Hindawi
Journal of Sensors
Volume 2022, Article ID 6730677, 14 pages
https://doi.org/10.1155/2022/6730677

Research Article

A Load Forecasting Algorithm for Power Internet of Things Using Training Data Dimension Expansion and Ensemble Learning Technique

Chang Liu, Wei Rao, Jin Wang, Zeyang Tang, Chang Liu, Jie Wang, Li Tian, Liang Zhou, and Jiangpei Xu

State Grid Hubei Electric Power Research Institute, Wuhan 430000, China

Correspondence should be addressed to Chang Liu; 12612209@qq.com

Received 28 April 2022; Accepted 3 June 2022; Published 20 June 2022

Academic Editor: Han Wang

With the development of the power internet of things (IOT), load forecasting will play an important role the power system. It can optimize the power generation planning and improve the economical operation of power IOT. In this paper, a new loading forecasting algorithm for power IOT is proposed using training data dimension expansion and ensemble learning. In the offline phase, the obtained meteorological data and time information is normalized to remove the unit effect at first. Then, the Hampel filter is used to cope with the outliers of the meteorological data from sensors. Through the preprocessing, the fingerprint of the training data is constructed. Next, the matrix multiplication method is proposed to expand the dimension of training data fingerprint information. Finally, the ensemble learning combining multiple long short-term memory (LSTM) networks are proposed to obtain multiple power load forecasting models and corresponding weight coefficients. In the online phase, the obtained meteorological data and time information are preprocessed to form the input of and power load forecasting models. The final power load forecasting is obtained by linear weighted sum method with intermediate forecasting result. In the proposed algorithm, more features of training data can be obtained by the data dimension expansion. Moreover, the ensemble learning using LSTM can make fully use of the timing sequence of training and improve the generalization performance of offline training. Experiment results illustrate that the proposed algorithm has better forecasting performance than existing methods.

1. Introduction

With the increasing requirement of energy consumption, a clean, low-carbon, safe, and shared power system has received many attentions for both academic and industry [1, 2]. The new energy revolution called "energy + Internet" becomes a research hotspot [3]. In China, the State Grid Corporation began to develop the ubiquitous power internet of things (IOT) for smart grid in 2019 [3]. The power IOT connects people and devices that are related to the power grid. By collecting device-related data using sensor, the data is sent to the server which is processed by big data analysis, cloud computing, and artificial intelligence technologies. It effectively integrates communication infrastructure resources and power system infrastructure resources which can realize the interconnection of all things in the power system. Now, the power IOT can promote the efficiency of energy services and new energy consumption dramatically. In [4], an architecture for ubiquitous power IOT is proposed which brings great convenience to the collection, transmission, processing, and storage of data. In [5], the characteristics of power IOT, its application, and the needs of smart grid for communication coverage and data acquisition are proposed.

For the requirement of power IOT construction, power load forecasting plays an important role in the operation planning of power system. The authors of [6] proposed a load forecasting method by two groups of features. The half-hour electrical load variables obtained from the smart

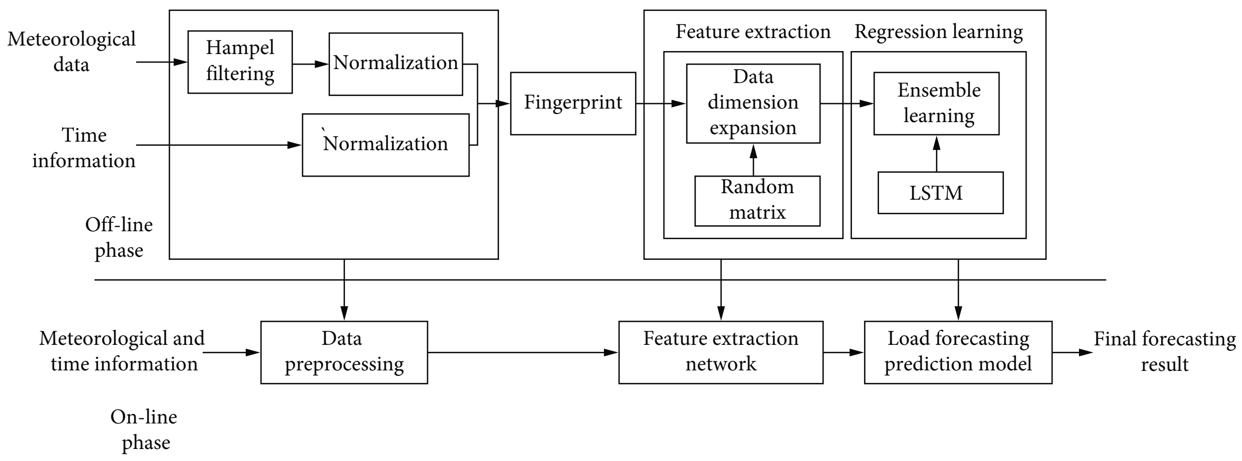


FIGURE 1: The block diagram of the proposed algorithm.

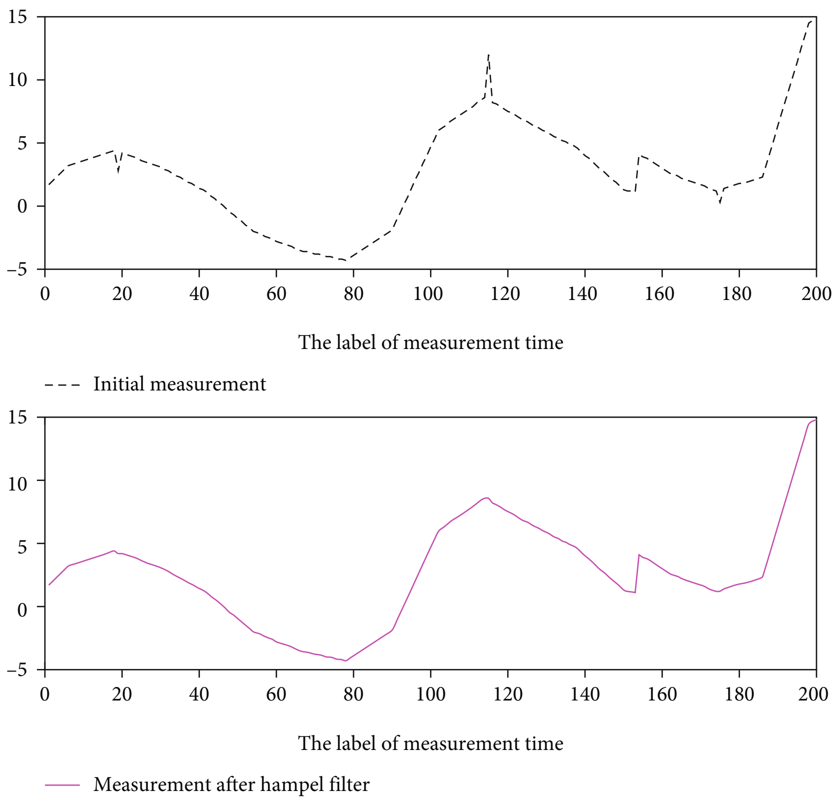


FIGURE 2: The description of temperature information process by Hampel filter.

meters through the IOT technology together with the lagged load and calendar variables are used for short-term load forecasting by multilayer perceptron neural networks. In [7], an ultrashort term load forecasting method for the power IOT is proposed by federated learning. It learns the model parameters from the data distributed in multiple edge nodes. Since rapid and accurate power load forecasting can optimize the power generation planning and improve the economical operation of power IOT, it is urgent to study the load forecasting technique for power IOT [8].

Recently, the load forecasting is performed with different kinds of techniques in the literature which can be broadly divided into two groups: traditional and artificial intelligence-based techniques.

