Applied Energy 283 (2021) 116328

Contents lists available at ScienceDirect

Applied Energy

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

Multi-temporal-spatial-scale temporal convolution network for short-term load forecasting of power systems

Linfei Yin [a,*], Jiaxing Xie [a]

[a] College of Electrical Engineering, Guangxi University, Nanning, Guangxi 530004, China

HIGHLIGHTS

- Time series and nonlinear relationship of load data are simultaneously considered.
- The multi-temporal-spatial-scale method is introduced into the proposed method.
- A multi-temporal-spatial-scale temporal convolutional network is proposed.
- The proposed method can grasp the trends and laws of load data of power systems.
- Short-term load forecasting can be solved by the proposed method with high accuracy.

ARTICLE INFO

Keywords:
Multi-temporal-spatial-scale temporal convolution network
Short-term load forecasting
Dilated causal convolution
Residual connection

ABSTRACT

With the advancement of power market reform, accurate load forecasting can ensure the stable operation of power systems increasingly. The randomness of feature change such as climate and day type increases the complexity of short-term load forecasting. To simplify the data processing process to facilitate the practical application and predict short-term loads more accurately, this paper takes the past load data as a feature and considers the time series characteristics of load data simultaneously. The multi-temporal-spatial-scale method is applied to process the load data by reducing the load data noise error and enhancing the time series characteristics. Then, a novel short-term load forecasting model, which is named a multi-temporal-spatial-scale temporal convolutional network, is applied to load forecasting tasks in this paper. The proposed approach can learn the nonlinear feature and time series characteristics of load data simultaneously. To predict the power load of a city in Guangxi Zhuang Autonomous Region (China) in the next day and the next week, the forecasting model is trained by the historical feature load of 7 days, 21 days, 99 days, and 199 days. Compared with 22 artificial intelligent short-term load forecasting models, such as backpropagation neural network and bagging regression, the simulation results show that the proposed multi-temporal-spatial-scale temporal convolutional network can obtain higher accuracy for the short-term load forecasting of power systems than other compared methods.

1. Introduction

Short-term load forecasting (STLF) means load forecasting in the next day and week generally and is mainly applied for tasks such as economic operation planning and short-term maintenances of power systems [1]. With the advancement of power market reform, the increasing of distributed energy sources and randomness load brought challenges for predicting load value precisely. The result of the STLF is an important basis for the power market to determine liquidation price [2]; the accurate load forecasting can greatly reduce the operating cost of power grids. Therefore, high accuracy STLF results can reduce electricity costs significantly [3].

Generally, a data-driven STLF model contains two categories roughly: time series model and feature learning model. Autoregressive integrated moving average model (ARIMA), which is a time series model, has been applied to carry out power systems load forecast [4]; a bagging ARIMA has been combined with an exponential smoothing method for mid-long-term power load forecasting [5]. In recent years, long short-term memory neural network (LSTM) has been applied to solve the regression problems. An improved LSTM model considering relevant factors has been employed in STLF tasks [6]. The wavelet model has been proposed to solve the probabilistic load forecasting problems.

* Corresponding author.
E-mail address: yinlinfei@163.com (L. Yin).

https://doi.org/10.1016/j.apenergy.2020.116328
Received 23 September 2020; Received in revised form 12 November 2020; Accepted 27 November 2020
Available online 10 December 2020

The wavelet model has been trained by generalized extreme learning machine models [7]. The feature learning model is a type of model that can learn significant features from original data for regression and classification tasks, such as tree model and deep neural networks. The artificial neural network has been applied to carry out load forecasting research [8]; a deep residual network based on feature extraction has been utilized in load forecasting problems [9]; Yeo-Johnson transformation quantile algorithm has been proposed for load forecasting [10]; a novel hybrid model [11] and Gaussian process quantile regression [12] have been proposed to predict power load; clustering method has been applied to solve the load forecasting of residential electricity [13]; ensemble incremental learning has been applied in load forecasting problems [14]; bi-square kernel regression has been proposed for load forecasting task; phase space reconstruction [15] and multi-objective optimization hybrid model [16] have been proposed to predict power load; a K-means grey relational analysis-Elman model has been applied to the STLF of photovoltaic power [17]; an ensemble method has been proposed to predict wind power [18]; a robust short-term electrical load forecasting framework that can capture variations in building operation has been proposed to predict power loads [19].

Without the ability to learning nonlinear mapping, the nonlinear relationship of load data cannot be considered by a time series model. The feature learning model predicts the load value by extracting the feature of input data and fitting the output load. Therefore, the feature learning model does not consider the time series characteristics of load data. The load data not only has time series characteristics but also has a nonlinear mapping relationship. Therefore, a model considering the nonlinear relationship and the time series characteristics of load data simultaneously, which can predict the load change trend more accurately, is proposed in this paper.

To learn time series characteristics and the nonlinear relationship of load data simultaneously and to improve forecasting accuracy, a multi-temporal-spatial-scale temporal convolutional network (MTCN) is proposed for STLF models. Firstly, a multi-temporal-spatial-scale (MTSS) method, which can divide the original data according to multiple temporal and spatial scales, is applied to process the sample data for reducing data noise error and enhancing the time series characteristics of the data. Secondly, the dilated causal convolution layer of the MTCN has the function of information memory and can learn time series characteristics. The stack of residual blocks is employed to form the network structure of the MTCN. The output layer selects the convolution layer instead of the fully connected layer to form a fully-convolutional network. Since the structure has a strong nonlinear mapping ability, the MTCN can predict short-term load more accurately. The network structure characteristics of the MTCN are listed as follows.

(1) Fully-convolutional network structure. The MTCN replaces traditional neural networks with convolution layers. The fully-convolutional network structure can ensure that the output results dimensions of the networks are consistent with the inputs data dimensions. The MTCN can theoretically calculate the inputs of any size and can produce outputs for corresponding spatial dimensions.
(2) Dilated causal convolution layer. Since causal convolution stipulates that the outputs of the current time only perform convolution operation with the neurons of the previous time of the previous layer and the current time of the previous layer, the network can remember information. Therefore, the MTCN can learn the time series characteristics of load data. The dilated convolution has a larger receptive field than the normal convolution if the same size of the convolution kernel of a normal convolution is equal to that of a dilated convolution. The networks of the dilated convolution can remember longer information and can reduce the training complexity with the increasing of network depth.
(3) Residual block-stacking structure. Since the residual block-stacking structure of the MTCN can greatly increase the network depth, the learning ability of the MTCN can be greatly improved. Therefore, although the data sample is very large, the MTCN can stably be trained without vanishing gradient and exploding gradient problems.
(4) The MTSS module. Since the MTSS of the MTCN can reduce the noise error of load data by load division, the error of the training sample is smaller. The length of load data is varied by the MTSS method. Since the MTSS highlights time series characteristics, the MTCN can grasp the trends and laws of load data change.

The following sections of this paper are arranged as follows. Section 2 introduces the principle of the MTCN model. Section 3 forecasts historical data of a city in Guangxi Zhuang Autonomous Region (China). Section 4 gives a summary of this paper.

2. Multi-temporal-spatial-scale temporal convolution network

2.1. Multi-temporal-spatial-scale method of multi-temporal-spatial-scale temporal convolution network

The MTSS method is a data processing method that subdivides the original data according to multiple temporal and spatial scales. The MTSS method is particularly suitable for processing load data that is greatly affected by the temporal and spatial distribution of load data. The load data is divided into multiple spatial scales and temporal scales, which are applied to train the forecasting model. The MTSS method can reduce the noise error of load data, can enhance the time series characteristics, and can improve prediction accuracy.

The spatial scale of the load data can be divided by electricity types. Electricity types include industrial electricity, commercial electricity, and other electricity. The total load data of the area and the divided load data of the subsidiary areas can be simultaneously obtained from multiple spatial scales load data. The temporal scale of the load data is divided into multiple types according to the length of the prediction period. For example, the sampling time scale of the mid-term and long-term should refer to “day”, “week”, “month” and “quarter”; the sampling time scale of short-term and ultra-short-term should refer to “second”, “minute” and “hour”. The MTSS method samples the load data of the same period and forms load data with multiple temporal scales.

2.2. Forecasting process of multi-temporal-spatial-scale temporal convolution network

The MTCN contains an information memory structure, which can reflect the relationship of load data in time series and can represent the nonlinear mapping function. To apply the MTCN into STLF, the preprocessing of load data and the designing of the MTSS network structure should be considered (Fig. 1). After the original load data is divided into multiple regions and the load data of the same day is sampled at multiple temporal scales, the MTSS load data is formed. Then, construct the MTCN into the STLF model, and determine the inputs and outputs variables of the MTCN. The MTCN training model is built corresponding to the inputs and outputs structures of the MTSS method. Then the iterations are set and the model is trained, each training epoch applies the validation data to calculate the mean absolute percentage error and saves the best model with a smaller mean absolute percentage error. When all iterations are completed, the best model is outputted as the test model of the experiment.

