Multi-Step-Ahead Time Series Prediction Method with Stacking LSTM Neural Network

XiaoFeng Wang*
School of Science
Xi'an University of Technology
Xi'an, China
e-mail: xfwang66@sina.com.cn

Ying Zhang
School of Science
Xi'an University of Technology
Xi'an, China
e-mail: zhangying__work@163.com

Abstract—**The issue of multi-step-ahead time series prediction is a daunting challenge of predictive modeling. In this work, we propose a multi-output iterative prediction model with stacking LSTM neural network (MO-LSTMs). In the proposed model, we utilize a stacking LSTM network that consists of multiple hidden layers to learn the features of time series data, and use the dropout algorithm to improve the generalization ability and robustness of the deep learning method. In our stacking LSTM neural network, each hidden layer contains different neural units and the memory state of the cells in each layer are reset. The proposed method solves the problem that the single LSTM network structure is difficult to maintain the time-sequence characteristics between samples in the training process. Additional, in the prediction stage, we utilize the strategy of multi-output iterative prediction to reduce the errors accumulation and errors propagation for long-term time series prediction. It also reduces the computational complexity of the iterative strategy. The simulation experiments are conducted on actual engineering datasets, and the results show that the proposed method is provided with better prediction performance.**

Keywords-multi-step-ahead prediction; time series prediction; recurrent neural network; lstm network

I. Introduction

Multi-step-ahead time series prediction has been a research hotspot in many fields. In one-step-ahead prediction, the predictor uses all or some of the observations to predict a time-step immediately following the latest observation. Predicting two or more steps ahead, called as multi-step-ahead prediction. Unlike one-step-ahead prediction, multi-step-ahead prediction is up against more uncertain factors. The error accumulation and information lack make multi-step-ahead prediction more difficult [1]. Thus, for the time series data that come from different fields, it has significantly practical application value to select suitable modeling strategy for multi-step-ahead time series prediction.

There are two kinds of multi-step-ahead prediction strategies, namely iterated strategies and direct strategies. Work [2] proposed a nonlinear-learning ensemble method, called EnsemLSTM, for multi-step-ahead time series prediction. However, since the iterative strategy is adopted, the inputting distribution in future time step will change when a long value is predicted [3], and thus will result in a large accumulative error. The training process of iterative prediction model is essentially a one-step prediction, thus its model optimization is also a one-step error optimization. Of course, there is some mismatch between the purpose of single-step optimization and multi-step prediction, which leads to optimistic error estimation during training [4]. If the real one-step model is not identified during training, then the issue of inaccurate prediction will appear, this is almost always true in the prediction for nonlinear problems [5]. Cox [6] first proposed direct strategy for multi-step-ahead time series prediction. In this work, authors used past observation data to establish a set of prediction models for each horizon, and then minimized the squared multi-step-ahead errors rather than the squared one-step-ahead error [7].

Multi-step prediction methods using direct strategies are also abundant [8-11]. In the work [12], an ensemble method based on decision tree, gradient boosted trees and random forest was proposed, in which, the directed strategy is adopted for wind speed multi-step time series prediction. In the work [13], a multi-output deep LSTM neural network model was proposed for multi-step-ahead time series prediction. In work [14], a deep network model with self-encoding decoding structure was proposed for multi-step-ahead prediction of time series. This model was used for bus travel time prediction, and then, was used for rain forecasting in work [15]. Since using direct strategies for multi-step-ahead prediction, the prediction accuracy of above models was improved to some extent. However, the model with direct prediction strategy can't take the correlation among the predicted values into account, so the prediction accuracy will become worse and worse as the prediction length increases. Work [16] reported a multi-output support vector regression (M-SVR), multi-input and multi-output (MIMO) prediction strategies for multi-step prediction of time series.

In this work, aiming at the problem that the prediction accuracy of long-term time series prediction will decrease with the increasing of time, we propose a multi-output iterative prediction model with stacking-based LSTM neural network. The main contributions are as follows.

1) We propose a multi-output iterative prediction (MO-LSTMs) model based on stacking LSTM neural network. In the proposed MO-LSTMs model, the stacking LSTM network is provided with different hidden layers and neural units, in which, the memory states of the cells in each layer

are reset. The proposed method solves the problem that the single LSTM network structure is difficult to maintain the time-sequence characteristics between samples in the training process.

