasting results for each set.

Step 9: all the forecasting results are combined by summation to formulate an ensemble output for time series.

The existing attention models use the single scalar score for a context vector $\boldsymbol{C}_t$ at time t. It is observed that instead of using a single scalar of context vector $\boldsymbol{C}_t$, it might be beneficial to calculate and use the scalar score for each dimension of the hidden state $\boldsymbol{h}_t$ at time t, as each dimension represents a different perspective of the captured internal structure. In the encoder-decoder model computation, $\boldsymbol{C}_t$ shares the same attention score, resulting in an equal contribution of all dimensions of $\boldsymbol{h}_t$.

Reference [32] shows that when different dimensions of encoded information are considered differently, and the attention is applied to each dimension, it results in a better-performing model. Inspired by [32], we propose a fine-grained attention model. In the proposed model, scalars are maintained for each dimension in $\boldsymbol{H}$, which results in an increase in the number of attention scalars from T to nT. Equations (13) and (14) illustrate the comparison between the basic attention model and the fine-grained attention model.

$$\alpha_1 \boldsymbol{h}_1 + \alpha_2 \boldsymbol{h}_2 + \ldots + \alpha_T \boldsymbol{h}_T = \boldsymbol{c}_t \tag{13}$$

$$\begin{cases} \alpha_1^1 \boldsymbol{h}_1^1 + \alpha_t^1 \boldsymbol{h}_2^1 + \ldots + \alpha_T^1 \boldsymbol{h}_T^1 = \boldsymbol{c}_t^1 \\ \alpha_1^2 \boldsymbol{h}_1^2 + \alpha_t^2 \boldsymbol{h}_2^2 + \ldots + \alpha_T^2 \boldsymbol{h}_T^2 = \boldsymbol{c}_t^2 \\ \quad \vdots \qquad\quad \vdots \qquad\quad \vdots \quad \vdots \\ \alpha_1^n \boldsymbol{h}_1^n + \alpha_2^n \boldsymbol{h}_2^n + \ldots + \alpha_T^n \boldsymbol{h}_T^n = \boldsymbol{c}_t^n \end{cases} \tag{14}$$

In the proposed model, we extend the score function $f_{att}(\cdot)$ to return a set of scores corresponding to the dimensions of the hidden state vector $\boldsymbol{h}_t$, which is expressed as:

$$\boldsymbol{e}_t^n = f_{att}^n(\boldsymbol{h}_t, \boldsymbol{d}_{t-1}) \tag{15}$$

where $\boldsymbol{e}_t^n$ is the score assigned to the n^{th} dimension of context vector; and f_{att}^n is the fully-connected neural network. These dimension-specific scores are further normalized dimension-wise, which is expressed as:

$$\boldsymbol{\alpha}_t^n = \frac{\exp(\boldsymbol{e}_t^n)}{\sum_{t=1}^{T}\exp(\boldsymbol{e}_t^n)} \tag{16}$$

The context vectors are then computed as:

$$\boldsymbol{C}_t = \begin{bmatrix} \sum_{t=1}^{T}\alpha_t^1 \boldsymbol{h}_t^1 \\ \sum_{t=1}^{T}\alpha_t^2 \boldsymbol{h}_t^2 \\ \vdots \\ \sum_{t=1}^{T}\alpha_t^n \boldsymbol{h}_t^n \end{bmatrix} \tag{17}$$

Algorithms 2-4 explain the pseudocode for the steps used in data preparation, LSTM encoder, and attention and decoder LSTM, respectively.

Algorithm 2: Pseudo code for data preparation

Step 1: scale the stable IMFs (X_s), unstable IMFs (X_{uns}), and residual IMFs (X_r) using MinMaxScalar.
Step 2: define and initialize the different values to be used such as batch size, train size, time steps, and forecasting steps.
Step 3: divide the three series into train, test, and validation sets and create 9 series, i. e., $X_{s,train}$, $X_{s,test}$, $X_{s,validation}$, $X_{uns,test}$, $X_{uns,validation}$, $X_{r,train}$, $X_{r,test}$, and $X_{r,validation}$, respectively.
Step 4: create three models for each training series, i.e., $model_{st}$ for $X_{s,train}$, $model_{uns}$ for $X_{un,strain}$, and $model_{res}$ for $X_{r,train}$ using Algorithm 3.

Algorithm 3: Pseudo code for LSTM encoder

Step 1: encoder takes N (the length of X_{train}), M (the number of LSTM units), and T (the number of time steps initialized in Algorithm 2) along with X_{train} as the inputs.
Step 2: encoder using the LSTM cell as an encoded model ($model_{en}$) is created using N and M.
Step 3: define the initial hidden states and cell state at time t as $\boldsymbol{h}_t$ and $\boldsymbol{d}_t$ with zeros using the size of the X_{train} and M, respectively.
Step 4: iterate over the input through time T and take X_{train} at time t as $X_{train,t}$.
Step 5: take $X_{train,t}$ as the input to the $model_{en}$ along with $\boldsymbol{h}_t$ and $\boldsymbol{d}_t$.
Step 6: take the output hidden states in $\boldsymbol{h}_t$ and $\boldsymbol{d}_t$ and use them as the input to the model in the next iteration.
Step 7: output $\boldsymbol{h}_t$ to the encoded input.
Step 8: at the end of this iteration, the encoded input is completely populated for the complete size of the input.