The experiment of this paper predicts the load value of the next day and week. For the prediction experiment of the power load data of the next day: the input data of the MTCN x is n days power load data before the forecasting day; the data length m is determined by the sampling temporal scale; the output data of the MTCN $\widehat{y}$ is the real load data on

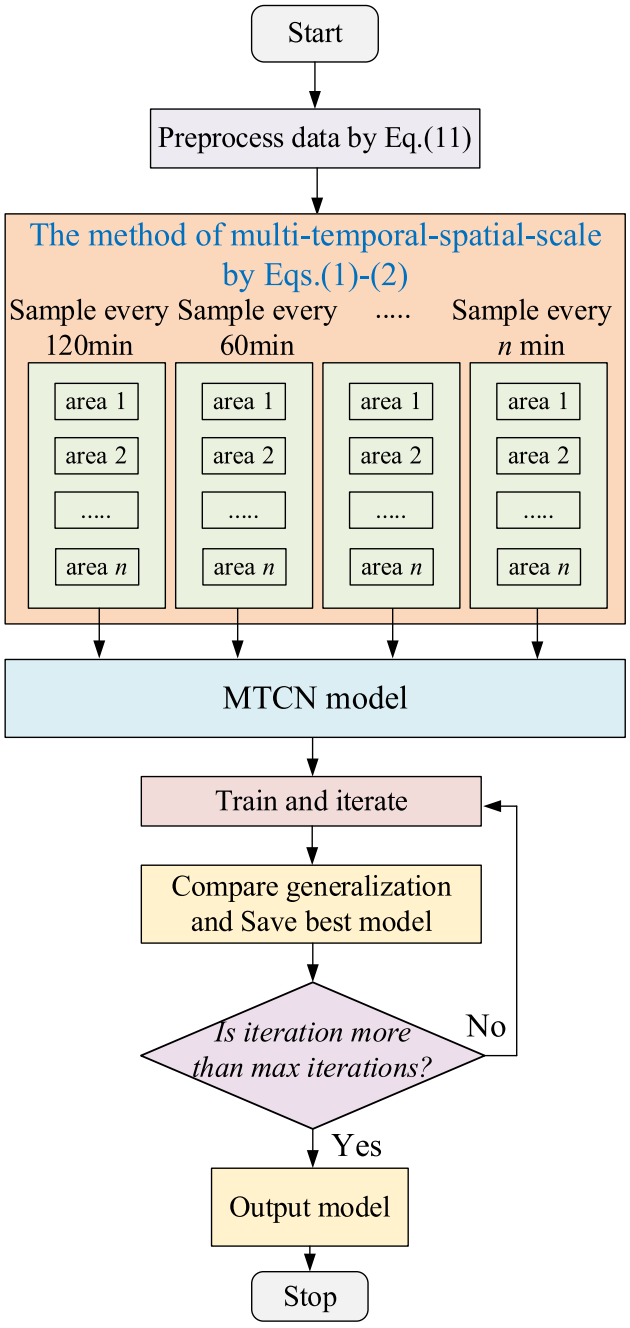

Fig. 1. Flowchart of load forecasting based on multi-temporal-spatial-scale temporal convolution network.

the forecasting day; $\{x,\hat{y}\}$ is a set of training sample; where x and $\hat{y}$ are structured as Eqs. (1) and (2), respectively.

$$x = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \tag{1}$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_m \end{bmatrix} \tag{2}$$

For example, the original load data is divided into three subsidiary areas with sampled every 60 min and every 15 min; then, the three-spatial scales and two-temporal scales load data are formed. Set the total load of the area sampled every 60 min to be x_{24}, the load data of these three subsidiary areas which sampled every 60 min are x_{24_1}, x_{24_2}, and x_{24_3}. Set the total load of the area sampled every 15 min to be x_{96}, the load data of these three areas which sampled every 15 min are x_{96_1}, x_{96_2}, and x_{96_3}. These total loads in this example can be described as

$$x_{24} = x_{24\_1} + x_{24\_2} + x_{24\_3} \tag{3}$$

$$x_{96} = x_{96\_1} + x_{96\_2} + x_{96\_3} \tag{4}$$

Matrices x_{24_1}, x_{24_2}, and x_{24_3} are defined as the same data shapes; taking x_{24_1} as an example, the data shape can be presented as

$$x_{24\_1} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{124} \\ x_{21} & x_{22} & \cdots & x_{224} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{n24} \end{bmatrix} \tag{5}$$

Similarly, matrices x_{96_1}, x_{96_2}, and x_{96_3} are defined as the same data shapes; taking x_{96_1} as an example, the data shape can be described as

$$x_{96\_1} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{196} \\ x_{21} & x_{22} & \cdots & x_{296} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{n96} \end{bmatrix} \tag{6}$$

The original load data is divided into six groups of training data according to the MTSS method. Although the lengths of these six groups of load data are not uniform, the flexible structure of the MTCN can accommodate these data as the inputs of the training model.

Similarly, for the prediction experiment of the power load data of the next week: the model input data x is n days power load data before the forecasting days; the data length m is determined by the sampling temporal scale. The model output data $\hat{y}$ is the real load data on the predicted week. The shape of data x is the same as that of the experiment load data of the next day. Thus, the shape of $\hat{y}$ can be presented as

$$\hat{y} = \begin{bmatrix} \hat{y}_{11} & \hat{y}_{12} & \cdots & \hat{y}_{1m} \\ \hat{y}_{21} & \hat{y}_{22} & \cdots & \hat{y}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{y}_{71} & \hat{y}_{72} & \cdots & \hat{y}_{7m} \end{bmatrix} \tag{7}$$

In the same way, the data shape of each group of load data for the MTSS method is set to the same as the data shape of the experiment of forecasting the load data of the next week.

The proposed MTCN method provides six sets of predicted output data. These six sets of predicted data are integrated to predict the load forecasting results. Then, the predicted data every 60 min of these three areas are added as $\hat{y}_{24}$ by Eqs. (3)-(4). After that, the predicted data every 15 min of these three areas are added as $\hat{y}_{96}$ by Eqs. (3)-(4). Afterward, sample the data corresponding to 24 whole times of $\hat{y}_{96}$ as a new data set $\hat{y}'_{24}$ by Eq. (8). Finally, average $\hat{y}_{24}$ and $\hat{y}'_{24}$ as the final load forecasting result by Eqs. (8)-(9).

$$\hat{y}'_{24} = sampling(\hat{y}_{96}) \tag{8}$$

$$\hat{y} = mean(\hat{y}_{24}, \hat{y}'_{24}) \tag{9}$$

For example, if the seven-day historical load data is applied to predict the 24-hour load value in the next day, the original load data should be grouped into a group of eight consecutive days. The first seven days of each group of data are input data, and the eighth day is output data. The input data is three spatial scale data and two temporal scale data, and the output is total load data. The x^1_{nm} is the load value of the first area; the x^2_{nm} is the load value of the second area; the x^3_{nm} is the load value of the third area. The shape of input and output can be presented as

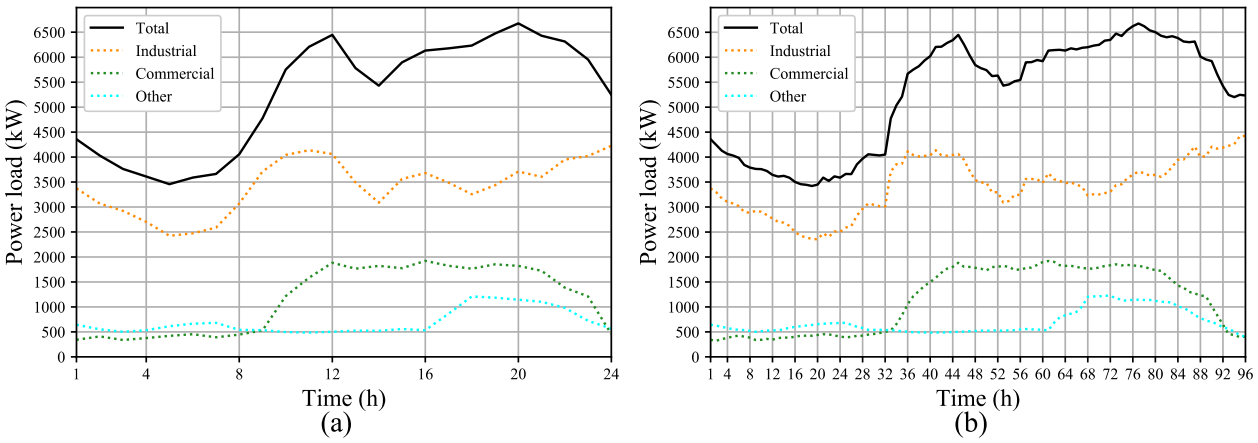

Fig. 2. Daily load curves of multi-temporal-spatial-scale load data.