2) In the prediction stage, we utilize the strategy of multi-output iterative prediction to reduce the errors accumulation and errors propagation for long-term time series prediction. It also reduces the computational complexity of the iterative strategy for long-term multi-step prediction.

The simulation experiments are product on actual engineering datasets.

II. The Proposed MO-LSTMs Model

In this section, we describe the proposed multi-output iterative prediction model with stacking LSTM neural network (MO-LSTMs). In the proposed MO-LSTMs model, we introduce the stacking LSTM neural network with different hidden layers and neural units, and combine with dropout algorithm to improve the generalization ability and robustness of the model. Meanwhile, the memory state of the cells in each hidden layer are reset, which also improves the memory performance of the network model. The architecture of the proposed network model is shown in the Fig.1.

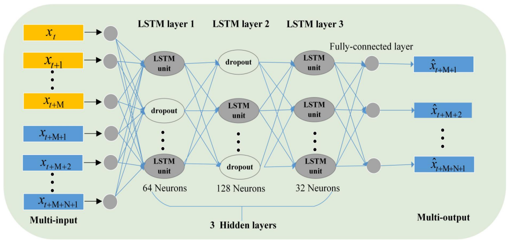

Figure 1. The architecture of the proposed MO-LSTMs model.

A. Long Short-Term Memory Network

The Long Short-Term Memory (LSTM) network [17] is a variant of the recurrent neural network. It consists of two states and three gates, where, the state c_t is called memory cell, and is used to handle the long-term dependencies, and another state $\tilde{c}_t$ represents the candidate state at time t. LSTM uses the gating mechanism that includes an input gate i_t, a forget gate f_t and an output gate o_t to control the state c_t and $\tilde{c}_t$. LSTM uses the gating mechanism and memory cells to control the accumulation of information, which effectively solves the problem of gradient explosion or disappearance in simple recursive neural network. Therefore, it is widely used in the prediction of time series data. In this work, we stack the LSTM neural networks that consists of different hidden layers and neural units to form a stacking LSTM neural network model. Fig.2 shows the schematic diagram of the LSTM neural network.

In Fig.2, X_t represents the sample of time series data, h_{t-1} represents outputted information at time $t-1$. The feature learning process of LSTM network is influenced by the output value h_{t-1} of the previous moment and the input value X_t at the current time. Therefore, at time t, the input values of the memory cell are h_{t-1} and X_t.

The function of the forgetting gate f_t is to control the amount of information that needs to be forgotten in the internal state c_{t-1} at $t-1$. The function of the input gate i_t is to control the amount of information of candidate state $\tilde{c}_t$ that needs to be saved at the current time. f_t and i_t are mapped to [0,1] by the Sigmoid function, and $\tilde{c}_t$ is mapped to [-1,1] by the tanh activation function. The output gate o_t controls the amount of information of the internal state c_t that needs to be transmitted to h_t. The relationship of the LSTM states and three gates is expressed in Equation (1)-(6).

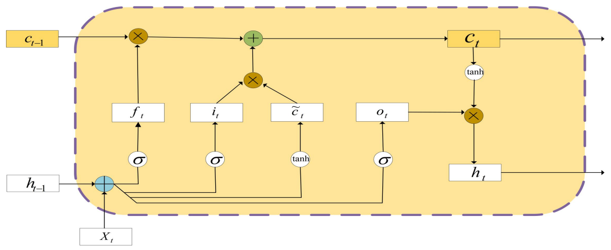

Figure 2. The schematic diagram of the LSTM network.

$$i_t = \sigma(W_{Xi} \cdot [X_t, h_{t-1}] + b_i) \tag{1}$$
$$f_t = \sigma(W_{Xf} \cdot [X_t, h_{t-1}] + b_f) \tag{2}$$
$$\tilde{c}_t = \tanh(W_{Xc} \cdot [X_t, h_{t-1}] + b_c) \tag{3}$$
$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \tag{4}$$
$$o_t = \sigma(W_{Xo} \cdot [X_t, h_{t-1}] + b_o) \tag{5}$$
$$h_t = o_t \otimes \tanh(c_t) \tag{6}$$

Here, W_{Xi}, W_{Xf}, W_{Xo} and W_{Xc} represent the weight matrix of the input gate, the forget gate, the output gate and the candidate state, respectively. b_i, b_f, b_o and b_c represent the biases matrix of the input gate, the forget gate, the output gate and the candidate state, respectively. Besides, operation $\cdot$ indicates matrix multiplication and the operation $\otimes$ indicates element-wise multiplication, $[X_t, h_{t-1}]$ represents the operation that connects two inputted vectors X_t and h_{t-1} into a long vector.