Algorithm 4: Pseudo code for attention and decoder

Step 1: encoder takes N, M, and T along with X_{train} as the input.
Step 2: encoder uses LSTM cell as an encoded model $model_{en}$ using N and M.
Step 3: define the initial hidden states and cell state at time t as $\boldsymbol{h}_t$ and $\boldsymbol{d}_t$ with zeros using the size of the X_{train} and M, respectively.
Step 4: define the initial context vector $\boldsymbol{c}_t^n$ at time t and hidden state $\boldsymbol{h}_t^n$ using encoded input size.
Step 5: iterate over the input through time T and define $encoded_{input}$ at time t as $encoded_{input,t}$.
Step 6: concatenate the hidden states $\boldsymbol{h}_t$ and $\boldsymbol{d}_t$ and store it in $\boldsymbol{h}_{dt}$.
Step 7: apply the linear weight to $\boldsymbol{h}_{dt}$ to use it as the weight for attention mechanism and store it in w_1.
Step 8: iterate over the different dimensions n of hidden state $\boldsymbol{h}_t$.
Step 9: apply the linear weight to $encoded_{input}(n)$, which is the n^{th} dimension of the hidden state $\boldsymbol{h}_t$ and store it in w_2.
Step 10: add w_1 and w_2, and apply tanh(·) to it, then store the output into w.
Step 11: apply the linear weight to w and store it in w.
Step 12: apply softmax(·) to w and store it into n.
Step 13: multiply n_t to $encoded_{input}$ using the 1st dimension, as the 1st dimension stores the dimension of hidden state.
Step 14: store the output as context vector at time t for the n^{th} hidden state, i.e., $\boldsymbol{c}_t^n$.
Step 15: store the sum of all the $\boldsymbol{c}_t^n$ obtained from the loop into $\boldsymbol{c}_t$ as context vector at time t.
Step 16: concatenate y_{train} at time t to $\boldsymbol{c}_t$ and apply linear weight to it.
Step 17: store the result in $\tilde{\boldsymbol{y}}_t$.
Step 18: apply $model_{dec}$ to $\tilde{\boldsymbol{y}}_t$ along with the inputs $\boldsymbol{h}_t$ and $\boldsymbol{d}_t$ hidden states.
Step 19: store the output hidden states in $\boldsymbol{h}_t$ and $\boldsymbol{d}_t$.
Step 20: use the updated hidden states in the next iteration of t.
Step 21: concatenate context vector and hidden state at step T and store it into $\boldsymbol{d}_{cT}$.
Step 22: apply the linear weight to $\boldsymbol{d}_{cT}$ to calculate the output and store the output in decode output.

III. Experimental Work

A. Variable Decomposition Results of EMD

The number of decompositions depends on the complexity and length of the series, that is, the more complex the formation of data, the more IMFs obtained. In particular, the decomposition may be affected by various exogenous factors in addition to new data, as shown in Fig. 8. Similarly, it can be observed that there is a variable count of IMFs, i.e., 7, 8, and 9 when new data are added.

Although we are unaware of hidden factors affecting the number of IMFs, it is necessary to train the model using the most stable IMFs to obtain the best estimation.

We choose legitimate IMFs that will not significantly change when the quantity of deterioration results shifts with the addition of new data. Therefore, we add the state of new information in each step and subsequently evaluate the IMFs using EMD. Subsequently, each recording is taken at a sampling frequency of 30 min from the test set, and the EMD is re-executed to decompose the results every time. We record each decomposition and achieve the statistics discussed earlier. From the 480 decomposition results, there are 48 instances of 7 IMFs, 417 instances of 8 IMFs, and 15 instances of 9 IMFs. Based on this analysis, it is concluded that the final decomposition consisting of the entire data result is 8 IMFs.

It can be observed that most data have a similar decomposition outcome as 8 IMFs, whereas some data have variable results depending on hidden and exogenous factors. When the final result of decomposition has a constant number of IMFs, the decomposition results are categorized into groups A and B. Group A consists of most cases, i.e., 8 IMFs, and group B contains the other instances. Subsequently, all decomposition results are compared with the final result using the Pearson correlation [33] and LB [31] between the corresponding IMFs, which can ignore the order of magnitude.

Table III shows that only the IMFs 1-4 are stable, while the rest of them exhibit a larger variance. Therefore, we combine the unstable components as one frequency band and train the stable IMFs separately.

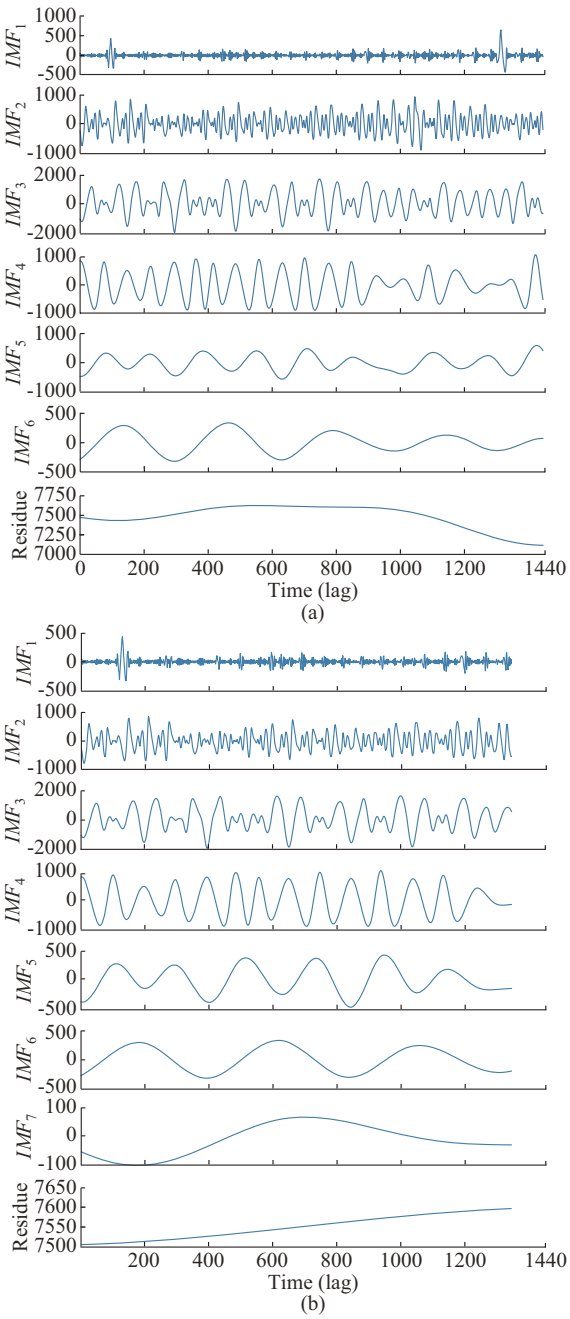

Fig. 8. Variable decomposition results of EMD after adding new information in existing data. (a) Decomposition results on training data of NSW dataset. (b) Decomposition results after adding the 17th sample from test data in training samples of NSW dataset.