$$input = \begin{bmatrix} x_{11}^1 & x_{12}^1 & \cdots & x_{124}^1 \\ x_{21}^1 & x_{22}^1 & \cdots & x_{224}^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_{71}^1 & x_{72}^1 & \cdots & x_{724}^1 \\ x_{11}^2 & x_{12}^2 & \cdots & x_{124}^2 \\ x_{21}^2 & x_{22}^2 & \cdots & x_{224}^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_{71}^2 & x_{72}^2 & \cdots & x_{724}^2 \\ x_{11}^3 & x_{12}^3 & \cdots & x_{124}^3 \\ x_{21}^3 & x_{22}^3 & \cdots & x_{224}^3 \\ \vdots & \vdots & \ddots & \vdots \\ x_{71}^3 & x_{71}^3 & \cdots & x_{724}^3 \\ x_{11}^1 & x_{12}^1 & \cdots & \cdots & x_{195}^1 & x_{196}^1 \\ x_{21}^1 & x_{22}^1 & \cdots & \cdots & x_{295}^1 & x_{296}^1 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ x_{71}^1 & x_{72}^1 & \cdots & \cdots & x_{795}^1 & x_{796}^1 \\ x_{11}^2 & x_{12}^2 & \cdots & \cdots & x_{195}^2 & x_{196}^2 \\ x_{21}^2 & x_{22}^2 & \cdots & \cdots & x_{295}^2 & x_{296}^2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ x_{71}^2 & x_{72}^2 & \cdots & \cdots & x_{795}^2 & x_{796}^2 \\ x_{11}^3 & x_{12}^3 & \cdots & \cdots & x_{195}^3 & x_{196}^3 \\ x_{21}^3 & x_{22}^3 & \cdots & \cdots & x_{295}^3 & x_{296}^3 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ x_{71}^3 & x_{72}^3 & \cdots & \cdots & x_{795}^3 & x_{796}^3 \end{bmatrix} \tag{10}$$

$$output = \begin{bmatrix} x_{81} & x_{82} & \cdots & x_{824} \end{bmatrix} \tag{11}$$

The original daily load curves of input data are displayed by three spatial scales and two temporal scales data (Fig. 2), the orange line represents the daily load curve of industrial electricity; the green line represents the daily load curve of commercial electricity; the blue line represents the daily load curve of other electricity; the black line represents the daily load curve of total electricity.

There are no factors such as day type and climate in the training data. Firstly, the change of load data has a strong periodically law if the period of the training data is long enough; the model can learn the historical change law of the load data at the same time-point and can accurately predict the load value. Secondly, the electricity consumption of some companies is almost not affected by climate and day type (such as server company, the load fluctuation is very small), thus, this kind of company does not need to collect climate and day type data as training data. Thirdly, if the model only needs historical load data to predict load value, this kind of model reduces the storage cost of data and simplifies the application process for large-scale load forecasting.

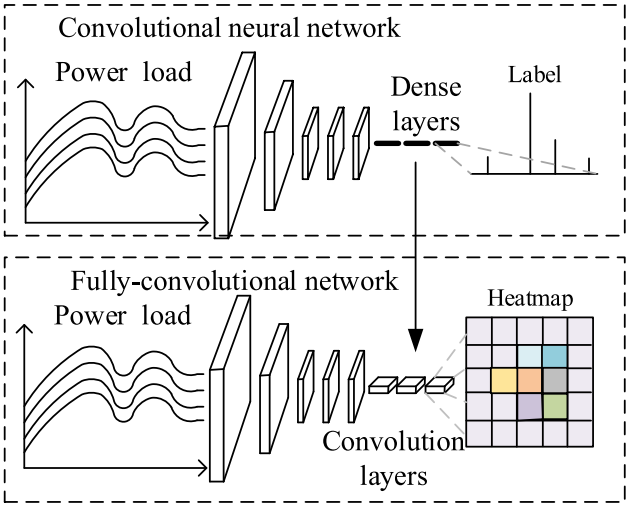

Fig. 3. Schematic diagram of convolutional neural network and fully-convolution network structure.

The MTCN is a special convolutional neural network that can process time series [20]. The MTCN applies one-dimensional causal convolution and dilates convolution as standard convolutional layers. Then, each residual module encapsulates these two standard convolutional layers, i.e., one identity mapping layer and one rectified linear unit function layer. After that, the residual module stacks up a deep network. Afterward, the MTCN employs convolutional layers in the last few layers instead of fully connected layers.

2.3. *Fully-convolution network structure of multi-temporal-spatial-scale temporal convolution network*

To ensure that the outputs and inputs have the same lengths, the MTCN replaces the dense layer with a convolutional layer to construct a fully-convolutional network structure [21]. The lengths of the hidden layer and the input layer of a fully-convolution network structure are the same; and then the "zero paddings" is applied to ensure that the subsequent network layers have the same numbers.

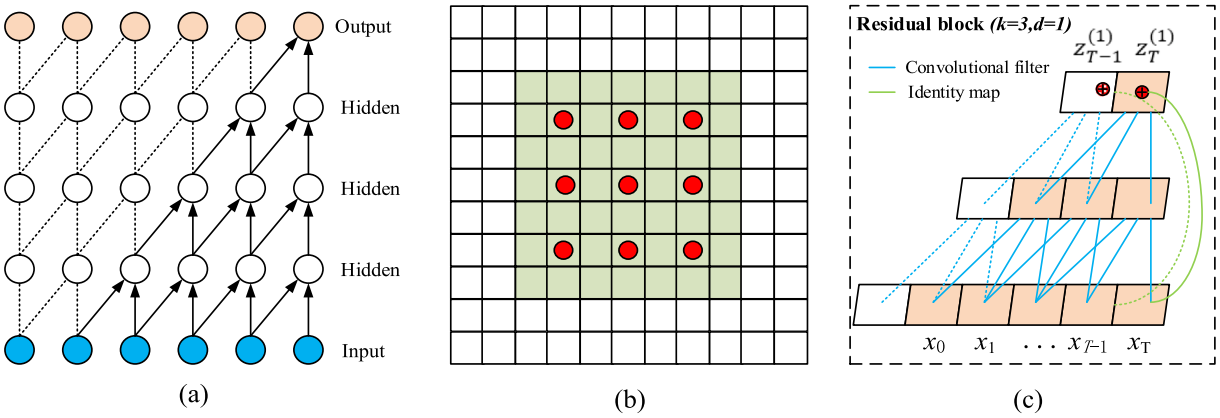

Fig. 4. Architectural elements in multi-temporal-spatial-scale temporal convolution network: (a) schematic of causal convolution; (b) dilated convolution; (c) residual connection.

The convolution layer neurons of the convolution kernel are only connected to the part of input data and can share the parameters of the weights of the input data and the convolution kernel. The neurons of the dense layer and the convolution layer have the same function forms. Therefore, the dense layer and convolution layer can be converted into each other.

Although the final output data dimensions of the dense layer and convolutional layer are the same, the concept and calculation process of the convolution layer and dense layer are different. The weights and biases trained by the convolutional neural network (CNN) are applied in the convolution training processes. Fig. 3 is an example of CNN and a fully-convolutional network.

The upper part of Fig. 3 is a CNN; the last three layers are dense; the output data is one-dimensional. The lower part of Fig. 3 is a fully-convolution network; the last three layers are convolution layers. The output of fully-convolutional networks can be configured as two-dimensional or even higher-dimensional data, which aims to match the dimensions of the input data and ensure that the output results dimensions are consistent with the input data dimensions. Fully-convolution networks can theoretically calculate the inputs of any size and can produce outputs corresponding to spatial dimensions.

2.4. Causal convolution of multi-temporal-spatial-scale temporal convolution network

To process time series data, a causal convolution is applied into the MTCN [22]. Therefore, the output at time t of causal convolution is convolved with the element at time t and earlier.

After the length of time series data x_T is set to be T, the time series data is configured as an input sample and is applied to predict the corresponding output y_T value at each moment. The prediction process can be expressed as

$$y_0, \cdots, y_T = f(x_0, \cdots, x_T) \tag{12}$$

Therefore, y_t is determinate by $x_0, \cdots, x_t$ rather than $x_{t+1}, \cdots, x_T$; the goal of the training process aims to fit a function $f(\cdot)$ that minimizes actual and predicts output errors. Causal convolution considering the time series is generated by restricting convolution operation (Fig. 4(a)). The input data enters from the blue layer in Fig. 4(a); and the time series is from left to right. The element of the next layer at time t is only convolved with the inputs of the previous layer at time t and $t-1$; and the outputs can be provided by the output layer through three hidden layers.

Although the causal convolution of the MTCN can learn time series data and can solve the problem of "information leaks" caused by a conventional convolution operation. Thus, the causal convolution must remember longer information with the increasing of time series data. Consequently, the model complexity is increased with the number of convolutional layers. The vanishing gradient problem during the training process will cause the instability of training. To solve this problem, dilated convolution is introduced into the MTCN and is applied in conjunction with causal convolution.

2.5. Dilated convolution of multi-temporal-spatial-scale temporal convolution network

Compared with ordinary convolution, (i) the dilated convolution (Fig. 4(b)) of the MTCN [23] has dilation coefficients that can change the size of dilation; dilated convolution is similar to ordinary convolution in the size of the convolution kernel; (ii) the dilated convolution of the MTCN has larger receptive fields [24]. The receptive field exponentially is depending on the dilation coefficients rather than the number of convolution kernels.

The dilation is the addition of a fixed step between filters. Dilated convolution is just an ordinary convolution if $d = 1$. The dilated convolution, which can let the outputs layer consider a larger range of inputs, can effectively expand the receptive fields of convolution kernels. The filter coefficient k and dilation coefficient d of convolution kernels can expand the receptive fields as

$$F(s) = \sum_{i=0}^{k-1} f(i)\hat{\mathrm{A}}\cdot x_{s-d\hat{\mathrm{A}}\cdot i} \tag{13}$$

where d means dilation coefficient; k means filter coefficient; s-$d\hat{\mathrm{A}}\cdot i$ is the past direction of input data.