For traditional technique, the statistical method is mostly used for load forecasting. In [9], considering the load forecast is a conditional expectation of load given the time, weather conditions, and other explanatory variables; load forecasting can be calculated directly from given parameters. In [10], the general exponential smoothing is used to develop an adaptive load forecasting system using observed values of integrated hourly demand. In [11], a modified autoregressive moving average- (ARMA-) based method with non-Gaussian process is used to improve the

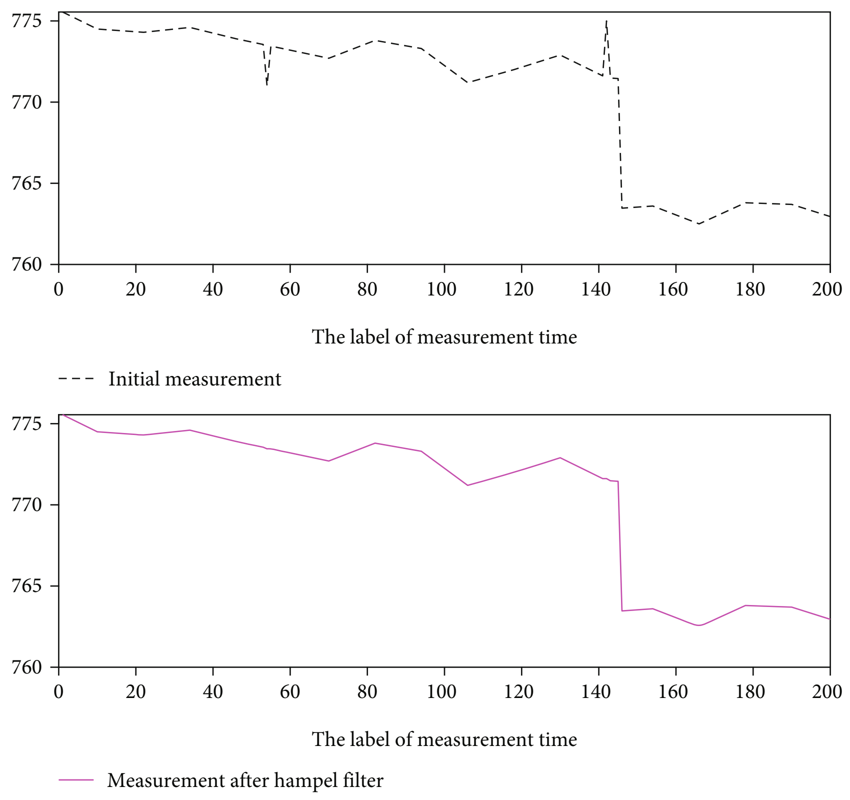


Figure 3: The description of pressure information process by Hampel filter.

forecasting performance. Note that due to characteristics of nonlinear features of time series univariate load data, this kind of technique does not always perform well.

In order to cope with the limits of traditional technique, artificial intelligence-based technique is developed. First, some machine learning-based methods are proposed. In [12], a kernel-based support vector regression (SVR) method is used for load forecasting by choosing kernel function. In [13], the artificial neural network (ANN) is used to obtain the load forecasting model for weekdays and weekends, respectively. Now, with the development of machine learning theory, some deep learning methods are used for load forecasting, because it has greater number of hidden layers to deal with the complicated nonlinear patterns. In [14], a multiscale convolutional neural network with time cognition is proposed for load forecasting. Since the recurrent neural network (RNN) is effective to capture non-stationary training data, a pooling-based deep RNN in [15] is proposed to solve the overfitting problem in load forecasting. Note that vanishing gradient and exploding gradient problem arise in RNN can reduce the prediction accuracy. Thus, the long short-term memory (LSTM) network has been taken into account to solve these problems. In [16], an effective methodology using the LSTM network is developed to make a precise forecasting under more complex time series load data condition.

According to the above discussion, we continue to study deep learning-based load forecasting algorithm. Since the time and meteorological data, such as wind speed, temperature, and pressure, are critical affect the power consumption [17], in this paper, a deep learning technique is used to training the relationship between the above factors and power load. The contributions of this paper can be summarized as follows:

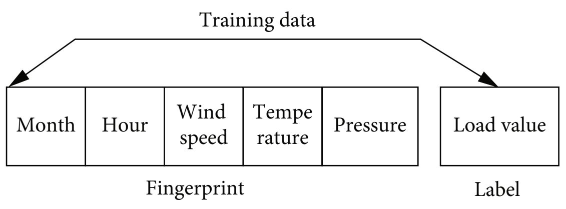


Figure 4: The description of the training data.

(1) A deep learning-based load forecasting algorithm is proposed by LSTM. It can effectively avoid the insensitivity of training data time series in the prediction process

(2) For data preprocessing, the Hampel filter can eliminate the outlier caused by sensor noise. Moreover, the feature extraction network using random matrix multiplication can obtain better feature representation which can improve the offline training performance

(3) For regression learning, the ensemble learning framework which combines multiple LSTM networks is used to train the relationship between the

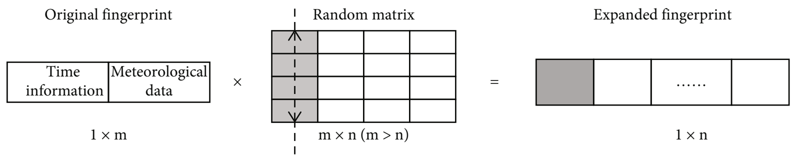


Figure 5: The schematic diagram of the proposed dimension expansion.

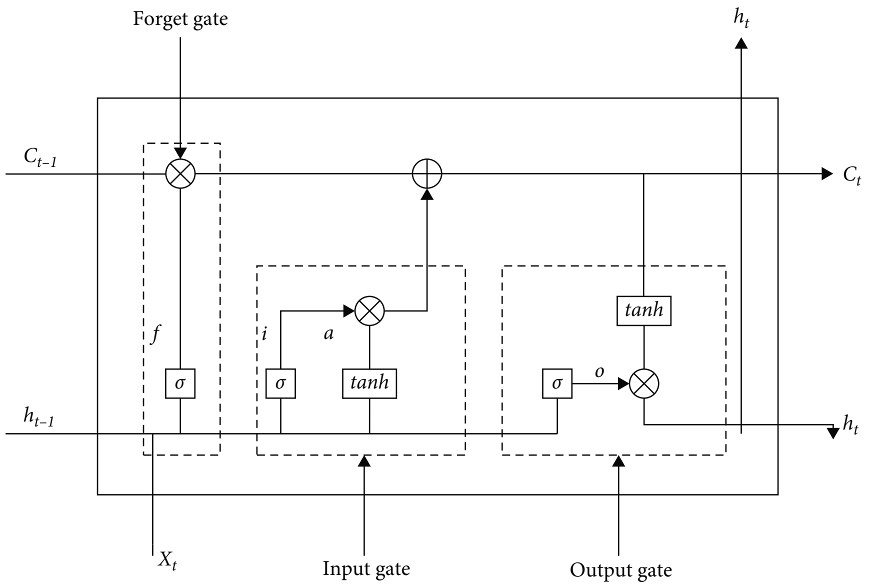


Figure 6: The description of LSTM architecture.

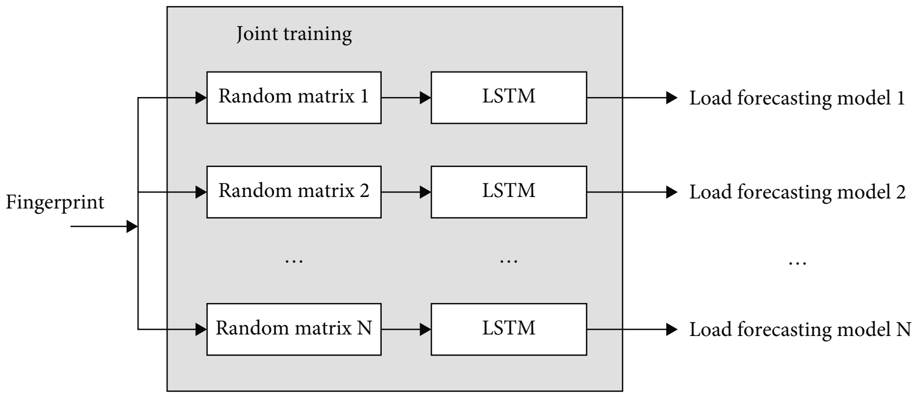


Figure 7: Flowchart of the proposed regression learning.