The frequency combination is performed to reduce the individual errors that could be induced on separate training. Thus, the 8 IMFs are formed in groups of three for training. IMFs 1-4 form the most stable group, whereas the frequencies of IMFs 5-7 from unstable and residual parts are trained separately to obtain the trend of the series. The forecasting result of each group is linearly added to obtain the final forecasting. Moreover, no IMF should be missed because it may mismatch with the scale of the end result. We divide each group into training, validation, and testing samples. The training samples are pre-processed, and the min-max scaler performs the scaling. This increases the convergence rate of the training mechanism. To avoid large weights in the high-extent training values, data are scaled between $[-1, 1]$. In this study, we use an LSTM-based fine-grained attention mechanism (EMD-Att-LSTM) because it successfully performs with temporal data. This model is selected owing to its ability to handle long-term dependencies and its fast convergence rate. The model training is performed using the lookback (sliding window) mechanism on the training dataset. The sliding window is the history data or preceding time steps used as the input to forecast the next time step. The forecasting is performed on a daily manner. An exhaustive search technique is used for hyper-parameter tuning. The input to the model has a shape with 96 time steps. Two LSTM layers are used in the model. The output of the first layer is used as the input to the second layer. The first LSTM layer uses 96 neurons, whereas the second layer contains 48 neurons. The rectified linear unit (ReLU) is used as the activation function in both layers. The layers also follow batch normalization and dropout with a value of 0.20 to avoid the model over-fitting. The output layer does not contain any activation functions. Only a dense layer is used as the output layer. The Adam optimizer is used in the model training mechanism. The learning rate and momentum values are 0.001 and 0.90, respectively. The model is trained for 150 epochs using a batch size of 20. We use the Pytorch machine-learning library [34] to implement our model. In addition, we use mean absolute percentage error (MAPE) and RMSE indices to assess the performance of the proposed model.

TABLE III
ANALYSIS OF EACH DECOMPOSITION ON NSW DATASET

Method	IMF	Mean value		
		All data	Group A	Group B
Pearson index	IMF_1	0.4661	0.4753	0.4376
	IMF_2	0.4859	0.4862	0.4850
	IMF_3	0.4426	0.4419	0.4447
	IMF_4	0.4599	0.4596	0.4608
	IMF_5	0.3805	0.3808	0.3798
	IMF_6	0.4202	0.4311	0.3869
	IMF_7	0.3261	0.3548	0.2374
	IMF_8	0.1507	0.1551	0.1371
Keogh index	IMF_1	693.1300	551.2600	592.5400
	IMF_2	2567.6800	2413.4200	3043.0600
	IMF_3	13878.3300	13942.9600	13678.9400
	IMF_4	14287.7700	14107.7600	14943.4500
	IMF_5	10216.4700	9925.5100	11114.1000
	IMF_6	11163.9300	10164.8900	14246.0800
	IMF_7	10258.2600	9148.0400	13683.3100
	IMF_8	29094.0100	1387.5200	76038.5400

IV. RESULTS

In this section, we compare the output of the proposed method to that of some existing advanced methods, including persistence models, SVR [35], ANN [36], DBN [37], RF [38], EMD-SVR [39], EMD-ANN [40], ensemble DBN [10], EMD-DBN [11], DMD [41], and VMD-Att-LSTM. We consider half-an-hour- and one-day-ahead as the two forecasting

horizons for the comparative study. To reflect different seasons, one month is selected from each season, that is, January, April, July, and October.

A. Comparison Results for Half-an-hour-ahead Forecasting

The persistence model is the most basic one of the existing models used for STLF. The series must be stationary to apply the persistence model; thus, these models consider limited skewness during STLF. Therefore, these models can be used as a basis for evaluating the efficiency of various deep learning, data-driven, and ensemble structures. Table IV presents half-an-hour-ahead forecasting of various advanced models. The bold font indicates the best performing model for certain time series. The basic machine-learning algorithms, such as SVR, ANN, DBN, and RF are used to model the electric load time series, which are subsequently coupled with EMD to form hybrid models and improve the forecasting performance.