With the expansion of the dilated convolution of the MTCN, the MTCN can fit each input and can remember very long-term input information. Then, dilated convolution improves training efficiency.

2.6. Residual connection of multi-temporal-spatial-scale temporal convolution network

Increasing the network depth can improve the network expression ability of the proposed model. However, the value of gradient information could "disappear" in propagation if the network layer is extremely deep. The deep residual network, which increases the network depth, can solve the problem of "gradient disappearing" during network training [9]. The MTCN is stacked by the residual connection models. And each residual block contains a branch, which is utilized to learn conversion function $F(x)$. The output of the conversion function is connected to the input of the next residual block, which can be described as

$$o = \mathrm{activation}(x + F(x)) \tag{14}$$

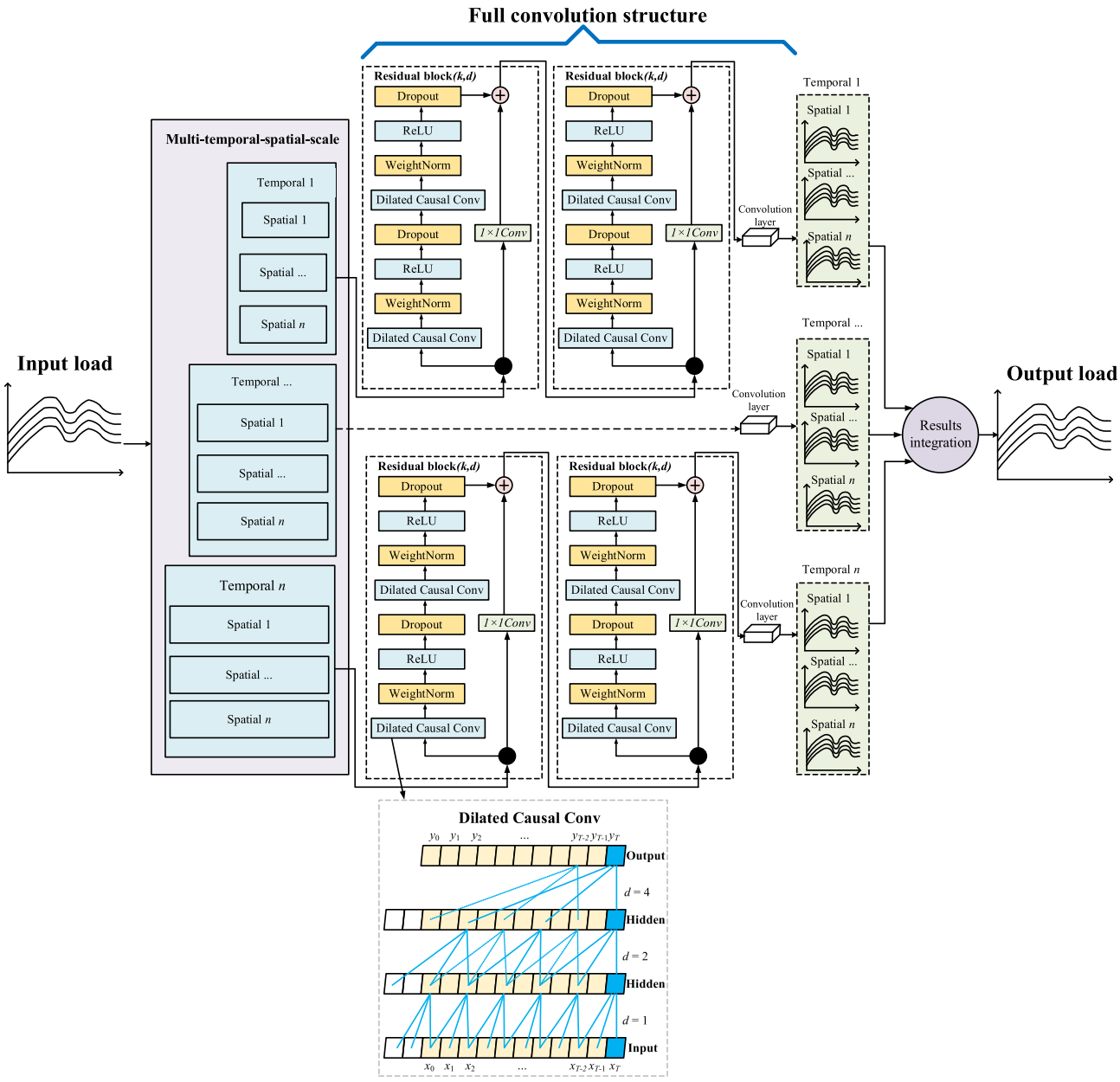

Fig. 5. Steps of multi-temporal-spatial-scale temporal convolution network.

Fig. 4(c) shows a residual connection diagram with convolution kernel size $k = 3$ and dilation coefficient $d = 1$. The size of the receptive field of the MTCN depends on the depth n of the network, the size k of the filter, and the dilation factor d. Therefore, the network depth and structural complexity of the MTCN should be increased with the increasing of longer remembered input information.

Each residual block has two dilated causal convolution layers. These two layers are normalized with the rectified linear unit activation function [25]. The convolution filter is weighted with normalization [26]. Besides, a dropout layer [27] is added to the dilated convolution for regularization. Since the inputs and outputs in MTCN have different widths, the inputs cannot be directly added to the outputs. Considering the difference between the widths of the inputs and the outputs, a one-dimensional convolution is added to each residual block for ensuring that the adder layers can receive the data with the same shapes.

2.7. Multi-temporal-spatial-scale temporal convolution network based short-term load forecasting model

An MTCN model includes an input data module, an MTSS module, residual blocks, convolution layers, an MTSS method, a result integration module, and an output data module. After the preprocessing, the load data is scaled by the MTSS method. Then, the calculated load forecast value can be provided by two layers of the residual block and a convolution layer (Fig. 5).

The load sample data that has been preprocessed by data cleaning and normalization is set to be the inputs of the input data module; and these load sample data can be directly applied for model training. The MTSS module aims to divide load sample data into multiple sets of sub-samples with different temporal scales and spatial scales. Two residual blocks and a convolution layer are formed as a fully-convolution network structure, which can memorize the time series characteristics and can fit the nonlinear relationship in the sample data. In theory, the more residual blocks, the nonlinear fitting ability of the MTCN is stronger.

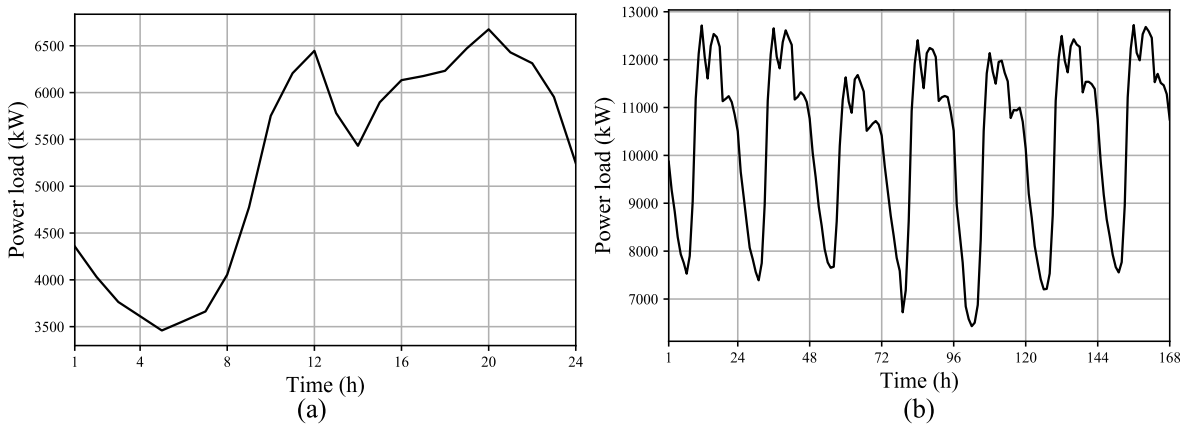

Fig. 6. Curves of original load data distribution: (a) 24 h load; (b) 168 h load.

3. Experiments and results

The MTCN is developed in a personal computer with Intel Core i5-6300HQ CPU, NVIDIA GeForce GTX 960 M GPU, and 8.0 GB RAM with Pytorch (version 1.4.0), which is a deep learning framework [28].

In this experiment, the power load data from the year 2008 to 2019 about 4300 days in a city of Guangxi Zhuang Autonomous Region (China) is utilized to verify the feasibility and effectiveness of the proposed MTCN. The power load data refers to some part of the electricity usage load collected from the company and the community. Some electricity consumers collect load data every 15 min; some electricity consumers collect load data every 60 min. The original load data distribution of a day and a week is given in Fig. 6. Fig. 6(a) is the daily load curve of the 2507th day in the region in experiment datasets; Fig. 6(b) is the daily load curve of the 4264th day in the region in experiment datasets. The load data distribution of a day and a week (Fig. 6) shows that the load data greatly fluctuates in one day and has two peaks. The load data regularly fluctuates within a week.

In the MTCN model, the load data is divided into three spatial scales by electricity types containing industrial electricity, commercial electricity, and other electricity. These two temporal scales of the inputs of the MTCN are sampled every 60 min and every 15 min, respectively.