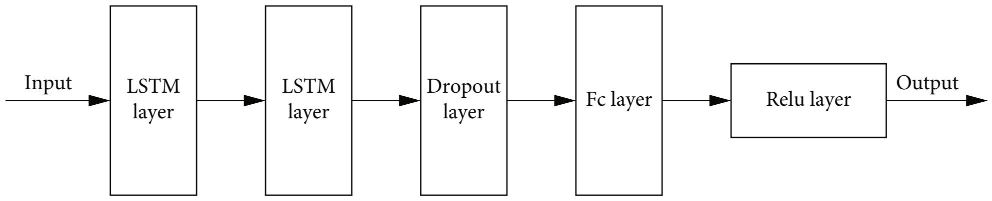


Figure 8: The description of LSTM network.

measurements and load. It can reduce the risk of falling into local optimal solutions and improve the generalization performance

(4) Feature extraction network (random matrix) and regression learning network (LSTM network) are combined for joint learning and optimization. Thus,

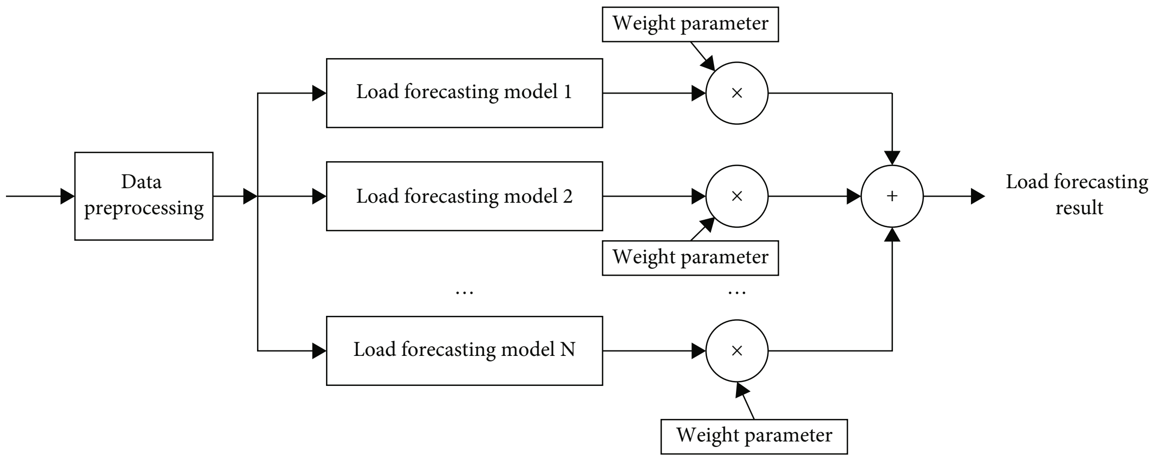


Figure 9: The block diagram of the online phase description.

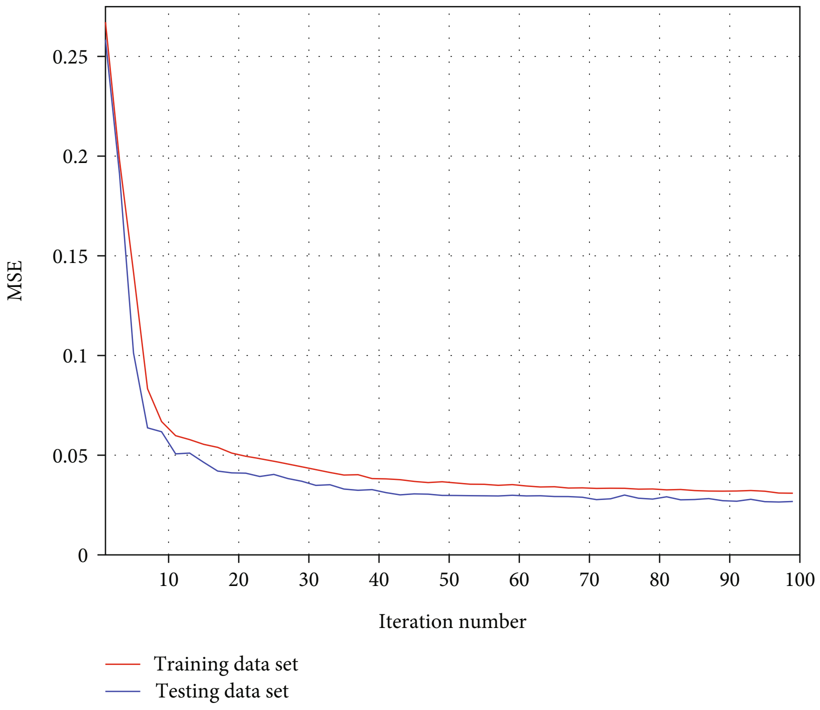


Figure 10: The description of the training performance of the proposed algorithm.

the optimal parameters in the global sense can be obtained, and the final load forecasting performance can be improved

The remainder of this paper is organized as follows. Section 2 gives the algorithm framework. The offline phase description and the online phase description of the proposed algorithm are explained in Sections 3 and 4, respectively. Field test and performance analysis are illustrated in Section 5, and conclusions are drawn in Section 6.

2. Algorithm Framework

According the block diagram shown in Figure 1, the proposed algorithm contains two main phases: offline training phase and online forecasting phase. For offline training phase, it contains three important steps: (1) training data preprocessing, (2) feature extraction using dimension expansion, and (3) regression learning using ensemble learning. For another, in the online forecasting phase, after the similar preprocessing of the obtained data, the trained feature extraction network and forecasting model are used to estimate the final load forecasting result. In the following, each step of the proposed algorithm will be described in detail.

3. Offline Phase Description

3.1. Training Data Preprocessing. In this step, data normalization and outlier detection are used for data preprocessing. First, as we know, some measurement environments and the

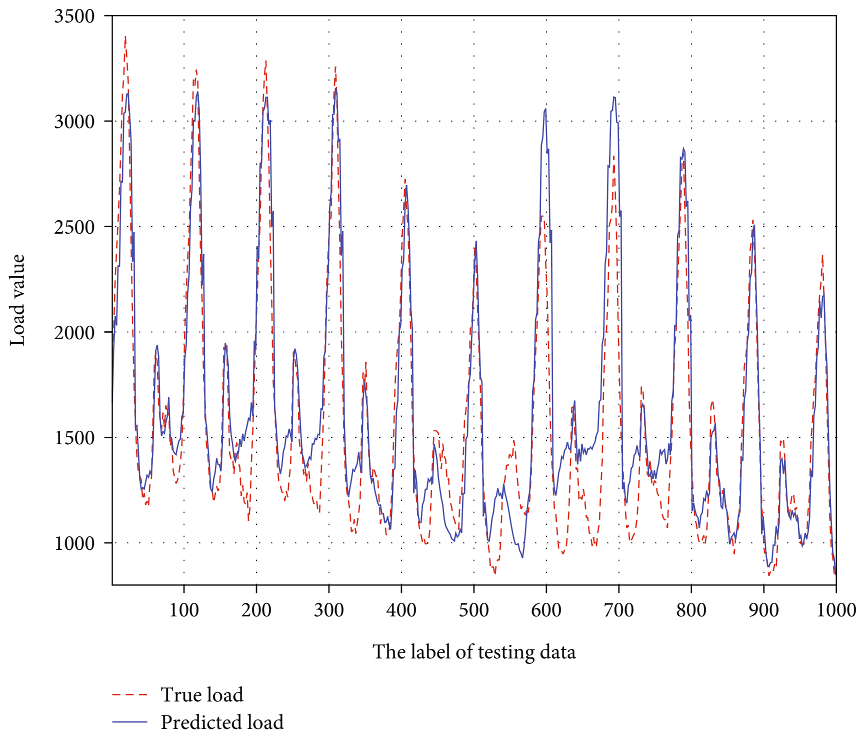


FIGURE 11: The load forecasting description of the proposed algorithm.

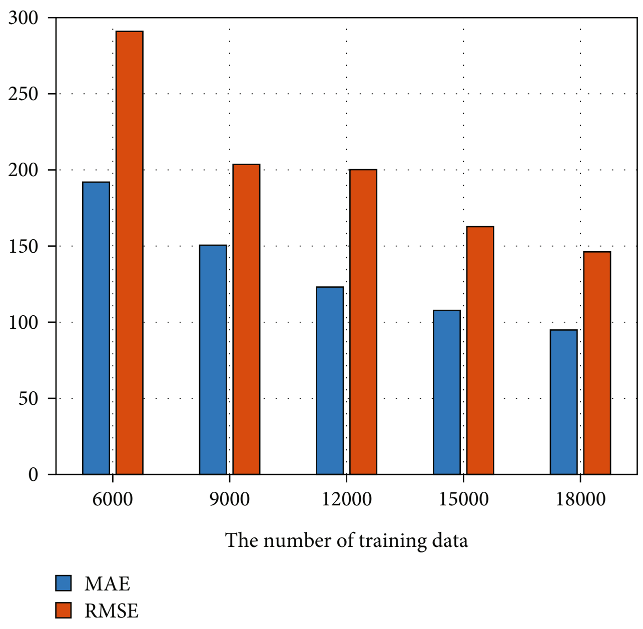


FIGURE 12: MAE and RMSE description of the proposed algorithm.

sensor noise may lead some measurements become outlier in the data collection. In the paper, the Hampel filter is used to remove these outliers [18]. For a given vector $x = \{x_1, x_2, \cdots, x_n\}$, the observation window of each measurement is defined at first. Assume the half-length of the window is k; the total length of the defined window is calculated as $2k + 1$ (including the given measurement). Then, the median of all measurement data in the window is calculated as

$$\overline{x_i} = \text{median}(x_{i-k}, x_{i-k+1}, \cdots, x_i, \cdots, x_{i+k-1}, x_{i+k}). \tag{1}$$

The standard deviation of each measurement for the absolute value of the median is written as

$$e_i = 1.4286\text{median}(|x_{i-k} - \bar{x}_i|, \cdots, |x_{i+k} - \bar{x}_i|), \tag{2}$$

After filter, the measurement can be described as

$$b_i = \begin{cases} x_i(|x_i - \bar{x}_i| \leq 3\, e_i, \\ \bar{x}_i(|x_i - \bar{x}_i| \geq 3\, e_i. \end{cases} \tag{3}$$

If the measurements exceed three times of the obtained standard deviation, the median is used to substitute the measurement data.