B. Dropout Algorithm

To improve the generalization ability, we introduce the dropout algorithm into the proposed MO-LSTMs model. The idea behind in the Dropout algorithm is to randomly discard neurons with a certain probability p during the training process of the neural network, and the purpose is to prevent the co-adaptation of neurons [18]. Therefore, the average number of the activated neurons is p times of the number of original neurons. However, during model testing, all neurons can be activated, which will result in inconsistent network output. To alleviate this problem, the outputted value of each neuron in testing model needs to be multiplied by p. The experimental result shows that it works best when p =0.5. That is, in the training process, one half of the neurons are discarded and others are survived. By this way, the network results that are generated randomly are diversified, and thus the robustness of the network model is improved. From the perspective of reducing the complexity of the model structure, the dropout algorithm is also considered as an effective method to handle the over-fitting problems of LSTM models ([19] and [20]).

III. Multi-Step-Ahead Prediction

In this section, using the proposed MO-LSTMs model, we present a multi-step-ahead prediction method for time series data. Our method consists of three phases: data preprocessing, model training, and predicting.

A. Data Preprocessing

1) Data Normalization

Let $T=\{x_1,x_2,\cdots,x_t,\cdots\}(t=1,2,\cdots,n)$ represents the time series sample datasets. We normalize T into the interval [0,1] using the statistical limit values. First, we estimate the maximum value Max and the minimum value Min in the sample datasets. Then we map the maximum value to 1, and the minimum value is mapped to zero.

$$x_t'=\frac{x_t-Min}{Max-Min} \tag{7}$$

Where, x_t and x_t' represent the raw signal value and the normalized value at time t, respectively, $T'\{x_1',x_2',\cdots,x_n'\}$ represents the normalized time series.

For the output of the predicting process, we use the following formula to de-normalize to restore the physical significance of the outputting results.

$$\hat{x}_t=Min+x_t'\times(Max-Min) \tag{8}$$

Where, x_t' represents the prediction value outputted from the MO-LSTMs model, and $\hat{x}_t$ represents de-normalized prediction value.

2) Data slicing

For the proposed model, due to a multi-output structure is used in the training process, thus the format of the training data should fit for the multi-output format of the model. In this work, we introduce a data slicing mechanism that use the sliding window function $Slicing(\cdot)$ to segment the normalized time series datasets $T'\{x_1',x_2',\cdots,x_t',\cdots\}$ into a sequence of fragments. Assuming the M and N are the sequence lengths of the input and output sequences, respectively. The sliding step is 1, and the length of our predicted data is L. At first, we need to take a piece of data $\{x_{n-L}',x_{n-L+1}',\cdots,x_n'\}$ from dataset T' as the testing data. Here, the length of $\{x_{n-L}',x_{n-L+1}',\cdots,x_n'\}$ is L, and $L>N$. Then we carry out data slicing, the segmented training datasets $\{train\_x,train\_y\}$ is shown as follows.

$$\{train\_x:train\_y\}=\left\{\begin{Bmatrix}\{x_1',x_2',\cdots,x_{1+M}'\}\\ \{x_2',x_3',\cdots,x_{2+M}'\}\\ \vdots\\ \{x_t',x_{t+1}',\cdots,x_{t+M}'\}\\ \vdots\end{Bmatrix}:\begin{Bmatrix}\{x_{2+M}',\cdots,x_{2+M+N}'\}\\ \{x_{3+M}',\cdots,x_{3+M+N}'\}\\ \vdots\\ x_{t+M+1}',\cdots,x_{t+M+N+1}'\\ \vdots\end{Bmatrix}\right\} \tag{9}$$

Where, M and N are super-parameters, which need to be adjusted through experiments.