The simulations contain two parts: one aims to predict the power load at 24 time-points in the next day. The load data of the first two thousand days are set as the training data. The load data from the 2000th day to the 2200th day are set as the validation data. The 2507th day load data are set as the test data. The training data are applied to build training samples according to the different dimensions of input data. When the input data is the 7-day historical load and the output data is the historical load of the eighth day, the input data dimension is 7, and 275 groups of training samples can be constructed. Similarly, when the input data is the 199-day historical load and the output data is the historical load of the 200th day, the input data dimension is 199, and 11 groups of training samples can be constructed. The other one aims to predict the power load at 168 time-points in the next week. The load data of the first four thousand days is set as the training data. The load data from the 4000th day to the 4200th day are set as the validation data. The 4264th day load data are set as the test data. When the input data is the 7-day historical load and the output data is the historical load of the eighth day, the input data dimension is 7, and 500 groups of training samples can be constructed. Similarly, when the input data is the 199-day historical load and the output data is the historical load of the 200th day, the input data dimension is 199, and 20 groups of training samples can be constructed. Each part is divided into four subsidiary parts according to the number of input data feature dimensions. The historical load of one day is set as a one-dimensional feature. Because the power load is cyclical, the minimum cycle of the daily load is “week”, the minimum input dimension is selected as 7. Because of the limitation of time complexity and space complexity, the input data dimension should not be too large. Combined with the size of the experimental data, the maximum input dimension is selected as 199. In the upper and lower bounds, two types with moderate complexity of 21 and 99 dimensions are selected. Four types (i.e., 7-dimension feature, 21-dimension feature, 99-dimension feature, and 199-dimension feature) are selected to be simulated separately. Therefore, four MTCNs are needed in each case. In addition, a total of 22 STLF algorithms are compared with the MTCN: (1) back propagation neural network (BP); (2) back propagation neural network optimized by ant colony algorithm (ABP); (3) long short-term memory neural network (LSTM); (4) linear regression (LR); (5) ridge regression (RD); (6) Bayesian ridge regression (BR); (7) automatic relevance determination regression (ARD); (8) Huber regression (HB); (9) Theisen regression (TS); (10) gradient descent regression (SGD); (11) random forest (RF); (12) k-nearest neighbor regression (KN); (13) adaptive boosting regression (AB); (14) gradient boosting regression (GB); (15) bagging regression (BAG); (16) extreme random tree regression (ET); (17) extreme gradient boosting regression (XGB); (18) kernel ridge regression (KR); (19) support vector machine (SVM); (20) radial basis neural network (RBF); (21) gaussian process regression (GP) and (22) temporal convolution network (TCN). The sample data is uniformly preprocessed with the min–max normalization. Except for the proposed MTCN model, other algorithms do not apply the MTSS module.

The simulation results are uniformly compared with four forecasting performance indices: mean absolute percentage error (MAPE), root mean squared error (RMSE), median absolute error (MAE), and R-squared (R^2). In load forecasting tasks, the RMSE and the MAPE are the two most common performance indices. The RMSE is widely applied in the result evaluation of various regression tasks. However, this RMSE is sensitive to outliers, i.e., a small amount of prediction error has a greater impact on the RMSE value. Compared with the RMSE, the MAPE, which can normalize the error of each point and reduce the impact of individual outliers on the final results, has stronger robustness. Compared with the RMSE, the MAE is characterized by employing the median instead of the average value, which reduces the impact of outliers on the results. The R^2 is the ratio of the sum of the squares of regression to the sum of squares of total deviations. The R^2 represents the proportion of the sum of the squares of total deviations that can be explained by the sum of squares of regression. These performance indices are commonly applied in regression tasks.

The parameters of these compared algorithms are given in Table 1.

The models of the LR, the RD, the BR, the ARD, the HB, the TS, the SGD, the RF, the KN, the AB, the GB, the BAG, the ET, the XGB, the KR, the SVM, and the GP are developed in Scikit-learn (version 0.22.1). The model parameters of these algorithms are set to default parameters as Scikit-learn (version 0.22.1).

Table 1
Parameters of these compared methods in two cases.

No.	Method	Hyper-parameter	Search space of hyper-parameters	Value
1	BP	HiddenLayer1	[20 30 40 50]	40
		HiddenLayer2	[20 30 40 50]	40
		HiddenLayer3	[20 30 40 50]	40
		Epochs	[50 100 150 200]	100
		Activation function	[ReLU Sigmoid]	ReLU
		Optimizer	[Adam SGD]	Adam
2	ABP	HiddenLayer1	[20 30 40 50]	30
		HiddenLayer2	[20 30 40 50]	30
		HiddenLayer3	[20 30 40 50]	30
		Epochs	[50 100 150 200]	100
		Activation function	[ReLU Sigmoid]	ReLU
		Optimizer	[Adam SGD]	Adam
		Ant amount	[40 60 80 100 120]	100
3	LSTM	HiddenLayer1	[50 100 150 200]	50
		HiddenLayer2	[50 100 150 200]	100
		HiddenLayer3	[50 100 150 200]	50
		Epochs	[20 50 100 150]	100
		Activation function	[ReLU Sigmoid]	ReLU
		Optimizer	[Adam SGD]	Adam
		Drop rate	[0.1 0.2 0.3 0.4 0.5]	0.2
4	RD	Maximum iteration	[100 500 1000 1500]	1000
5	BR	Maximum iteration	[50 100 200 300 400]	300
6	ARD	Maximum iteration	[50 100 200 300 400]	300
7	HB	Maximum iteration	[50 100 200 300 400]	100
8	TS	Maximum iteration	[50 100 200 300 400]	300
9	SGD	Maximum iteration	[100 500 1000 1500]	1000
10	RF	Number of trees	[10 50 100 150 200]	100
		Max depth of the tree	[2 4 8 15 20 25 30]	15
11	AB	Loss	[Linear Square Exponential]	Linear
12	GB	Loss	[Ls Lad Huber Quantile]	Ls
13	ET	Max depth of the tree	[2 4 8 15 20 25 30]	15
14	SVM	K-fold	[2 4 6 8 10 12]	10
		Kernel function	[Linear Gaussian]	Gaussian
15	RBF	Spread	[50 100 150 200]	100
16	TCN	HiddenLayer1	[10 20 30 40 50 60]	30
		HiddenLayer2	[10 20 30 40 50 60]	40
		HiddenLayer3	[10 20 30 40 50 60]	50
		HiddenLayer4	[10 20 30 40 50 60]	40
		HiddenLayer5	[10 20 30 40 50 60]	30
		Kernel size	[2 3]	2
		Epochs	[50 100 150 200]	100
		Activation function	[ReLU Sigmoid]	ReLU
		Optimizer	[Adam SGD]	Adam
		Drop rate	[0.1 0.2 0.3 0.4 0.5]	0.1
		Learning rate	[0.005 0.01 0.02 0.03 0.04]	0.01
17	MTCN	Temporal scales	2	2
		Spatial scales	3	3
		HiddenLayer1	[10 20 30 40 50 60]	30
		HiddenLayer2	[10 20 30 40 50 60]	40
		HiddenLayer3	[10 20 30 40 50 60]	50
		HiddenLayer4	[10 20 30 40 50 60]	40
		HiddenLayer5	[10 20 30 40 50 60]	30
		Kernel size	[2 3]	2
		Epochs	[50 100 150 200]	100
		Activation function	[ReLU Sigmoid]	ReLU
		Optimizer	[Adam SGD]	Adam
		Drop rate	[0.1 0.2 0.3 0.4 0.5]	0.1
		Learning rate	[0.005 0.01 0.02 0.03 0.04]	0.01

3.1. Data preprocessing and forecasting performance indices

(1) The data normalization applies the min–max normalization method. Each value in the dataset is normalized as

$$x_{\text{norm}} = \frac{x - \min}{\max - \min} \tag{15}$$

where x_{norm} is the normalized value.

(2) The MAPE, which is the mean of absolute values of deviations from the arithmetic mean of the real and predicted values, is calculated as

$$\text{MAPE} = \frac{\sum_{i=1}^{n} \frac{|y_i - y_i^{'}|}{|y_i|}}{n} \tag{16}$$

where n is the length of sequence; y_i is the real value; $y_i^{'}$ is the predicted value.

(3) The RMSE, which is the result of averaging the square root of the deviation of predicted value from the real value, is presented as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^{'})^2} \tag{17}$$

(4) The MAE, which is the median value of the absolute value of deviation between the predicted value and the real value, is described as

$$\text{MAE} = median(|y_i - y_i^{'}|) \tag{18}$$

(5) The R^2, which is the ratio of the sum of squared regressions to the total sum of squares in multiple regression, measures the degree of fitting in multiple regression. The R^2 is closer to 1 means the larger the proportion of the sum of squared regressions in the total sum of squares and the higher regression fitting accuracy, which is presented as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_i^{'})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{19}$$

where $\bar{y}$ can be calculated as

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{20}$$

3.2. Predict load values of next day (Case 1)

The simulation results obtained by these compared algorithms under Case 1 are given in Fig. 7. To clearly show the curves of comparison results, nine prediction results including the MTCN, the TCN, the LR, the ARD, the RF, the GB, the KR, the LSTM, and the BP are selected for visualization.