Taking the temperature and pressure information as an example, Figures 2 and 3 describe the measurement data preprocessing results. It can be seen that after Hampel filter process, some outliers are removed. Thus, Hampel filter has effects on outlier deletion.

Next, the collected meteorological data and time information are scaled in order to lead them fall into a special range. In this paper, the min-max method is used to make the obtained data information into [0 1] range which can be given by [19]

$$x'_n = \frac{x_n - x_{\min}}{x_{\max} - x_{\min}}, \tag{4}$$

where x_n, x'_n describe the original and normalized data, respectively, and $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of this kind of given data.

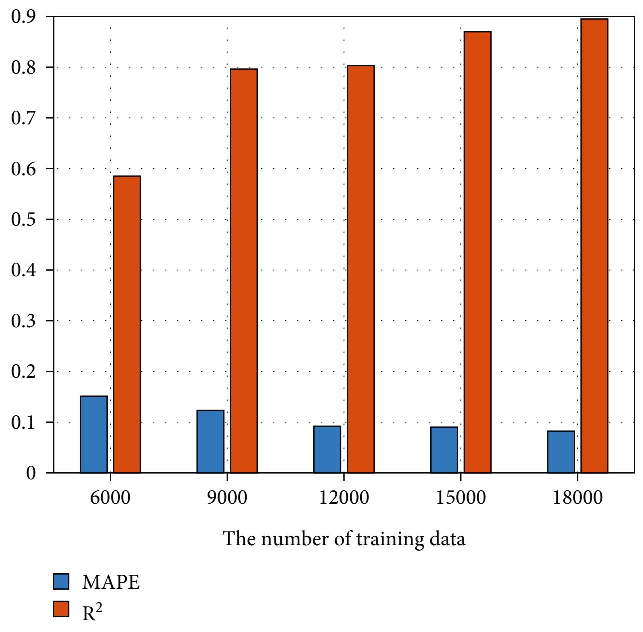


Figure 13: MAPE and R^2 description of the proposed algorithm.

Table 1: Statistical error analysis of the proposed algorithm with different numbers of LSTM.

	2	4
MAE	94.8697	86.64721
RMSE	146.1882	130.1099
MAPE	0.08238	0.07374
R^2	0.8949	0.91678

The purpose of data normalization is to remove the unit effects for the following training. Through this process, different kinds of data are at the same level. And the comprehensive evaluation and analysis can be carried out.

Thus, through the above preprocessing, the fingerprints of training data are described in Figure 4, where the fingerprint contains time information (month, hour) and meteorological data (wind speed, temperature, and pressure). The size of fingerprint is five dimensions. The label of training data is actual load.

3.2. Feature Extraction Using Training Data Dimension Expansion. After the data preprocessing, in the next, the training data expansion is used to obtain more features for training. Traditional data dimension expansion for deep learning contains random augmenter and lower-bound cropper [20]. The idea of random augmenter is to randomly generate a binary mask that can be multiplied by the data vector. The lower-bound cropper augmentation technique leverages this observation to increase the training data size. In particular, for a given collected data vector, any entry whose value is less than a certain threshold is a candidate to be removed (set to zero). All combinations of these entries can then be added to form the new training data. As shown in Figure 5, in this step, the random matrix is proposed to expand the dimension of the fingerprint.

Assume the original fingerprint is defined as A with size $1 \times m$; in order to expand the dimension of a vector, a random matrix B is defined for multiplication operation. In order to meet the requirement of calculation rule and dimension expansion, the size should be $m \times n$ (where $n > m$). The optimal element can be obtained from the training with following regression learning.

At last, the expanded fingerprint can be written as

$$C = A * B, \tag{5}$$

where the size of C is n. Each element is obtained by multiplying vector with matrix.

Note that since the size of expanded fingerprint is larger than that of the original fingerprint and the element can be determined with the offline training, more detailed feature for training data can be obtained through above dimension expansion.

3.3. Regression Learning Using Ensemble Learning. In this section, in order to improve the training performance of the offline phase, ensemble learning framework is proposed for regression learning [21]. Moreover, the LSTM network is chosen as the sublearner which can make fully use of the time series of the training data.

First, some basic knowledge of LSTM network is described [22]. According to the LSTM architecture shown in Figure 6, it contains four basic components: cell, input gate, output gate, and forget gate. The function of each component is described as follows. Cell is the core of the compute node. The information can be transferred over random time intervals through the cell. The gate traces the flow of the input and output data from the cell. The input gate and output gate are used to control the input and output. The forget gate is used to control the retention degree of historical information. Moreover, the sigmoid activation function is introduced to make the output between [0, 1]. When the output is 0, it means that all the information in the previous state is discarded. When the input gate is 1, all information in the previous state is retained.

A unit of LSTM is defined as

$$\begin{aligned} i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + b_i\right), \\ f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right), \\ o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + b_o\right), \end{aligned} \tag{6}$$

Input is defined as

$$g_t = \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{7}$$

Cell update is defined as

$$\begin{aligned} c_t &= f_t \circ c_{t-1} + i_t \circ g_t, \\ h_t &= o_t \circ \tanh\left(c_t\right). \end{aligned} \tag{8}$$

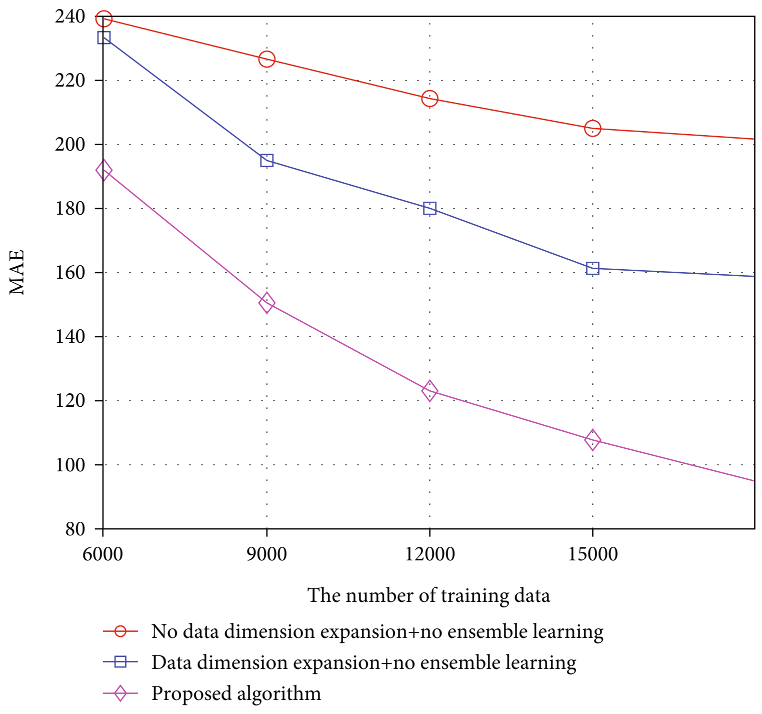


Figure 14: MAE comparison of different algorithms.