B. The Training Process

In this work, we send the training dataset $\{train\_x,train\_y\}$ into the proposed MO-LSTMs modle, and use dropout algorithm and Adam optimizer to update model parameters. The training process is shown in Fig. 3. The implementation procedure is described as follows.

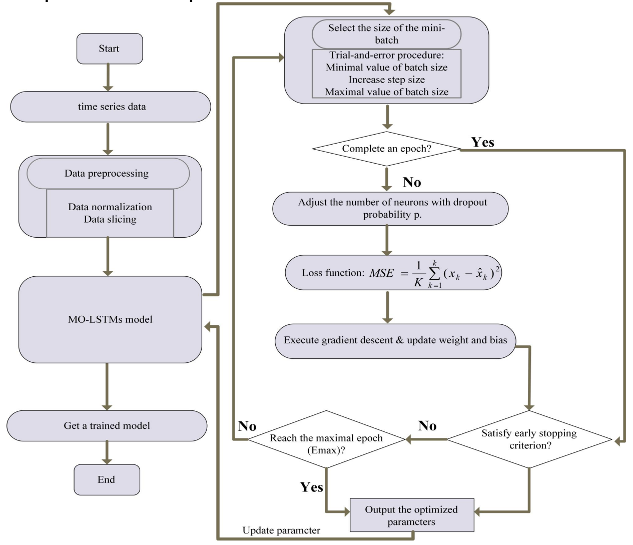
Start
time series data
Data preprocessing
Data normalization
Data slicing
MO-LSTMs model
Get a trained model
End
Select the size of the mini-batch
Trial-and-error procedure:
Minimal value of batch size
Increase step size
Maximal value of batch size
Complete an epoch?
Yes
No
Adjust the number of neurons with dropout probability p.
Loss function: $MSE=\frac{1}{K}\sum_{k=1}^{k}(x_k-\hat{x}_k)^2$
Execute gradient descent & update weight and bias
Satisfy early stopping criterion?
No
Reach the maximal epoch (Emax)?
No
Yes
Output the optimized paramcters
Update paramcter

Figure 3. The training process of proposed method.

Step 1. Parameter initialization. Set the maximum number of epoch, the number of hidden layers, the number of neurons, and the mini-batch size.

Step 2. According to the characteristics of the time series data, determine the size of M and N, the exact values need to be obtained via experiment.

Step 3. Update model parameters.

(a) Check whether an epoch is completed. If an epoch is not completed, the number of neurons in the hidden layer should be adjusted according to the dropout probability p.

(b) Compute the loss function in terms of MSE. Then the weights W and bias b can be calculated using the Adam optimizer to perform gradient descent algorithm, and utilize the next mini-batch to repeat the Step 3. If an epoch is completed, proceed to the next step. Here, generally, mini-batch-sized samples are randomly selected from the training data set, which will result in the time-ordered of samples being shuffled. Therefore, in this work, we set each small batch sample to be an ordered output to ensure the time-ordered of samples.

Step 4. We end the computation process via terminating conditions, which include the early stopping and the maximum epoch. In this work, we set the tolerance for early stop to be 100. That is, if the value of the loss function does not decrease over 100 consecutive epochs, the prediction accuracy will not be improved, which will trigger to stop the computation. If the maximal number of epochs is reached, the training process will be stopped. Otherwise, update the epoch, and repeat the steps 2 and step 3.

Finally, save the optimized model parameters, including the maximal epoch ($E_{\max}$), the number of neurons, the learning rate (α) , the mini-batch size, the dropout probability (p), the weight matrix W and the bias vector b .

According to works [21], there is not any theoretical knowledge that can predefine the structure of network model for specific data. Therefore, in practical application, it is a feasible solution to select the hyper-parameters by trial-and-error experiments. In this work, to achieve the trade-off between the learning performance and computational complexity, we structure a 3-layer multi-output model with stacked LSTM network. In the proposed MO-LSTMs model, the first LSTM layer contains 64 neurons, the second LSTM layer contains 128 neurons and the third LSTM layer contains 32 neurons. The numbers of the layers and the neurons are determined via extensive experiments.