The simulation results obtained by these 9 algorithms (Fig. 7) show that: (i) 7-day historical load is applied to predict the load of 24 h in the future, all methods achieve high prediction accuracy except the BP and the LSTM models; (ii) as the input historical data dimension increases to 21 days, 99 days, and 199 days, the prediction errors of the LR, the RF, the ARD, the GB, and the KR models increase significantly; the prediction accuracy of other models fluctuate slightly; (iii) while the resulting curve predicted by the MTCN and the real load curve have the same amplitude changing, that is to say, the MTCN can obtain high accuracy with large historical data dimension.

Each method is applied to predict the power load of the next day with 7-day, 21-day, 99-day, and 199-day historical data and to obtain four sets of load prediction results. All these sets of the results are calculated by the MAPE, the RMSE, the MAE, and the R^2 forecasting performance indices. Then, the MAPE, the RMSE, the MAE and the R^2 of the four results were averaged to output the final model performance index. The simulation results obtained by these 23 algorithms in Case 1 (Fig. 8) show that: (i) the MAPE, the RMSE and the MAE obtained by the MTCN are the smallest and the R^2 is the largest; these indices imply that the load prediction accuracy of the MTCN is the highest; (ii) the MAPE of the MTCN is 0.53% lower than the second smallest TCN and 13.19% lower than the largest SGD; (iii) the RMSE of the MTCN is 34.7 lower than the second smallest TCN and 697.6 lower than the largest ET; (iv) the MAE

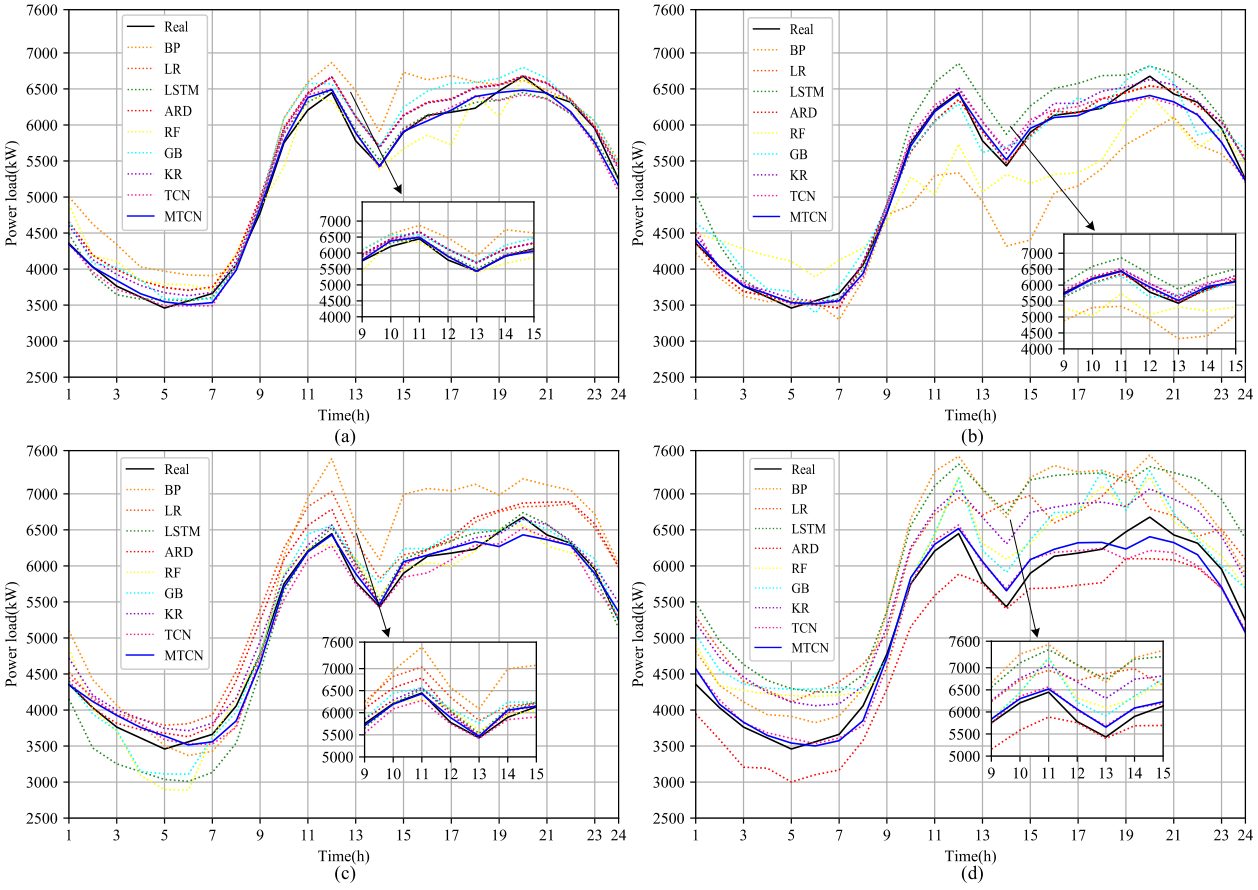

Fig. 7. Curves of obtained by compared algorithms with different dimensions history load in Case 1: (a) 7-day history load; (b) 21-day history load; (c) 99-day history load; (d) 199-day history load.

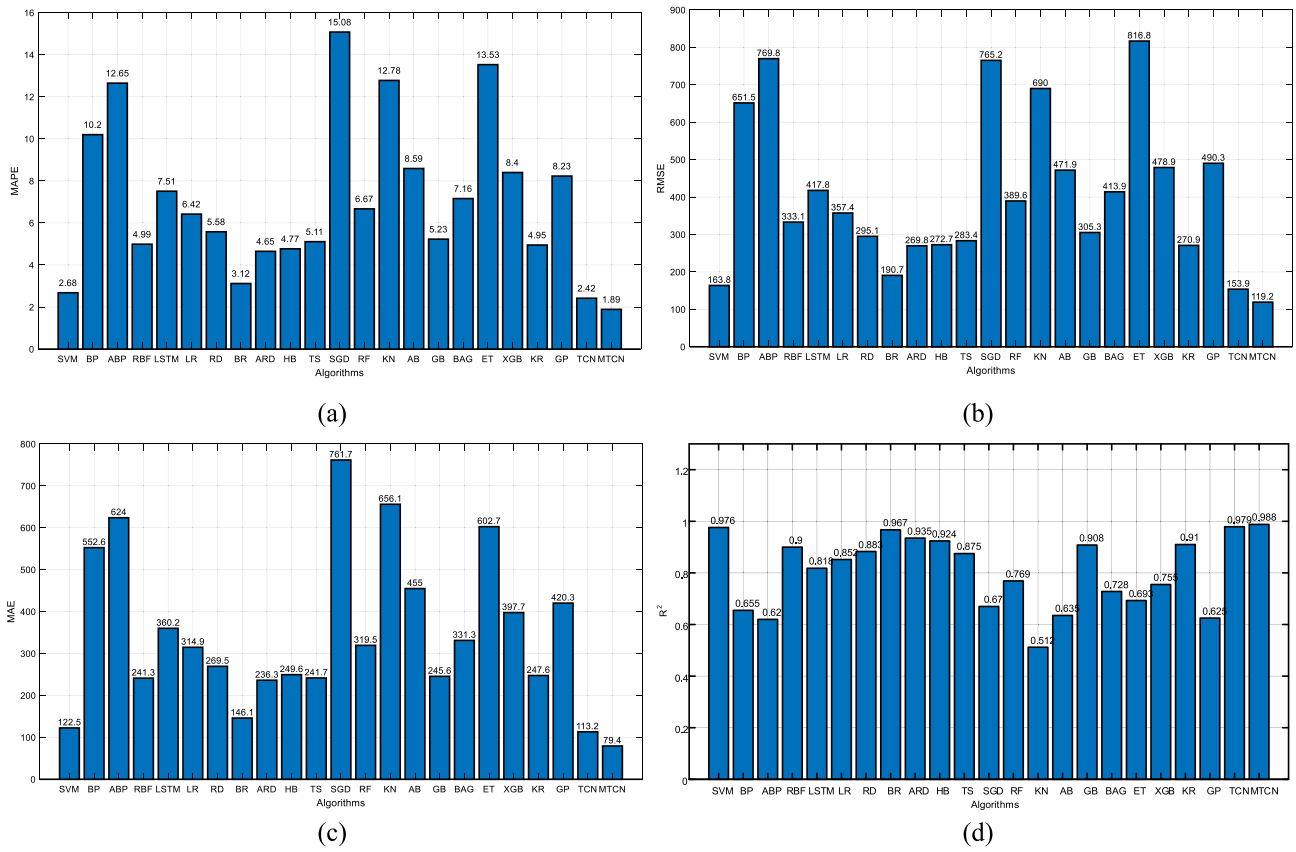

Fig. 8. Forecasting performance indices obtained by compared algorithms in Case 1: (a) mean absolute percentage error index; (b) root mean squared error index; (c) median absolute error index; (d) R-squared index.

of the MTCN is 33.8 lower than the second smallest TCN and 682.3 lower than the largest SGD; (v) the R^2 of the MTCN is 0.009 higher than the second-largest TCN and 0.476 higher than the smallest KN.

Therefore, the MTCN model can effectively apply the sequence information of the historical load to accurately predict the power load of the next day.