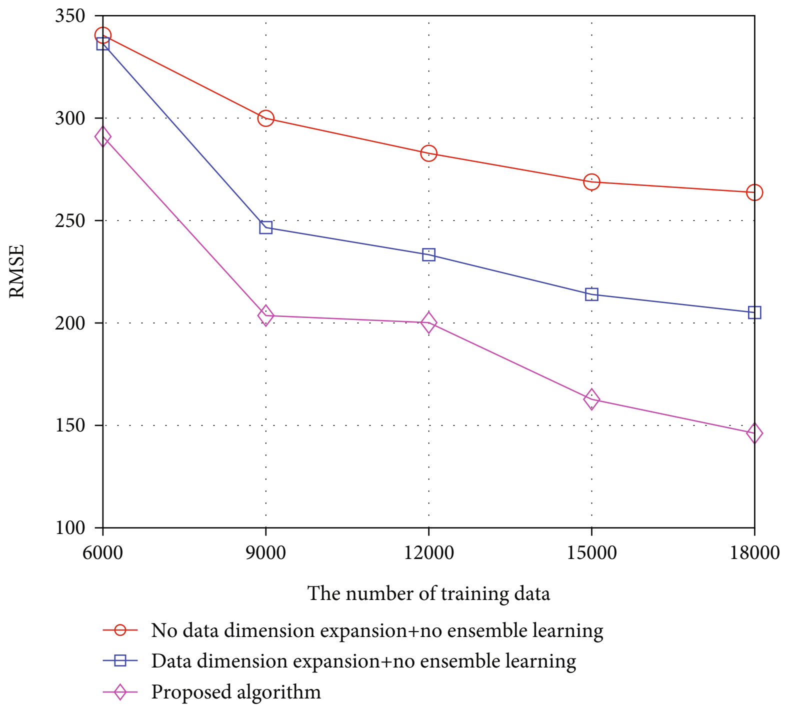


Figure 15: RMSE comparison of different algorithms.

where σ is sigmoid activation function; $\circ$ is the Hadamard product; x_t and h_t are described as the input and output of the current node; W_{xi}, W_{hi}, W_{xf}, and W_{hf} represent weight matrix; b_i, b_f, b_o, and b_c are the biased values; i_t, f_t, and o_t describe the results of input gate, output gate, and forget gate, respectively; g_t is the update condition; and x_t is the data input.

Second, the regression learning using ensemble learning framework is proposed. As we all know, an ensemble with a

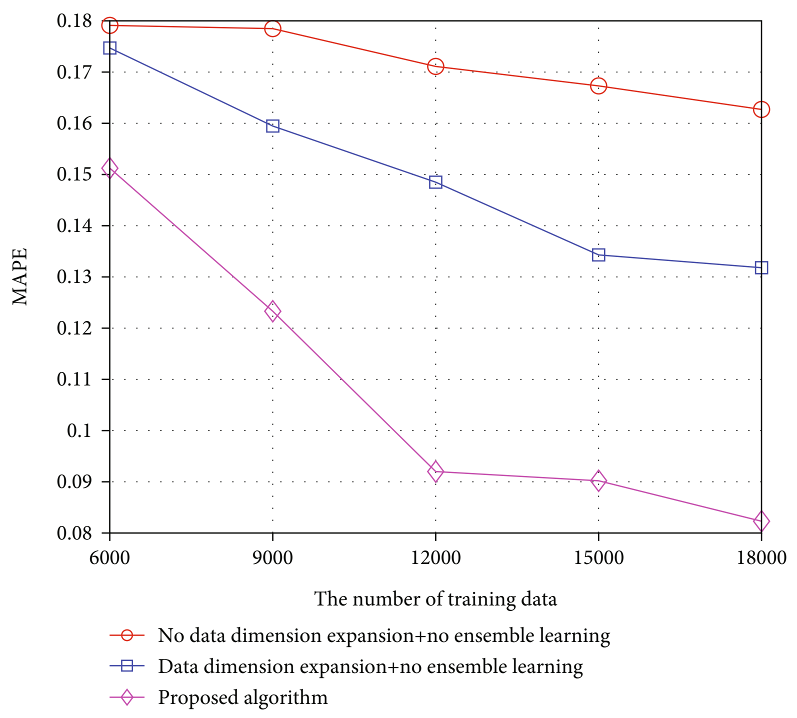


Figure 16: MAPE comparison of different algorithms.

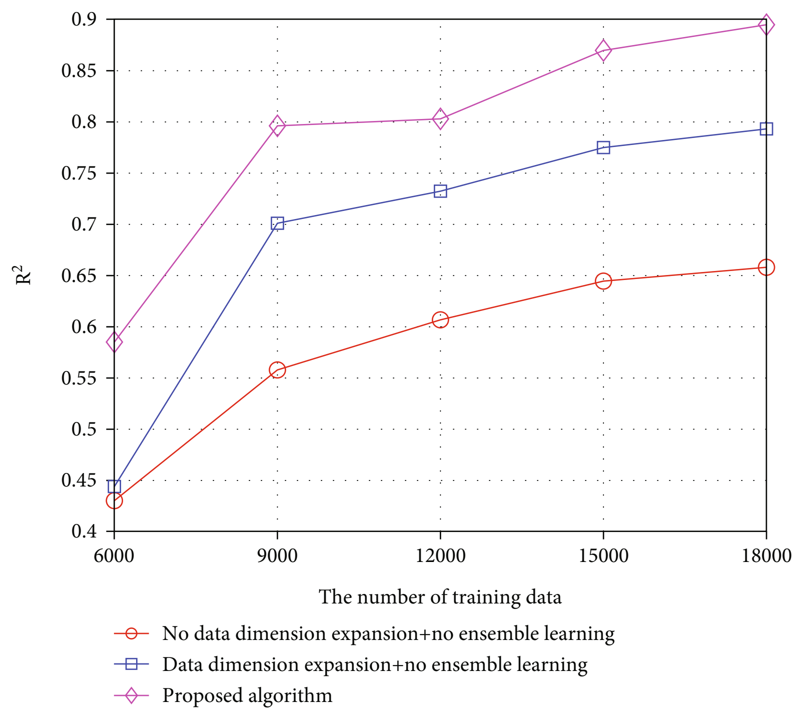


Figure 17: R^2 comparison of different algorithms.

number of sublearners has much stronger generalization ability than that of sublearners. Sublearners are usually generated from training data by a basic learning algorithm such as neural network, SVM, or other machine learning approaches. The diversity of each sublearner is vital to training performance of ensemble learning. Therefore, most ensemble methods use some effective strategies to improve the diversity of sublearners [23, 24]. Three main kinds of ensemble learning contain Adaboost, bagging, and boosting. As shown in Figure 7, by the data normalization and Hampel filter preprocessing in Section 3.1, the obtained fingerprint is used for data dimension expansion and regression learning. For each sublearner, the random matrix is generated at first and then the LSTM is used for regression learning. The task of learning is to obtain the optimal element of random matrix and the parameters of LSTM network. At last, the feature extraction network with optimal element-based random matrix and the load forecasting model is given. In the offline phase, more sublearners are used for regression learning.

Figure 8 shows the proposed framework of LSTM network. It can be seen that it contains two LSTM layers, one dropout layer, one full connection (FC) layer, and RELU layer. Both function and parameters of each layer are described as follows:

(1) LSTM layer: for the first layer, the length of the state vector unit is 8, the time stride is 1, and the activation function is tanh. For the second LSTM layer, the length of the state vector unit is 16 and the time stride is 1

(2) Dropout layer: the aim of this layer is to reduce the number of actual training parameters by disconnecting neural network randomly. In this paper, the parameter is chosen as 0.1

(3) FC layer: in this layer, it can map the feature from one space to another. The activation function is chosen as RELU

(4) RELU layer: the aim of this layer is to mitigate the effect of overfitting

At last, the weight parameter is chosen for online forecasting data fusion. In this paper, the weight parameter is determined with the offline training error. If the offline training error of the sublearner is small, the corresponding weight parameter is larger. Otherwise, the weight parameter is small. For ith sublearner, the weight parameter can be described as

$$w_i = \frac{1/e_i}{\sum_i 1/e_i}, \tag{9}$$

where e_i is the training error of the ith sublearner.

Note that in the proposed offline training, the random matrix is chosen as the feature extraction network. The feature extraction network and regression learning network are combined with each other for offline training. Thus, the performance of offline phase can be improved.

4. Online Phase Description

In this section, when the meteorological data and time information are obtained, the load forecasting model is used for load forecasting. According to the block diagram shown in Figure 9, the load forecasting process is described as follows:

First, based on the obtained time information and meteorological data, the data preprocessing proposed in offline phase is used to construct the fingerprint by normalization and Hampel filtering. Then, the fingerprint is chosen as the input data for each load forecasting model. At last, the final load is estimated by the linear weighting method which is given by

Figure 18: The figure of raspberry pi.

$$\hat{g} = \sum_i w_i g_i, \tag{10}$$

where g_i is the intermediate load forecasting of the ith load forecasting model.