TABLE IV
HALF-AN-HOUR-AHEAD LOAD FORECASTING RESULTS

Dataset	Time	Index	Prediction models													
			Persistence	SVR [35]	ANN [36]	DBN [37]	RF [38]	LSTM	EDBN [10]	EMD-SVR [39]	EMD-ANN [40]	EMD-RF	EMD-LSTM	EMD-DBN [11]	VMD-Att-LSTM	Proposed
NSW	January	RMSE	164.02	64.24	96.66	79.16	93.36	78.01	75.42	78.56	82.11	76.10	57.29	49.86	**46.61**	47.94
		MAPE (%)	1.64	0.93	0.98	0.78	0.88	0.76	0.70	0.78	0.88	0.76	0.64	0.53	**0.41**	0.45
	April	RMSE	248.14	162.57	140.74	70.36	142.85	132.78	134.47	114.09	87.76	120.16	71.91	69.55	57.93	**44.58**
		MAPE (%)	2.43	1.88	1.26	0.64	1.18	1.13	1.14	1.07	0.82	1.10	0.64	0.65	0.61	**0.46**
	July	RMSE	235.66	117.87	165.42	105.63	114.83	72.26	78.09	74.29	81.22	120.77	63.67	75.09	68.65	**52.81**
		MAPE (%)	2.31	1.20	1.68	1.09	1.20	0.71	0.67	0.67	0.81	1.22	0.51	0.70	0.54	**0.44**
	October	RMSE	159.98	58.26	76.64	62.58	69.06	57.87	64.36	54.58	66.00	76.58	55.76	51.68	49.04	**37.54**
		MAPE (%)	1.65	0.64	0.82	0.70	0.75	0.62	0.66	0.60	0.74	0.82	0.62	0.55	0.63	**0.39**
TAS	January	RMSE	17.80	13.87	13.84	12.90	11.80	12.03	13.24	12.54	11.39	13.52	9.03	11.59	9.43	**6.43**
		MAPE (%)	1.27	1.09	1.09	1.01	1.03	1.08	1.06	0.97	0.78	1.09	0.57	0.74	0.59	**0.46**
	April	RMSE	37.42	25.26	27.03	19.53	24.82	21.99	21.35	21.76	18.03	25.19	16.81	16.23	13.78	**8.80**
		MAPE (%)	2.32	1.49	1.63	1.25	1.26	1.24	1.21	1.36	1.16	1.45	1.11	1.07	0.89	**0.57**
	July	RMSE	43.84	34.03	33.97	30.43	33.59	28.71	22.62	30.90	29.14	32.34	18.45	24.44	19.22	**11.35**
		MAPE (%)	2.73	2.10	2.03	1.76	2.07	1.79	1.36	1.91	1.98	2.17	1.15	1.54	1.15	**0.72**
	October	RMSE	22.80	15.89	18.49	16.80	16.94	14.37	20.41	14.94	9.37	13.69	9.49	8.81	**7.96**	8.03
		MAPE (%)	1.63	1.09	1.36	1.19	1.19	1.02	1.34	1.06	0.70	0.97	0.72	0.66	**0.55**	0.55
QLD	January	RMSE	109.39	51.31	62.97	44.95	40.30	43.51	51.03	42.12	33.88	33.34	33.72	**25.39**	34.25	31.20
		MAPE (%)	1.50	0.70	0.85	0.62	0.59	0.61	0.63	0.57	0.48	0.44	0.46	**0.34**	0.47	0.42
	April	RMSE	137.11	71.30	65.84	48.48	57.20	57.98	60.27	51.30	56.02	54.78	43.65	48.34	34.98	**24.55**
		MAPE (%)	1.91	0.94	0.93	0.66	0.86	0.82	0.75	0.61	0.73	0.81	0.59	0.67	0.51	**0.34**
	July	RMSE	127.23	46.07	51.79	38.45	45.52	40.19	42.00	35.53	41.48	45.39	29.31	30.61	29.03	**21.08**
		MAPE (%)	1.92	0.72	0.82	0.54	0.85	0.57	0.60	0.53	0.62	0.69	0.42	0.44	0.42	**0.31**
	October	RMSE	110.19	57.90	63.17	61.03	61.37	48.93	55.49	54.89	46.78	48.41	41.23	40.46	39.98	**34.63**
		MAPE (%)	1.61	0.80	0.92	0.86	0.77	0.68	0.71	0.75	0.69	0.67	0.59	0.56	0.50	**0.47**
VIC	January	RMSE	174.95	117.79	114.73	106.25	120.34	87.60	78.58	82.96	117.45	115.66	58.75	98.75	51.71	**38.85**
		MAPE (%)	2.52	1.54	1.55	1.32	1.58	1.20	1.05	1.09	1.56	1.59	0.91	1.35	0.87	**0.58**
	April	RMSE	162.18	148.89	149.20	104.48	102.34	78.11	75.59	96.24	77.02	68.30	53.72	64.11	51.40	**38.24**
		MAPE (%)	2.15	1.97	1.99	1.49	1.53	1.02	0.98	1.33	0.99	0.92	0.67	0.87	0.62	**0.58**
	July	RMSE	171.99	69.41	119.43	86.63	62.57	69.47	66.36	119.27	114.34	61.84	56.34	58.70	**45.07**	45.12
		MAPE (%)	2.44	1.01	1.74	1.26	1.00	1.06	0.91	1.73	1.67	0.88	0.85	0.88	**0.60**	**0.60**
	October	RMSE	139.47	62.88	96.63	91.20	87.11	67.74	68.50	90.49	91.38	55.19	46.19	57.95	43.16	**33.00**
		MAPE (%)	1.97	0.92	1.42	1.35	1.20	0.93	0.89	1.23	1.23	0.85	0.71	0.84	0.66	**0.48**
SA	January	RMSE	72.11	50.37	55.65	59.17	53.92	47.11	39.87	58.29	46.34	42.52	29.37	51.91	27.23	**18.97**
		MAPE (%)	3.04	1.92	2.33	2.54	2.23	1.81	1.69	2.29	1.73	1.55	1.32	1.69	1.19	**1.19**
	April	RMSE	58.53	45.40	39.55	44.85	46.42	42.63	35.65	26.14	27.98	31.27	26.01	33.44	21.40	**19.80**
		MAPE (%)	3.03	1.93	2.33	2.54	2.50	1.81	1.75	1.37	1.54	1.83	1.35	1.89	**1.24**	1.25
	July	RMSE	75.05	47.68	56.12	48.55	59.99	38.54	38.03	37.38	39.16	31.93	29.75	29.18	**20.27**	20.51
		MAPE (%)	4.25	2.34	3.53	3.07	3.49	1.97	1.98	2.17	2.33	1.86	1.73	1.70	**1.12**	1.15
	October	RMSE	48.49	42.74	37.94	40.56	42.48	36.44	43.59	30.16	25.14	26.59	22.53	34.17	21.73	**15.85**
		MAPE (%)	2.62	1.77	2.23	2.17	2.30	2.19	1.92	1.67	1.69	1.71	1.51	1.62	1.48	**0.95**

B. Comparison Results for One-day-ahead Forecasting

Table V shows one-day-ahead forecasting results. The persistence model and the same set of machine-learning and hybrid models are used for half-an-hour-ahead forecasting. The persistence model is used as a reference model for other models as the dataset includes daily seasonality.