3.3. *Predict load value of next week (Case 2)*

The predicted results obtained by these compared algorithms under Case 2 are given in Fig. 9. Similarly, to clearly show the curves of comparison results, nine prediction results including the MTCN, the TCN, the LR, the ARD, the RF, the GB, the KR, the LSTM, and the BP are selected for visualization. The simulation results obtained by these nine algorithms (Fig. 9) show that: (i) 7-day historical load is applied to predict the load of 168 h in the next week, the results errors obtained by the BP and the LSTM models are larger than others methods; with the increase of the input historical data dimensions, the prediction accuracy of the KR, the ARD, the LR, the GB, and the RF models fluctuate greatly; (ii) the prediction errors obtained by the KR, the ARD, the LR, the GB, and the RF models are much larger than that of other models in the case of applying the 199-day historical load to predict the load; (iii) with the increase of the input data dimension from 7 dimensions to 199 dimensions, the predicted result curve of the MTCN is more consistent with the real load curve amplitude change; the MTCN achieves higher accuracy than others methods.

Similar to the experiment of predicting the load value of the next day, the experiment in this case compares the four forecasting performance indices of these 23 methods. The simulation results obtained by these 23 algorithms in Case 2 (Fig. 10) show that: (i) the MAPE, the RMSE and the MAE obtained by the MTCN are the smallest and the R^2 is the largest; the MTCN is the most accurate for predicting the load value of the next week; (ii) comparing the MAPE, the MTCN is 0.24% lower than the second smallest TCN and 7.18% lower than the largest AB; (iii) comparing the RMSE, the MTCN is 22.4 lower than the second smallest TCN and 831.2 lower than the largest AB; (iv) comparing the MAE, the MTCN is 19.3 lower than the second smallest TCN and 845.9 lower than the largest AB; (v) comparing the R^2, the MTCN is 0.008 higher than the second-largest TCN and 0.516 higher than the smallest AB.

Therefore, the results under Case 2 show that although the length of the predicted load value is increased, compared with other models, the MTCN can achieve the highest prediction accuracy.

As shown in these two cases of predicting the load value of the next day and the next week, the MTCN can predict short-term load precisely, the characteristics of the MTCN model are summarized as follows.

(1) The MTCN has strong nonlinear expression ability, which can efficiently fit the nonlinear relationship between the input data and output data. Since the MTCN can learn the characteristics of time series data and can remember long-term time information, the MTCN can accurately learn the periodic law of load data changing.
(2) Since the gradient of the MTCN is a stable gradient, the “vanishing gradient problem” and “exploding gradient problem” that often occur in the neural networks training process are mitigated in the training process of the proposed MTCN.
(3) Since the MTCN has a flexible structure and receives the input data of any length by sliding one-dimension convolution kernel, the data of different temporal scales can be trained with the same MTCN model.
(4) The MTCN adopts an MTSS structure to reduce the impact of noise errors in load data and to generate the data of varying

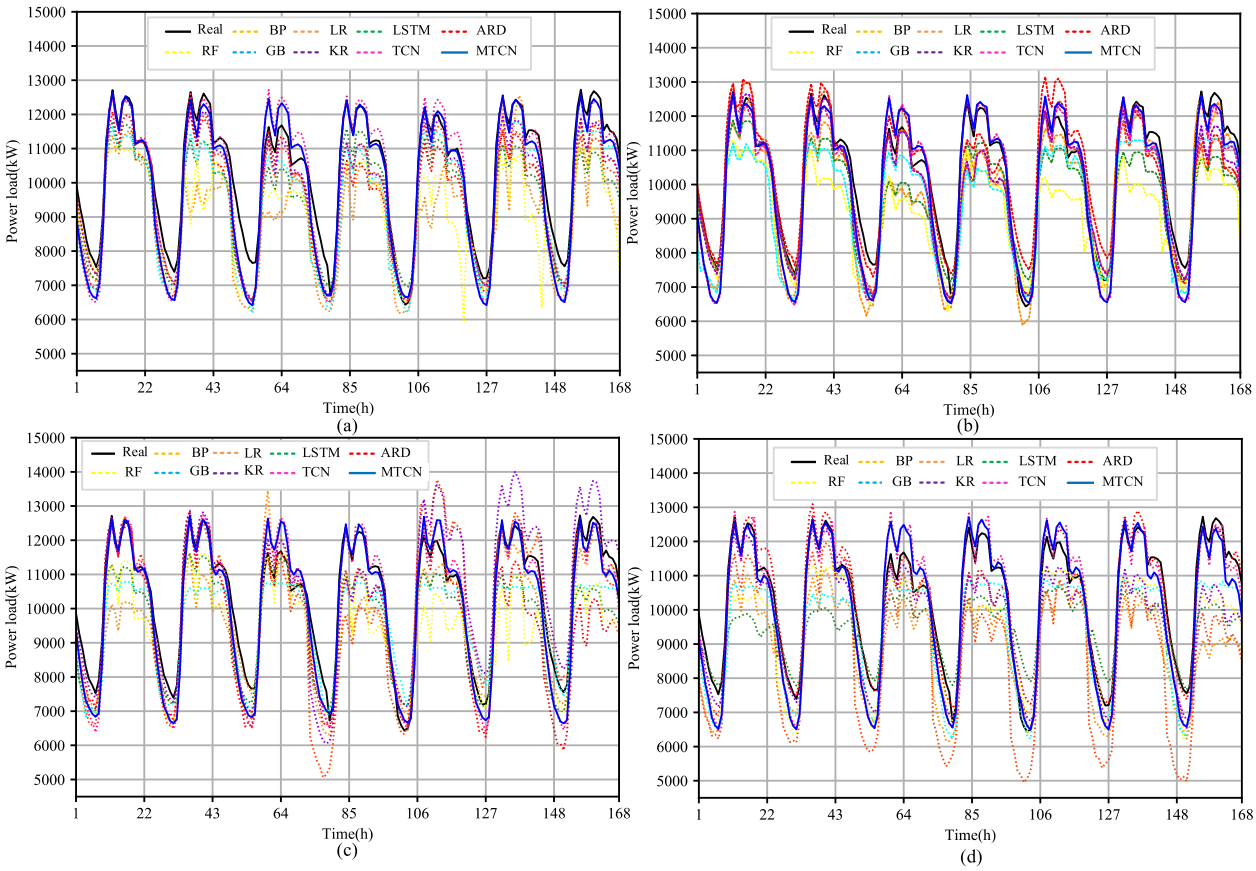

Fig. 9. Curves of obtained by compared algorithms with different dimensions history load in Case 2: (a) 7-day history load; (b) 21-day history load; (c) 99-day history load; (d) 199-day history load.

lengths for further highlighting the time series characteristics. Thus, the quality of training samples for the MTCN is higher and the training network of the MTCN can be easier trained.

The stability and robustness of the multi-temporal-spatial-scale temporal convolution structure can be guaranteed by residual connection and datasets. The residual connection can solve the problem of "vanishing gradient problem" during network training, therefore, the model can be trained steadily. The load data applied in the experiment covers a variety of load types including industrial electricity, commercial electricity and other electricity. The power amplitude range of load data of about 5 years can ensure the diversity of load level. Therefore, the model has great robustness. In addition, the standardized training strategy of the MTCN can always guarantee a fully trained forecasting network.

4. Conclusions

This paper proposes a multi-temporal-spatial-scale temporal convolutional network for the short-term load forecasting problem of power systems. The power load data from the year 2008 to 2019 about 4300 days in a city of Guangxi Zhuang Autonomous Region (China) is selected as the training samples for comparing the multi-temporal-spatial-scale temporal convolutional network and 22 compared algorithms. The load values of each hour of the next day and the next week are predicted in the experiments. The simulation results show that the results obtained by the multi-temporal-spatial-scale temporal convolutional network are the most accurate results under forecasting performance indices of mean absolute percentage error, root mean squared error, median absolute error, and R-squared. The major conclusions of this paper are listed as follows.

(1) The multi-temporal-spatial-scale temporal convolutional network improved by the multi-temporal-spatial-scale method is more accurate than the original data-driven short-term load forecasting model without the multi-temporal-spatial-scale method. Therefore, the multi-temporal-spatial-scale characteristics of load data should be considered for the short-term load forecasting problems.
(2) The multi-temporal-spatial-scale temporal convolutional network based short-term load forecasting model with 7-day, 21-day, 99-day, and 199-day historical load data has higher prediction accuracy than other compared methods. Although the meteorological factors are not considered as input feature data, the load forecasting model based on the multi-temporal-spatial-scale temporal convolutional network has the highest performances.
(3) The multi-temporal-spatial-scale temporal convolutional network can learn not only the nonlinear mapping relationship of the data but also the characteristics of the time series. The proposed method can fully learn the characteristics of the load data and can accurately predict results. Therefore, choosing a model with both feature learning ability and time series memory ability is easier to achieve high accurate load forecasting results.