5. Experiment Results and Performance Analysis

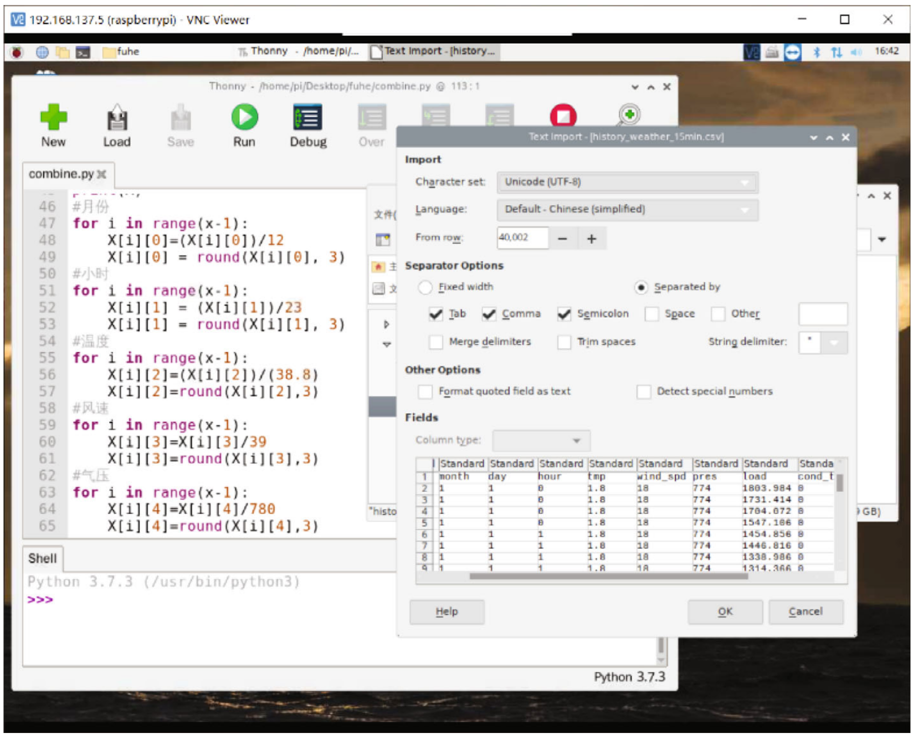
5.1. Experiment Parameter and Environment. In the experiment, the training data are chosen for a community of Suzhou, Jiangsu Province. All the measurement data are collected with the time interval of 15 min in 24 hours. The procedures are based on Windows 10 operating system Python 3.7. The pandas, numpy, and matplotlib library routines in Anaconda and Keras training framework are chosen for offline training.

5.2. Performance Evaluation Index. In this paper, the mean absolute error (MAE), mean absolute percentage error (MAPE), root mean square error (RMSE), and determinate coefficient (R^2) are chosen for performance evaluation which are shown in (11)–(14). The above performance evaluation indexes describe the forecasting performance from different aspects. The MAE is defined as the average absolute error between the predicted value and the observed value. The MAPE is the percentage value description of the MAE. The RMSE represents the standard deviation of the difference between the predicted value and the observed value. The determination coefficient is used to evaluate the fitting degree of regression model coefficients. The higher the value is, the better the model is.

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N} |\hat{q}_n - q_n|, \tag{11}$$

$$\mathrm{MAPE} = \frac{100\%}{N}\sum_{n=1}^{N} \left|\frac{\hat{q}_n - q_n}{q_n}\right|, \tag{12}$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{n=1}^{N} (\hat{q}_n - q_n)^2}{N}}, \tag{13}$$


192.168.137.5 (raspberrypi) - VNC Viewer
fuhe
Thonny - /home/pi/...
Text Import - [history...
16:42
Thonny - /home/pi/Desktop/fuhe/combine.py @ 113 : 1
New
Load
Save
Run
Debug
Over
combine.py
Shell
Python 3.7.3

```
#月份
for i in range(x-1):
    X[i][0]=(X[i][0])/12
    X[i][0] = round(X[i][0], 3)
#小时
for i in range(x-1):
    X[i][1] = (X[i][1])/23
    X[i][1] = round(X[i][1], 3)
#温度
for i in range(x-1):
    X[i][2]=(X[i][2])/(38.8)
    X[i][2]=round(X[i][2],3)
#风速
for i in range(x-1):
    X[i][3]=X[i][3]/39
    X[i][3]=round(X[i][3],3)
#气压
for i in range(x-1):
    X[i][4]=X[i][4]/780
    X[i][4]=round(X[i][4],3)
```

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

Text Import - [history_weather_15min.csv]

Import
Character set: Unicode (UTF-8)
Language: Default - Chinese (simplified)
From row: 40,002

Separator Options
Fixed width [✓ Separated by]
[✓ Tab] [✓ Comma] [✓ Semicolon] Space Other
Merge delimiters Trim spaces String delimiter: "

Other Options
Format quoted field as text Detect special numbers

Fields
Column type:

	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standa
1	month	day	hour	tmp	wind_spd	pres	load	cond_t
2	1	1	0	1.8	18	774	1803.984	0
3	1	1	0	1.8	18	774	1731.414	0
4	1	1	0	1.8	18	774	1704.072	0
5	1	1	0	1.8	18	774	1547.106	0
6	1	1	1	1.8	18	774	1454.856	0
7	1	1	1	1.8	18	774	1446.816	0
8	1	1	1	1.8	18	774	1338.986	0
9	1	1	1	1.8	18	774	1314.366	0

Help OK Cancel

Figure 19: The description of testing data import.

$$R^2 = \frac{\sum_{n=1}^{N}(\hat{q}_n - \bar{q})^2}{\sum_{n=1}^{N}(q_n - \bar{q})^2}, \tag{14}$$

where q_n and $\hat{q}_n$ describe the true and forecasting of the load, $\bar{q}$ is the mean of the load forecasting, and N is the number of load to be forecasting.

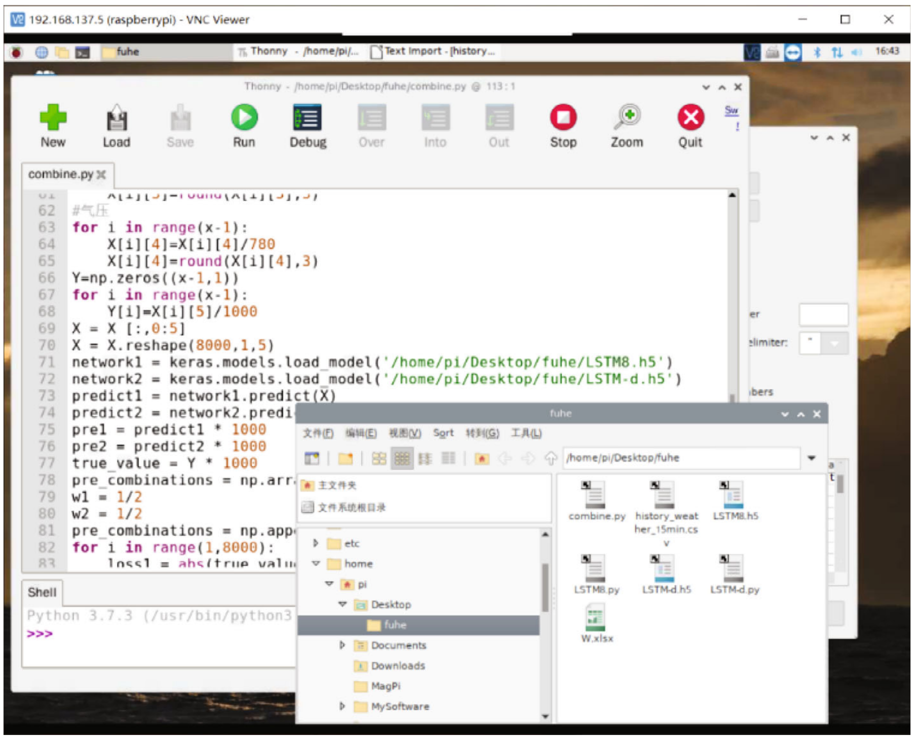
5.3. *Performance Description of the Proposed Algorithm*

5.3.1. The Description of Training Performance. In this experiment, the parameter of the running computer is given as follows: CPU—AMD R7-5800H, GPU—Nvidia RTX 3050 4G, Memery—16G, and software platform—Pycharm (Python 3.9.7)+TensorFlow 2.1.0+Keras 2.3.1. There are 18000 measurements for the experiment where the ratio between training data and testing data is 0.8 : 0.2. Figure 10 shows the training performance of the proposed algorithm, where the parameter of data dimension expansion is 16. We can find that when the number of iterations increases, the MSE of load forecasting will gradually decrease. Taking the iteration number is 80 as an example, the MSE approaches the minimum value of 0.032 which means the loss function converges. Therefore, under this iterative condition, the load forecasting model obtained from offline learning can be used for online prediction estimation.