TABLE V
ONE-DAY-AHEAD FORECASTING RESULTS

Dataset	Time	Index	Prediction model														
			Persistence	SVR [35]	ANN [36]	DBN [37]	RF [38]	LSTM	EDBN [10]	EMD-SVR [39]	EMD-ANN [40]	EMD-RF	EMD-LSTM	EMD-DBN [11]	DMD [41]	VMD-Att-LSTM	Proposed
NSW	January	RMSE	978.24	703.43	750.53	639.75	521.14	603.27	636.03	611.20	748.30	544.17	579.87	541.53	**445.79**	501.44	556.56
		MAPE (%)	8.55	6.23	7.20	5.95	4.26	5.09	5.70	5.19	6.66	4.54	4.98	4.62	**4.20**	4.37	4.55
	April	RMSE	729.54	474.38	578.05	361.63	500.70	377.10	551.74	569.28	512.59	495.28	349.08	377.63	504.39	404.24	**287.24**
		MAPE (%)	6.71	4.27	5.41	3.36	4.25	3.87	4.78	5.27	4.57	4.21	3.31	3.22	5.04	3.89	**3.17**
	July	RMSE	609.82	574.30	534.75	415.81	387.15	401.35	414.90	402.69	345.90	353.90	376.21	322.04	329.12	288.07	**263.15**
		MAPE (%)	6.22	5.86	5.38	4.11	4.01	3.91	4.07	3.95	3.09	3.67	3.80	3.08	3.13	2.71	**2.49**
	October	RMSE	587.14	393.32	345.07	350.82	296.53	307.22	334.12	272.01	299.34	333.82	247.91	282.34	333.23	211.66	**158.97**
		MAPE (%)	5.36	3.74	3.48	3.41	2.78	2.99	3.14	2.76	2.90	3.17	2.36	2.71	3.15	1.98	**1.69**
TAS	January	RMSE	89.82	60.97	69.92	63.96	65.90	58.11	60.68	61.73	63.38	58.51	53.56	56.10	55.38	**45.91**	46.64
		MAPE (%)	7.24	4.81	5.42	4.98	4.77	4.67	4.82	4.49	4.87	4.67	4.07	4.05	4.30	**3.26**	3.45
	April	RMSE	157.73	111.89	94.40	93.18	92.64	84.37	109.78	104.59	87.41	86.61	72.04	85.13	93.30	67.45	**49.63**
		MAPE (%)	10.22	7.48	6.30	6.12	6.10	5.75	7.28	6.87	5.92	5.80	4.80	5.80	7.21	4.98	**3.09**
	July	RMSE	120.47	90.99	89.17	87.30	90.48	94.67	85.19	92.54	82.92	81.34	91.33	**73.91**	77.77	80.77	90.50
		MAPE (%)	8.11	5.89	6.28	6.04	6.17	5.93	6.04	6.09	5.50	5.54	5.06	**4.93**	5.16	5.87	5.08
	October	RMSE	109.46	79.45	72.86	75.73	69.80	69.13	80.81	82.85	80.85	73.86	64.76	68.26	69.06	68.11	**64.73**
		MAPE (%)	7.48	5.55	5.24	5.15	4.63	**4.51**	5.05	5.60	5.63	4.88	4.70	4.75	5.29	4.50	4.61
QLD	January	RMSE	461.09	282.07	299.32	228.86	195.85	324.56	218.55	196.20	273.70	**178.63**	311.16	191.22	513.10	371.80	354.78
		MAPE (%)	5.25	3.65	3.61	2.78	2.41	5.03	2.69	2.56	3.28	**2.21**	4.91	2.56	7.36	5.68	5.14
	April	RMSE	489.63	266.39	339.93	247.56	231.01	201.30	259.34	264.00	237.58	201.74	190.87	243.68	326.72	175.01	**152.85**
		MAPE (%)	6.25	3.53	3.77	2.99	2.78	2.37	3.33	3.47	3.11	2.44	2.26	2.93	3.86	**2.13**	2.46
	July	RMSE	430.46	223.17	203.00	213.20	156.08	188.01	159.45	164.68	174.64	150.01	186.97	**142.84**	176.77	152.88	148.36
		MAPE (%)	5.90	3.10	3.03	2.95	2.32	2.62	2.32	2.46	2.45	2.29	2.71	2.08	2.42	2.18	**1.88**
	October	RMSE	417.33	298.76	263.12	251.34	236.50	263.65	292.93	218.71	248.55	260.94	215.44	219.19	247.37	193.07	**181.55**
		MAPE (%)	5.54	3.93	3.46	3.40	2.88	3.70	3.53	2.82	3.27	3.15	2.79	2.88	3.13	2.89	**2.59**
VIC	January	RMSE	990.74	587.98	811.43	915.21	739.65	603.17	762.16	806.29	781.17	783.58	567.50	762.57	**469.67**	532.96	514.93
		MAPE (%)	9.48	7.16	9.32	8.79	8.77	6.89	9.14	9.48	9.07	9.32	**6.06**	8.86	6.65	8.28	7.77
	April	RMSE	669.87	330.93	359.03	353.02	366.16	411.89	343.18	363.50	376.12	393.63	403.34	**321.59**	479.99	387.00	390.80
		MAPE (%)	8.40	4.43	4.95	4.55	4.65	5.23	4.49	4.67	4.79	5.04	5.38	**4.35**	5.08	4.78	5.01
	July	RMSE	721.85	297.07	305.88	276.25	302.15	283.45	285.14	298.12	386.64	300.65	269.50	285.45	401.92	277.35	**259.85**
		MAPE (%)	9.76	4.38	4.29	3.72	4.29	3.79	3.65	4.27	5.29	4.26	3.59	3.83	5.38	3.81	**3.43**
	October	RMSE	577.70	391.11	347.91	389.06	364.32	352.81	401.02	309.63	332.44	344.06	329.10	322.91	295.90	**260.46**	262.22
		MAPE (%)	8.30	4.50	4.79	4.85	4.16	4.66	4.72	3.78	4.15	3.92	**3.67**	3.73	4.09	3.70	3.79
SA	January	RMSE	433.57	337.10	411.66	401.25	349.87	307.12	363.49	280.70	397.66	288.85	244.31	238.09	240.72	195.94	**118.59**
		MAPE (%)	14.32	13.34	13.72	13.62	13.41	12.89	14.43	11.13	13.80	13.03	12.77	10.46	12.67	9.70	**7.05**
	April	RMSE	180.20	124.43	119.40	117.61	127.90	170.01	**105.39**	121.60	126.78	124.60	167.90	125.31	170.29	168.23	176.30
		MAPE (%)	9.36	6.71	**6.42**	6.67	6.88	6.98	6.56	6.78	6.87	6.65	9.10	6.76	10.95	9.17	6.60
	July	RMSE	289.94	150.84	151.06	148.23	154.67	151.41	148.55	141.78	153.22	161.71	153.22	160.82	160.48	151.58	**137.42**
		MAPE (%)	16.84	8.54	8.66	8.50	8.95	8.78	8.59	8.48	9.53	9.12	8.89	9.60	9.27	8.99	**7.91**
	October	RMSE	240.53	210.72	233.48	204.16	218.30	199.61	203.53	203.38	199.77	209.70	147.30	192.74	91.28	108.54	**64.50**
		MAPE (%)	11.54	8.94	10.03	9.33	9.11	7.90	9.32	8.39	8.54	8.22	7.01	8.22	5.22	6.82	**3.54**

Figure 9 shows the comparison result of the original and forecasting load series for one-day-ahead forecasting using the dataset of NSW. The forecasting and original time series show the same pattern, proving that the proposed model has a better accuracy. Additionally, to compare the performance of the proposed model with the existing methods, a nonpara-index statistical test, namely the Friedman rank test [42], is employed. The Friedman rank test is a post-hoc test used to

compare regressor models.