In future works, the application of this method for the forecasting of new energy sources such as wind power and photovoltaic power generation could be further studied. Besides, numerous optimization algorithms could be applied to optimize the initialization weights and the thresholds of the neurons of the proposed method, which can reduce the complexity of model training and the training time. The quality of load data samples could be improved by adding high-performance sensors to reduce the complexity of prediction model training. The multi-temporal-

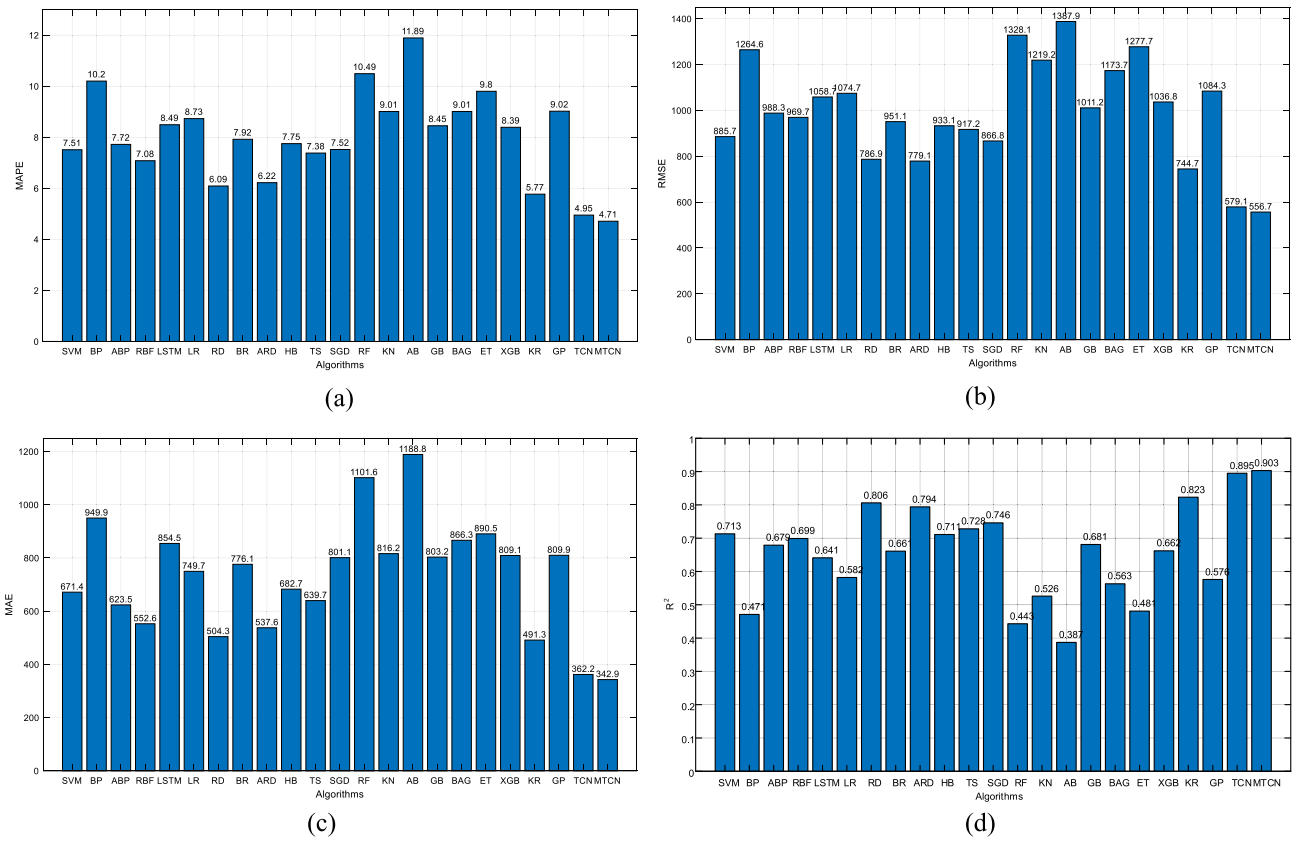

Fig. 10. Forecasting performance indices obtained by compared algorithms in Case 2: (a) mean absolute percentage error index; (b) root mean squared error index; (c) median absolute error index; (d) R-squared index.

spatial-scale method can be combined with other short-term load forecasting models for improving prediction accuracy.

CRediT authorship contribution statement

Linfei Yin: Conceptualization, Supervision, Methodology. **Jiaxing Xie:** Data curation, Software, Validation.

Declaration of Competing Interest

None.

Acknowledgments

This work was supported by the National Natural Science Foundation of Guangxi Province under Grant. AD19245001 and 2020GXNSFBA159025.

References

[1] Hafeez G, Alimgeer KS, Khan I. Electric load forecasting based on deep learning and optimized by heuristic algorithm in smart grid. Appl Energy 2020;269:114915.
[2] Liu J, Li Y. Study on environment-concerned short-term load forecasting model for wind power based on feature extraction and tree regression. J Cleaner Prod 2020; 264:121505.
[3] Debnath KB, Mourshed M. Forecasting methods in energy planning models. Renew Sustain Energy Rev 2018;88:297–325.
[4] Zendehboudi A, Baseer MA, Saidur R. Application of support vector machine models for forecasting solar and wind energy resources: A review. J Cleaner Prod 2018;199:272–85.
[5] Wang Q, Li S, Pisarenko Z. Modeling carbon emission trajectory of China, US and India. J Cleaner Prod 2020;258:120723.
[6] He F, Zhou J, Feng Z, Liu G, Yang Y. A hybrid short-term load forecasting model based on variational mode decomposition and long short-term memory networks considering relevant factors with Bayesian optimization algorithm. Appl Energy 2019;237:103–16.
[7] Rafiei M, Niknam T, Aghaei J. Probabilistic load forecasting using an improved wavelet neural network trained by generalized extreme learning machine. IEEE Trans Smart Grid 2018;9(6):6961–71.
[8] Singh P, Dwivedi P. Integration of new evolutionary approach with artificial neural network for solving short term load forecast problem. Appl Energy 2018;217: 537–49.
[9] Chen K, Chen K, Wang Q, He Z, Hu J, He J. Short-term load forecasting with deep residual networks. IEEE Trans Smart Grid 2018;10(4):3943–52.
[10] He Y, Zheng Y. Short-term power load probability density forecasting based on Yeo-Johnson transformation quantile regression and Gaussian kernel function. Energy 2018;154:143–56.
[11] Wang J, Yang W, Du P, Niu T. A novel hybrid forecasting system of wind speed based on a newly developed multi-objective sine cosine algorithm. Energy Convers Manage 2018;163:134–50.
[12] Yang Y, Li S, Li W, Qu M. Power load probability density forecasting using Gaussian process quantile regression. Appl Energy 2018;213:499–509.
[13] Wang Y, Chen Q, Sun M, Kang C, Xia Q. An ensemble forecasting method for the aggregated load with subprofiles. IEEE Trans Smart Grid 2018;9(4):3906–8.
[14] Raza MQ, Mithulananthan N, Li J. Multivariate ensemble forecast framework for demand prediction of anomalous days. IEEE Trans Sustain Energy 2018;11(1): 27–36.
[15] Fan GF, Peng LL, Hong WC. Short term load forecasting based on phase space reconstruction algorithm and bi-square kernel regression model. Appl Energy 2018;224:13–33.
[16] Tian C, Hao Y, Hu J. A novel wind speed forecasting system based on hybrid data preprocessing and multi-objective optimization. Appl Energy 2018;231:301–19.
[17] Lin P, Peng Z, Lai Y, Cheng S, Cheng Z, Wu L. Short-term power prediction for photovoltaic power plants using a hybrid improved Kmeans-GRA-Elman model based on multivariate meteorological factors and historical power datasets. Energy Convers Manage 2018;177:704–17.
[18] Wang H, Lei Z, Liu Y, Peng J, Liu J. Echo state network based ensemble approach for wind power forecasting. Energy Convers Manage 2019;201:112188.
[19] Chitalia G, Pipattanasomporn M, Garg V, Rahman S. Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks[J]. Appl Energy 2020;278:115410.
[20] Wang K, Qi X, Liu H. A comparison of day-ahead photovoltaic power forecasting models based on deep learning neural network. Appl Energy 2019;251:113315.
[21] Lian C, Liu M, Zhang J, Shen D. Hierarchical fully convolutional network for joint atrophy localization and Alzheimer's Disease diagnosis using structural MRI. IEEE Trans Pattern Anal Mach Intell 2020;42(4):880–93.
[22] Lore KG, Stoecklein D, Davies M, Sarkar S. A deep learning framework for causal shape transformation. Neural Networks 2018;98:305–17.

[23] Zhang Z, Wang X, Jung C. DCSR: Dilated convolutions for single image super-resolution. IEEE Trans Image Process 2018;28(4):1625–35.
[24] Xu B, Ye H, Zheng Y, Wang H, Lu T, Jiang Y. Dense dilated network for video action recognition. IEEE Trans Image Process 2019;28(10):4941–53.
[25] Wang G, Giannakis GB, Chen J. Learning ReLU networks on linearly separable data: Algorithm, optimality, and generalization. IEEE Trans Signal Process 2019;67(9):2357–70.
[26] Wu S, Li G, Deng L, Liu L, Dong W, Xie Y, et al. *L*1-Norm batch normalization for efficient training of deep neural networks. IEEE Trans Neural Networks Learn Syst 2018;30(7):2043–51.
[27] Achille A, Soatto S. Information dropout: Learning optimal representations through noisy computation. IEEE Trans Pattern Anal Mach Intell 2018;40(12):2897–905.
[28] Stevens Eli, Antiga Luca, Viehmann Thomas. Deep Learning with PyTorch[M]. Manning Publications Co.; 2020.