5.3.2. Performance Description of the Proposed Algorithm. In this section, the forecasting performances of the proposed algorithm with different parameters are described. Taking the number of training data as 18000 as an example, Figure 11 illustrates the true and forecasting load under different times, when the parameter of data dimension expansion is 16 and the number of LSTM is 2. It can be seen that under different time and meteorological data conditions, the load is changed dramatically, but the load forecasting can be also close to the actual value no matter how the actual load changes. In order to describe the algorithm performance more clearly, Figures 12 and 13 show the statistical error of the load forecasting. When the number of training data increases, the forecasting error will decrease and the forecasting performance can be improved. From the figures, when the number of training data is 18000, the MAE and RMSE are only 94.87 and 146.19, respectively. If the number of training data is more than 9000, the parameter R^2 is close to or larger than 0.8. Therefore, through the statistical analysis, it can be concluded that the proposed algorithm can fully satisfy for practical application.

Table 1 describes the statistical error analysis with different numbers of LSTM, when the number of training data is 18000. As expected, when the number of LSTM increases, the load forecasting performance can be improved dramatically. Thus, without considering the training time cost, it is a better method to increase number of LSTM for performance improvement.

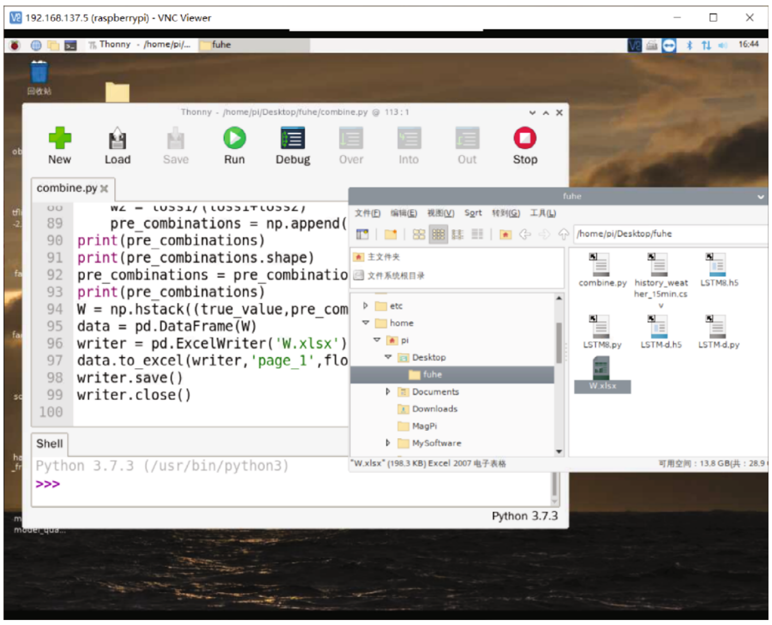
5.4. Performance Comparison. In this section, there are two existing load algorithms: (1) the training data is straightly used for training with LSTM (no data dimension expansion+no ensemble learning), and (2) the training data dimension is expanded at first, and then the LSTM is used for training (data dimension expansion+no ensemble learning) chosen for algorithm comparison. Figures 14–17 describe the statistical error comparison of different



```
#气压
for i in range(x-1):
    X[i][4]=X[i][4]/780
    X[i][4]=round(X[i][4],3)
Y=np.zeros((x-1,1))
for i in range(x-1):
    Y[i]=X[i][5]/1000
X = X [:,0:5]
X = X.reshape(8000,1,5)
network1 = keras.models.load_model('/home/pi/Desktop/fuhe/LSTM8.h5')
network2 = keras.models.load_model('/home/pi/Desktop/fuhe/LSTM-d.h5')
predict1 = network1.predict(X)
predict2 = network2.predi
pre1 = predict1 * 1000
pre2 = predict2 * 1000
true_value = Y * 1000
pre_combinations = np.arr
w1 = 1/2
w2 = 1/2
pre_combinations = np.app
for i in range(1,8000):
```

```
Python 3.7.3 (/usr/bin/python3
>>>
```

FIGURE 20: The description of load forecasting model import.



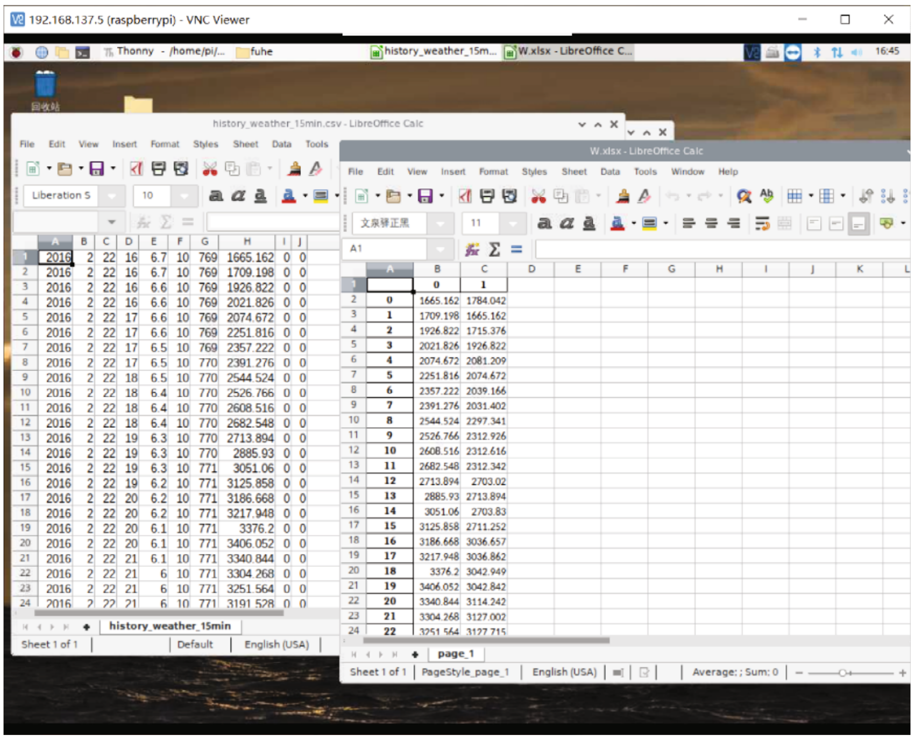
```
    pre_combinations = np.append(
print(pre_combinations)
print(pre_combinations.shape)
pre_combinations = pre_combinatio
print(pre_combinations)
W = np.hstack((true_value,pre_com
data = pd.DataFrame(W)
writer = pd.ExcelWriter('W.xlsx')
data.to_excel(writer,'page_1',flo
writer.save()
writer.close()
```

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

192.168.137.5 (raspberrypi) - VNC Viewer
Thonny - /home/pi/Desktop/fuhe/combine.py @ 113 : 1
New
Load
Save
Run
Debug
Over
Into
Out
Stop
combine.py
Shell
Python 3.7.3
fuhe
/home/pi/Desktop/fuhe
"W.xlsx" (198.3 KB) Excel 2007 电子表格

FIGURE 21: The description of load forecasting save.

algorithms, respectively. From the experiment results, it can be seen that the load forecasting performance of all three algorithms becomes better, when the number of training data increases. Taking 15000 training data as an example, the MAE, RMSE, MAPE, and R^2 of the algorithm are 107.76, 162.73, 0.09, and 0.87, respectively. However, the four error statistical parameters of the data dimension expansion+no ensemble learning method, which the performance is more close to the proposed algorithm, are 161.34, 213.94, 0.13, and 0.78. Thus, according to the statistical error analysis, the proposed algorithm has the best load forecasting performance among three methods. The reason can be concluded as follows. The data dimension expansion technology can describe the feature of training data more clearly. Moreover, the ensemble learning can improve the efficiency of offline learning.