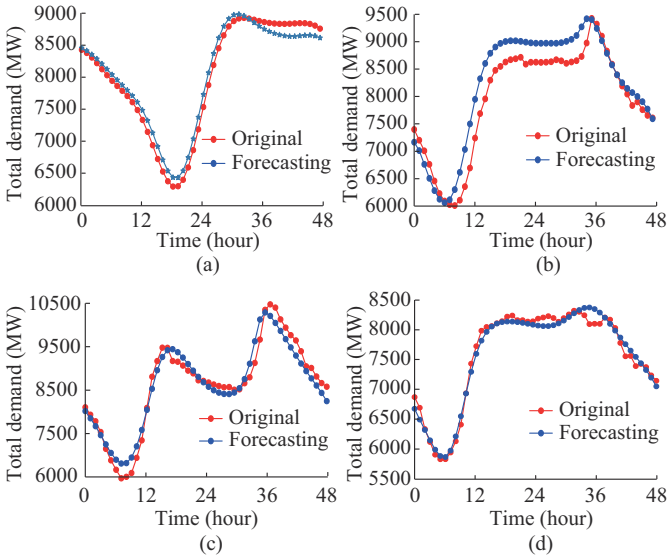

Fig. 9. Original and forecasting load series for one-day-ahead forecasting using NSW data. (a) January. (b) April. (c) July. (d) October.

The half-an-hour- and one-day-ahead forecasting outputs of the Friedman rank test [43], [44] are shown in Fig. 10.

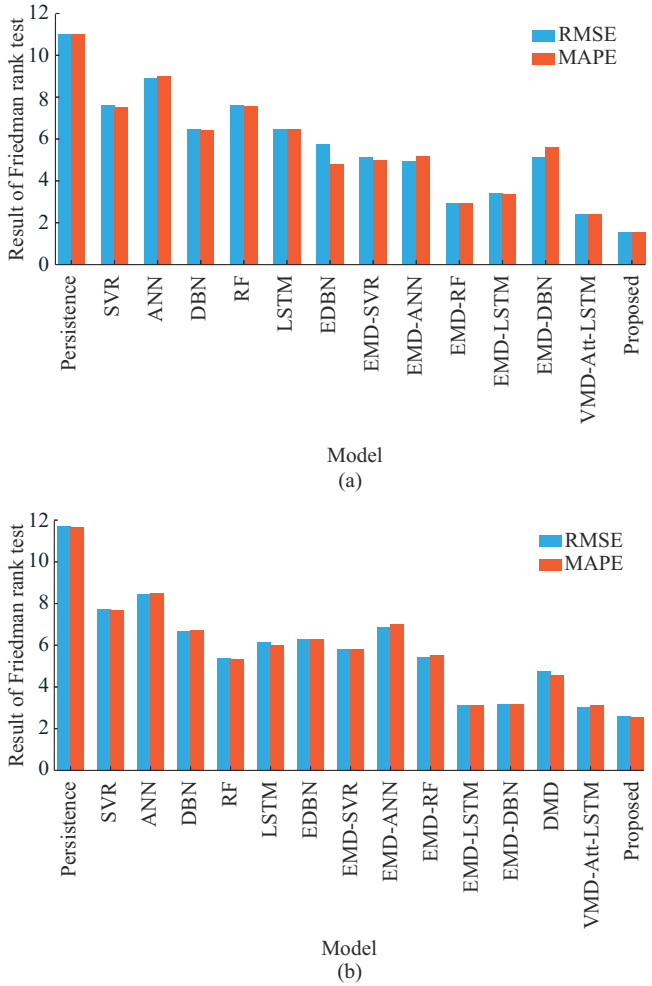

Fig. 10. Forecasting results of Friedman rank test. (a) Half-an-hour-ahead. (b) One-day-ahead.

A high rank implies that the model is not performing satisfactorily. The rank of the proposed model is the lowest, indicating that it is the best model for half-an-hour- and one-day-ahead forecasting. Additionally, p-value is the indication of variation in the procedure. A value smaller than 0.05 implies a high variation. Furthermore, the Wilcoxon signed-rank test [23], [45]-[47] is used to prove the statistical significance of the proposed model and a significance level of $\alpha = 0.05$ is used. Table VI shows the test outputs for the half-an-hour- and one-day-ahead forecasting. The results indicate that the proposed model outperforms other models.

TABLE VI
WILCOXON SIGNED-RANK TEST RESULTS FOR HALF-AN-HOUR- AND ONE-DAY-AHEAD FORECASTING RESULTS

Models for comparison with EMD-Att-LSTM	p-value	
	Half-an-hour	One-day
Persistence	0.0001	0.0001
SVR	0.0001	0.0019
ANN	0.0001	0.0028
DBN	0.0001	0.0045
RF	0.0001	0.0169
LSTM	0.0001	0.0136
EDBN	0.0001	0.0051
EMD-SVR	0.0001	0.0080
EMD-ANN	0.0001	0.0019
EMD-RF	0.0002	0.0137
EMD-LSTM	0.0002	0.0216
EMD-DBN	0.0028	0.0333
DMD		0.0051
VMD-Att-LSTM	0.0074	0.0318

A comparison between the proposed and existing models using half-an-hour- and one-day-ahead forecasting is shown in Fig. 11. The RMSE of the proposed model are lower than those of other models for half-an-hour-ahead forecasting. This can be seen for all the four selected months. Thus, the proposed model outperform the existing advanced models. Improved results can also be seen for one-day-ahead forecasting.

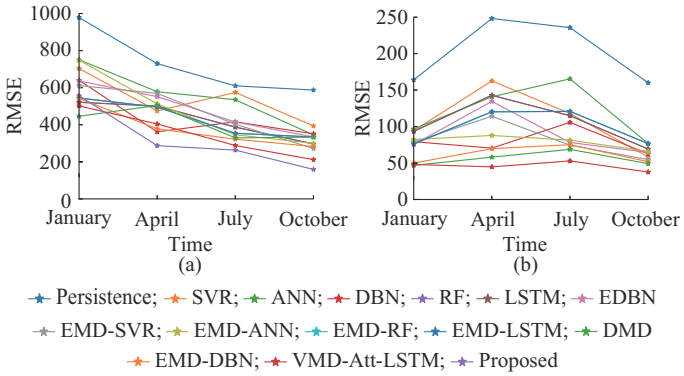

Fig. 11. Comparison of all models based on forecasting results. (a) Half-an-hour-ahead forecasting. (b) One-day-ahead forecasting.

Bar plots are shown in Fig. 12(a) to compare the perfor-

mance of DMD and VMD-Att-LSTM with the proposed model on the NSW dataset considering RMSE. Additionally, to prove the effectiveness of the proposed model, a three-dimensional (3D) plot is drawn between DMD, VMD-Att-LSTM, and the proposed model considering all the five datasets. Consequently, it is clear that the forecasting accuracy of the proposed model is higher than that of DMD. In addition, empirical evaluation shows that the performance of the proposed model is comparable to that of existing advanced models.