C. *Multi-Output Iterative Prediction Strategy*

For the multi-step-ahead prediction, there are two kinds of strategies, iterative prediction and direct prediction. The iterative prediction method uses the predicted values as an input to predict the following outputting values. When the length L of the prediction sequence is too long, the iteration-based prediction method will generate large cumulative error in the multi-step-ahead prediction. Therefore, such prediction method is not going to work for predicting long sequences. The direct prediction method is tend to construct multiple prediction models whose number is the same as the predicted length, and each model predicts the corresponding component. To generate predicted values with length L, the direction-based prediction method needs to construct L models. If L is too large, the computing overhead will explode. Additional, the direct prediction method reckon without the correlation among the predicted values, and thus the prediction accuracy will become worse and worse with passage of time. In order to accurately predict a longer time series, in this work, we combine multi-step iteration with multi-step direct prediction, and propose a multi-output iterative prediction model. Fig.4. shows the schematic diagram of the proposed prediction strategy.

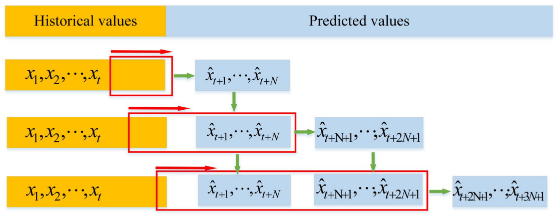
Historical values
Predicted values
$x_1, x_2, \cdots, x_t$
$\hat{x}_{t+1}, \cdots, \hat{x}_{t+N}$
$\hat{x}_{t+N+1}, \cdots, \hat{x}_{t+2N+1}$
$\hat{x}_{t+2N+1}, \cdots, \hat{x}_{t+3N+1}$

Figure 4. The schematic diagram of the proposed prediction strategy.

IV. Experimental Results and Analysis

In this section, we test the performance of the proposed method by using the gyroscope temperature data (dataset 1) of the control system of the equipment in space measurement and control engineering. Our method is implemented and tested using TensorFlow 1.10.0, anacoda3-5-1.0 and keras2.22. We run programs in the computer with i7-6800K CPU, GTX1080Ti GPU, 32.00GB RAM.

A. *Evaluation Metrics*

We use four statistical metrics, named MAE (mean absolute error), RMSE (root mean square error), MAPE (mean absolute percent error) and R (Correlation coefficient) to quantitatively assess the performance of the proposed method. These metrics can be described as follows.

$$MAE = \frac{1}{K}\sum_{k=1}^{k}\left|x_k - \hat{x}_k\right| \tag{10}$$

$$RMSE = \sqrt{\frac{1}{K}\sum_{k=1}^{k}(x_k - \hat{x}_k)^2} \tag{11}$$

$$MAPE = \frac{1}{K}\sum_{k=1}^{K}\left|\frac{x_k - \hat{x}_k}{x_k}\right| \times 100\% \tag{12}$$

$$R = \frac{\sum_{k=0}^{K}(x_k - \bar{x})(\hat{x}_k - \bar{\hat{x}})}{\sqrt{\sum_{k=0}^{K}(x_k - \bar{x})^2} \cdot \sqrt{\sum_{k=0}^{K}(\hat{x}_k - \bar{\hat{x}})^2}} \tag{13}$$

Here, K represents the total number of the data, x_k represents the real value at time k , $\hat{x}_k$ represents the corresponding prediction value. $\bar{\hat{x}}$ and $\bar{x}$ represent the mean of the predicted values and the actual values, respectively.

B. *Experimental Results*

In this section, we verify the effectiveness of the proposed MO-LSTMs model via experimental results. Our experiment consists of two parts. First, we explore the influence of parameters M and N for the proposed model, and then determine the values of M and N . Second, we evaluate the prediction performance of the proposed model via four evaluation indicators.

1) Determination of parameters M and N

In the proposed model, the parameters M and N of represent the dimensions of the inputted sample and the dimensions of the outputted sample, respectively. Therefore, the scaling relation between M and N will seriously affect the predictive performance of the proposed model. To achieve good prediction performance, according to the characteristics of the testing data, we take M =72 and 144, N =6,12 and 24. We conduct expensive experiments on the dataset 1, the experimental results are shown in the table I.