192.168.137.5 (raspberrypi) - VNC Viewer
Thonny - /home/pi/...
fuhe
history_weather_15m...
W.xlsx - LibreOffice C...
16:45
history_weather_15min.csv - LibreOffice Calc
W.xlsx - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation S
10
文泉驿正黑
11
A1
history_weather_15min
Sheet 1 of 1
Default
English (USA)
page_1
PageStyle_page_1
Average: ; Sum: 0

	A	B	C	D	E	F	G	H	I	J
1	2016	2	22	16	6.7	10	769	1665.162	0	0
2	2016	2	22	16	6.7	10	769	1709.198	0	0
3	2016	2	22	16	6.6	10	769	1926.822	0	0
4	2016	2	22	16	6.6	10	769	2021.826	0	0
5	2016	2	22	17	6.6	10	769	2074.672	0	0
6	2016	2	22	17	6.6	10	769	2251.816	0	0
7	2016	2	22	17	6.5	10	769	2357.222	0	0
8	2016	2	22	17	6.5	10	770	2391.276	0	0
9	2016	2	22	18	6.5	10	770	2544.524	0	0
10	2016	2	22	18	6.4	10	770	2526.766	0	0
11	2016	2	22	18	6.4	10	770	2608.516	0	0
12	2016	2	22	18	6.4	10	770	2682.548	0	0
13	2016	2	22	19	6.3	10	770	2713.894	0	0
14	2016	2	22	19	6.3	10	770	2885.93	0	0
15	2016	2	22	19	6.3	10	771	3051.06	0	0
16	2016	2	22	19	6.2	10	771	3125.858	0	0
17	2016	2	22	20	6.2	10	771	3186.668	0	0
18	2016	2	22	20	6.2	10	771	3217.948	0	0
19	2016	2	22	20	6.1	10	771	3376.2	0	0
20	2016	2	22	20	6.1	10	771	3406.052	0	0
21	2016	2	22	21	6.1	10	771	3340.844	0	0
22	2016	2	22	21	6	10	771	3304.268	0	0
23	2016	2	22	21	6	10	771	3251.564	0	0
24	2016	2	22	21	6	10	771	3191.528	0	0

	A	B	C
1		0	1
2	0	1665.162	1784.042
3	1	1709.198	1665.162
4	2	1926.822	1715.376
5	3	2021.826	1926.822
6	4	2074.672	2081.209
7	5	2251.816	2074.672
8	6	2357.222	2039.166
9	7	2391.276	2031.402
10	8	2544.524	2297.341
11	9	2526.766	2312.926
12	10	2608.516	2312.616
13	11	2682.548	2312.342
14	12	2713.894	2703.02
15	13	2885.93	2713.894
16	14	3051.06	2703.83
17	15	3125.858	2711.252
18	16	3186.668	3036.657
19	17	3217.948	3036.862
20	18	3376.2	3042.949
21	19	3406.052	3042.842
22	20	3340.844	3114.242
23	21	3304.268	3127.002
24	22	3251.564	3127.715

FIGURE 22: The output description of the load forecasting.

5.5. Hardware Transplantation Experiment. In order to show the performance of the proposed algorithm for practical application, in this paper, the raspberry pi shown in Figure 18 is used for hardware transplantation [25]. First, the deep learning tools, such as Tensorflow and Keras, are installed, and then, all the libraries for the experiment are configured. In the following, the transplantation are described by the trained load forecasting model and testing data.

According to Figure 19, the testing data set "history_weather_15min.csv" is imported in raspberry pi. The month, hour, wind speed, temperature, and pressure in the testing data are chosen for the load forecasting. Then, the obtained load forecasting model in the offline phase is loaded in the hardware platform which is shown in Figure 20. At last, the actual load and load forecasting are written in files "W.xlsx." From the output shown in Figures 21 and 22, columns B and C describe the actual load and load forecasting, respectively.

6. Conclusions

In this article, a deep learning-based loading forecasting algorithm for power IOT is proposed. In the proposed algorithm, two data preprocessing, min-max normalization and hampel filter, are used to construct the fingerprint of the training data. Then, matrix multiplication method is proposed to extract the fingerprint feature by dimension expansion. Finally, the ensemble learning using multiple LSTM networks is proposed for offline training and obtain power load forecasting models. In the proposed algorithm, the data dimension expansion can obtain more features for training data. The ensemble learning using LSTM can make fully use of the timing sequence of training and improve the generalization performance. The experiment is carried out to evaluate the forecasting performance. Through the experiment results, it can be seen that the proposed algorithm has better load forecasting performance than chosen existing approaches.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61771256).

References

[1] J. H. Enslin, "Power system infrastructure: do we face a complete power-electronics-based power system and energy-storage infrastructure?," *IEEE Power Electronics Magazine*, vol. 3, no. 2, pp. 42–45, 2016.

[2] F. Rahimi, A. Ipakchi, and F. Fletcher, "The changing electrical landscape: end-to-end power system operation under the transactive energy paradigm," *IEEE Power and Energy Magazine*, vol. 14, no. 3, pp. 52–62, 2016.

[3] L. Wang, L. Zhang, C. Xu, H. Wu, Y. Li, and H. Sun, “Three-dimensional maturity model of regional power users against the background of the ubiquitous power internet of things,” *IEEE Access*, vol. 8, pp. 20215–20223, 2020.

[4] Z. Zhai, L. Jia, Y. Wang, Y. Ma, W. Jing, and Z. Zhang, “Research on ubiquitous power internet of things architecture,” in *2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2)*, pp. 435–439, Changsha, China, 2019.

[5] Q. Wang and Y. G. Wang, “Research on power internet of things architecture for smart grid demand,” in *2018 2nd IEEE Conference on Energy Internet and Energy System Integration*, pp. 1–9, Beijing, China, 2018.

[6] M. Imani and H. Ghassemian, “Electrical load forecasting using customers clustering and smart meters in internet of things,” in *2018 9th International Symposium on Telecommunications (IST)*, pp. 113–117, Tehran, Iran, 2018.

[7] J. Li, Y. Ren, S. Fang, K. Li, and M. Sun, “Federated learning-based ultra-short term load forecasting in power internet of things,” in *2020 IEEE International Conference on Energy Internet (ICEI)*, pp. 63–68, Sydney, NSW, Australia, 2020.

[8] A. Y. Saber and T. Khandelwal, “IoT based online load forecasting,” in *2017 Ninth Annual IEEE Green Technologies Conference (GreenTech)*, pp. 189–194, Denver, CO, USA, May 2017.

[9] W. Charytoniuk, M. S. Chen, and P. Van Olinda, “Nonparametric regression based short-term load forecasting,” *IEEE Transactions on Power Systems*, vol. 13, no. 3, pp. 725–730, 1998.

[10] W. Christiaanse, “Short-Term load forecasting using general exponential smoothing,” *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-90, no. 2, pp. 900–911, 1971.

[11] S.-J. Huang and K.-R. Shih, “Short-term load forecasting via ARMA model identification including non-Gaussian process considerations,” *IEEE Transactions on Power Systems*, vol. 18, no. 2, pp. 673–679, 2003.

[12] J. Che and J. Wang, “Short-term load forecasting using a kernel-based support vector regression combination model,” *Applied Energy*, vol. 132, pp. 602–609, 2014.

[13] S. Singh, S. Hussain, and M. A. Bazaz, “Short term load forecasting using artificial neural network,” in *2017 Fourth International Conference on Image Information Processing (ICIIP)*, pp. 1–5, Shimla, India, December 2017.

[14] Z. Deng, B. Wang, Y. Xu, T. Xu, C. Liu, and Z. Zhu, “Multi-scale convolutional neural network with time-cognition for multi-step short-term load forecasting,” *IEEE Access*, vol. 7, pp. 88058–88071, 2019.

[15] H. Shi, M. Xu, and R. Li, “Deep learning for household load forecasting-a novel pooling deep RNN,” *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2018.

[16] M. S. Hossain and H. Mahmood, “Short-term load forecasting using an LSTM neural network,” in *2020 IEEE Power and Energy Conference at Illinois (PECI)*, pp. 1–6, Champaign, IL, USA, February 2020.

[17] M. Jawad, M. S. A. Nadeem, S. Shim et al., “Machine learning based cost effective electricity load forecasting model using correlated meteorological parameters,” *IEEE Access*, vol. 8, pp. 146847–146864, 2020.

[18] S. Bhowmik, B. Jelfs, S. P. Arjunan, and D. K. Kumar, “Outlier removal in facial surface electromyography through Hampel filtering technique,” in *2017 IEEE Life Sciences Conference (LSC)*, pp. 258–261, Sydney, NSW, Australia, 2017.

[19] N. Singh and P. Singh, “Exploring the effect of normalization on medical data classification,” in *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, pp. 1–5, Gandhinagar, India, 2021.

[20] H. Rizk, M. Torki, and M. Youssef, “CellinDeep: robust and accurate cellular-based indoor localization via deep learning,” *IEEE Sensors Journal*, vol. 19, no. 6, pp. 2305–2312, 2019.

[21] G. I. Webb and Z. Zheng, “Multistrategy ensemble learning: reducing error by combining ensemble learning techniques,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 980–991, 2004.

[22] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, “Short-term residential load forecasting based on LSTM recurrent neural network,” *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.

[23] F. Huang, G. Xie, and R. Xiao, “Research on ensemble learning,” in *2009 International Conference on Artificial Intelligence and Computational Intelligence*, pp. 249–252, Shanghai, China, 2009.

[24] N. Alon, A. Gonen, E. Hazan, and S. Moran, “Boosting simple learners,” in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 481–489, New York, NY, USA, 2021.

[25] A. Nadjaran Toosi, J. Son, and R. Buyya, “CLOUDS-Pi: a low-cost raspberry-pi based micro data center for software-defined cloud computing,” *IEEE Cloud Computing*, vol. 5, no. 5, pp. 81–91, 2018.