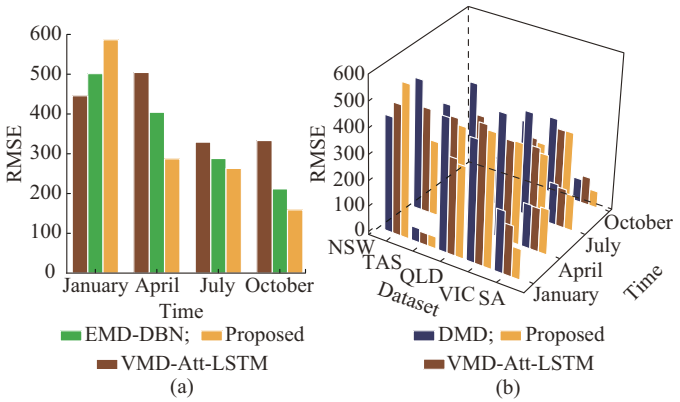

Fig. 12. Comparison of performance of DMD, VMD-Att-LSTM, and proposed method. (a) Using NSW dataset. (b) Using all five datasets.

V. Conclusion

This paper introduces a new approach using a hybrid model comprising EMD and a fine-grained attention mechanism. The introduction of a new EMD approach enables the identification of stable, unstable, and residual terms using the two statistical matrices, i.e., LB and Pearson correlation. The proposed attention mechanism EMD-Att-LSTM provides multiple attention scores for each hidden state $\boldsymbol{h}_t$, which helps obtain a better perspective of the internal structure of each hidden state. The proposed model is evaluated using the AEMO electricity load datasets. The proposed architecture is tested with some existing advanced models such as VMD-Att-LSTM, DMD, EMD-DBN, EMD-LSTM, EMD-RF, EMD-ANN, EMD-SVR, EDBN, RF, LSTM, DBN, ANN, and SVR using MAPE and RMSE. Some of the observations can be summarized as follows.

1) In general, the EMD-Att-LSTM approach outperforms the existing single-structure and hybrid models in half-an-hour- and one-day-ahead forecasting.

2) The proposed model provides the best results considering the Friedman rank test and Wilcoxon signed-rank test results.

3) The proposed architecture can process nonlinear features of the load time series when the forecasting horizon is increased.

References

[1] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199-222, Jan. 1998.

[2] N. Laptev, J. Yosinski, L. Li *et al.*, "Time-series extreme event forecasting with neural networks at uber," *Proceedings of International Conference on Machine Learning*, no. 34, pp. 1-5, Mar. 2017.

[3] L. Ying and M. Pan, "Using adaptive network based fuzzy inference system to forecast regional electricity loads," *Energy Conversion and Management*, vol. 49, no. 2, pp. 205-211, Apr. 2008.

[4] G. Liao, "Hybrid chaos search genetic algorithm and meta-heuristics method for short-term load forecasting," *Electrical Engineering*, vol. 88, no. 3, pp. 165-176, Jul. 2006.

[5] H. Li, S. Guo, C. Li *et al.*, "A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm," *Knowledge-based Systems*, vol. 37, pp. 378-387, Jan. 2013.

[6] K. Methaprayoon, W.-J. Lee, S. Rasmiddatta *et al.*, "Multistage artificial neural network short-term load forecasting engine with front-end weather forecast," *IEEE Transactions on Industry Applications*, vol. 43, no. 6, pp. 1410-1416, Nov. 2007.

[7] D. Chen and M. York, "Neural network based very short term load forecasting," in *Proceedings of 2008 IEEE/PES Transmission and Distribution Conference and Exposition*, Chicago, USA, Apr. 2008, pp. 1-9.

[8] A. R. Reis and A. A. Da Silva, "Feature extraction via multiresolution analysis for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 189-198, Jan. 2005.

[9] W. Xu, H. Peng, X. Zeng *et al.*, "A hybrid modelling method for time series forecasting based on a linear regression model and deep learning," *Applied Intelligence*, vol. 49, no. 8, pp. 3002-3015, Jan. 2019.

[10] X. Qiu, L. Zhang, Y. Ren *et al.*, "Ensemble deep learning for regression and time series forecasting," in *Proceedings of 2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)*, Orlando, USA, Dec. 2014, pp. 1-6.

[11] X. Qiu, Y. Ren, P. N. Suganthan *et al.*, "Empirical mode decomposition based ensemble deep learning for load demand time series forecasting," *Applied Soft Computing*, vol. 54, pp. 246-255, May 2017.

[12] Y. Lin, H. Luo, D. Wang *et al.*, "An ensemble model based on machine learning methods and data preprocessing for short-term electric load forecasting," *Energies*, vol. 10, no. 8, pp. 1-16, Aug. 2017.

[13] S. A. A. Karim and S. A. Alwi, "Electricity load forecasting in UTP using moving averages and exponential smoothing techniques," *Applied Mathematical Sciences*, vol. 7, no. 80, pp. 4003-4014, Apr. 2015.

[14] R. Abdel-Aal and A. Z. Al-Garni, "Forecasting monthly electric energy consumption in eastern Saudi Arabia using univariate time-series analysis," *Energy*, vol. 22, no. 11, pp. 1059-1069, Nov. 1997.

[15] E. D. Tserkezos, "Forecasting residential electricity consumption in greece using monthly and quarterly data," *Energy Economics*, vol. 14, no. 3, pp. 226-232, Jul. 1992.

[16] G. R. Newsham and B. J. Birt, "Building-level occupancy data to improve arima-based electricity use forecasts," in *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-efficiency in Building*, Zurich, Switzerland, Nov. 2010, pp. 13-18.

[17] G. Abledu, "Modeling and forecasting energy consumption in Ghana," *Journal of Energy Technologies and Policy*, vol. 3, no. 12, pp. 1-10, Jan. 2013.

[18] W. Hong, "Electric load forecasting by support vector model," *Applied Mathematical Modelling*, vol. 33, no. 5, pp. 2444-2454, Jul. 2009.

[19] X, Li, J. Lu, L. Ding *et al.*, "Building cooling load forecasting model based on LS-SVM," in *Proceedings of 2009 Asia-Pacific Conference on Information Processing*, Shenzhen, China, Jul. 2009, pp. 55-58.

[20] C. Deb, L. S. Eang, J. Yang *et al.*, "Forecasting energy consumption of institutional buildings in Singapore," *Procedia Engineering*, vol. 121, pp. 1734-1740, Jan. 2015.

[21] M. Rezaeian-Zadeh, S. Zand-Parsa, H. Abghari *et al.*, "Hourly air temperature driven using multi-layer perceptron and radial basis function networks in arid and semi-arid regions," *Theoretical and Applied Climatology*, vol. 109, no. 3-4, pp. 519-528, Feb. 2012.

[22] R. Efendi, Z. Ismail, and M. M. Deris, "A new linguistic out-sample approach of fuzzy time series for daily forecasting of malaysian electricity load demand," *Applied Soft Computing*, vol. 28, pp. 422-430, Mar. 2015.

[23] G Fan, L. Peng, W. Hong *et al.*, "Electric load forecasting by the SVR model with differential empirical mode decomposition and auto regression," *Neurocomputing*, vol. 173, pp. 958-970, Jan. 2016.

[24] N. Neeraj, J. Mathew, M. Agarwal *et al.*, "Long short-term memory-singular spectrum analysis-based model for electric load forecasting," *Electrical Engineering*, vol. 103, no. 2, pp. 1067-1082, Nov. 2020.

[25] X. Zhang and J. Wang, "A novel decomposition-ensemble model for forecasting short-term load-time series with multiple seasonal patterns," *Applied Soft Computing*, vol. 65, pp. 478-494, Apr. 2018.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, Dec. 1997.

[27] R. Yu, S. Zheng, A. Anandkumar *et al*. (2017, Aug.). Long-term forecasting using tensor-train RNNs. [Online]. Available: https://arxiv.org/pdf/1711.00073.pdf
[28] N. Huang, Z. Shen, S. Long *et al*., "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1971, pp. 903-995, Mar. 1998.
[29] K. Cho, B. Van Merriënboer, D. Bahdanau *et al*. (2014, Sept.). On the properties of neural machine translation: encoder-decoder approaches. [Online]. Available: https://aclanthology.org/W14-4012/
[30] Australian energy market operator AEMO. (2013, Dec.). [Online]. Available: http://www.aemo.com.au/=0pt
[31] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358-386, Mar. 2005.
[32] H. Choi, K. Cho, and Y. Bengio, "Fine-grained attention mechanism for neural machine translation," *Neurocomputing*, vol. 284, pp. 171-176, Apr. 2018.
[33] J. Benesty, J. Chen, Y. Huang *et al*., "Pearson correlation coefficient," *Noise Reduction in Speech Processing*, vol. 2, pp. 1-4, Mar. 2009.
[34] A. Paszke, S. Gross, F. Massa *et al*., "Pytorch: an imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, pp. 8026-8037, Dec. 2019.
[35] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, Mar. 1995.
[36] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2, no. 3, pp. 273-293, Jan. 2004.
[37] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, Jul. 2006.
[38] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, Jan. 2001.
[39] Y. Lin and L. Peng, "Combined model based on EMD-SVM for short-term wind power forecasting," *Proceedings of the CSEE*, vol. 31, no. 31, pp. 102-108, Nov. 2011.
[40] H. Liu, C. Chen, H. Tian *et al.*, "A hybrid model for wind speed forecasting using empirical mode decomposition and artificial neural networks," *Renewable Energy*, vol. 48, pp. 545-556, Dec. 2012.
[41] N. Mohan, K. Soman, and S. S. Kumar, "A data-driven strategy for short-term electric load forecasting using dynamic mode decomposition model," *Applied Energy*, vol. 232, pp. 229-244, Dec. 2018.
[42] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1-30, Jan. 2006.
[43] Z. Zhang and W. Hong, "Electric load forecasting by complete ensemble empirical mode decomposition adaptive noise and support vector regression with quantum-based dragonfly algorithm," *Nonlinear Dynamics*, vol. 98, no. 2, pp. 1107-1136, Sept. 2019.
[44] Z. Zhang, S. Ding, and Y. Sun, "A support vector regression model hybridized with chaotic krill herd algorithm and empirical mode decomposition for regression task," *Neurocomputing*, vol. 410, pp. 185-201, Oct. 2020.
[45] E. A. Gehan, "A generalized wilcoxon test for comparing arbitrarily singly-censored samples," *Biometrika*, vol. 52, no. 1-2, pp. 203-224, Jun. 1965.
[46] M. Li, J. Geng, W. Hong *et al*., "Periodogram estimation based on LSSVR-CCPSO compensation for forecasting ship motion," *Nonlinear Dynamics*, vol. 97, no. 4, pp. 2579-2594, Jul. 2019.
[47] W. Hong, "Electric load forecasting by seasonal recurrent SVR (support vector regression) with chaotic artificial bee colony algorithm," *Energy*, vol. 36, no. 9, pp. 5568-5578, Sept. 2011.

Neeraj received the B. Eng. degree in computer science and engineering from Gautam Budh Technical University, Uttar Pradesh, India, in 2013, and the M. Tech. degree in computer science and engineering from the National Institute of Technology (NIT), Durgapur, India, in 2016. He is currently pursuing the Ph.D. degree in computer science and engineering at the Indian Institute of Technology, Patna, India. His research interests include time series analysis, brain computer interface (BCI), and deep learning systems.

Jimson Mathew received the master's degree in computer engineering from Nanyang Technological University, Singapore, in 1999, and the Ph. D. degree in computer engineering from the University of Bristol, Bristol, U.K., in 2008. He is currently an Associate Professor and Head of the Department of Computer Science and Engineering, Indian Institute of Technology, Patna, India. His research interests include fault-tolerant computing, architectures for machine learning systems, deep learning systems, and Internet of Things (IoT) systems.

Ranjan Kumar Behera received the B. Eng. degree in electrical engineering from the Regional Engineering College (NIT), Rourkela, India, in 1998, and the M. Tech. and Ph. D. degrees from the Indian Institute of Technology, Kanpur, India, in 2003 and 2009, respectively. He was a Visiting Scholar in the Energy Systems Research Center, Tennessee Technological University, Nashville, USA, in 2008. He is currently an Associate Professor at the Department of Electrical Engineering, Indian Institute of Technology, Patna, India. In July 2016, he visited as a Research Collaborator at the Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Pretoria, South Africa. His research interests include nonlinear control theory application to power electronic converters, pulse width modulation techniques, multi-phase electric drive control, and battery charging infrastructure for electric vehicles.