TABLE I. Results of MO-LSTMs Model on Dataset 1

	M = 72			M = 144		
	N=6	N=12	N = 24	N = 6	N = 12	N = 24
RMSE	0.7753	2.5589	2.7663	4.1294	8.6017	12.4685
MAE	0.6506	2.2705	2.4245	3.5723	6.2626	9.5120
MAPE	0.0379	0.0997	0.1042	0.1734	0.2070	0.4634
R	0.9981	0.9799	0.9764	0.9472	0.8083	0.7182

Table I shows the prediction results of the proposed MO-LSTMs model with different values M and N . As can be seen from table Ⅰ, in the case of M =72, the prediction performance is better than that of M =144. Among all the testing results, in the case of M =72 and N =6, RMSE, MAE and MAPE achieve the lowest value, respectively, and R achieves the highest value, which indicate that the prediction performance of the model is the best. As can be seen from the experimental results, the values of parameters M and N have great influence on the prediction performance of the model. In this work, for the dataset 1, we let M =72 and N =6.

2) performance evaluation

To verify the performance of the proposed MO-LSTMs model, we compare the MO-LSTMs model with LSTM network model, least squares support vector machine (LS-SVM) model, back propagation neural network (BP-NN) model and the autoregressive (AR) model.

TABLE II. RESULTS OF DIFFERENT MODELS DATASET 1

model	RMSE	MAE	MAPE(%)	R
MO-LSTMs	0.7753	0.6506	0.0379	0.9981
LSTM	1.9975	1.7532	1.7532	0.9905
LS-SVM	3.9115	2.9617	1.1337	0.9667
ANN	2.4344	1.6436	0.2072	0.9906
AR	10.6458	8.0947	1.5998	0.7689

Table II shows the prediction results of different models on dataset 1. It can be seen that, comparing with the compared models, the proposed model has lower error for RMSE, MAE and MAPE metrics. The reason is that the proposed method has better nonlinear representation performance. At the same time, we adopt the multi-output iteration strategy to improve the cumulative error of direct prediction and iterative prediction for long-term prediction. It is proved that the proposed method has better predictive performance for multi-step-ahead time series prediction.

V. CONCLUSION

With the development of the information age, explosive information is growing at exponential rate, the issue of time series data analysis has attracted great attention. Multi-step-ahead time series prediction has important application in practical engineering. In the proposed MO-LSTMs model, we utilize a stacking LSTM network that consists of different hidden layers and neural units to learn the features of time series data. The simulation experiments are product on actual engineering datasets, and the results show that the proposed method is provided with better prediction effects on the actual engineering datasets.

In practical applications, the performance requirements for the prediction methods are different because the time series data in different fields have different characteristics. Therefore, in the subsequent research, we will consider the fusion strategy for prediction model, and try to present better models to meet different engineering needs.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant No.61772416; the Key Laboratory Project of the Education Department of Shaanxi Province under Grant No.17JS098; Shaanxi province technology innovation guiding project, No.2018XNCG-G-02.

REFERENCES

[1] Weigend. A. S, Gershenfeld. N. A, NATO Advanced Research Workshop on Comparative Time Series Analysis, Time Series Prediction-Forecasting the Future and Understanding the Past [C] 1994.

[2] Chen J, Zeng G Q, Zhou W, Wind speed forecasting using nonlinear-learning ensemble of deep learning time series prediction and extremal optimization[J], Energy Conversion and Management, 2018, 165:681-695.

[3] Venkatraman. A, Hebert. M. A, Improving multi-step prediction of learned time series models, in AAAI, pp. 3024–3030, 2015.

[4] Taieb. S.B, Machine learning strategies for multi-step-ahead time series forecasting, Universit Libre de Bruxelles, Belgium, 2014.

[5] Taieb. S. B. and Atiya. A. F, A bias and variance analysis for multistepahead time series forecasting, IEEE transactions on neural networks and learning systems, vol. 27, no. 1, pp. 62–76, 2016.

[6] Cox. D. R, Stat. JR, B, Soc, Prediction by exponentially weighted moving averages and related methods, 23 (1961) 414–422.

[7] Franses. PH, R. Legerstee, A unifying view on multi-step forecasting using an autoregression, J. Econ. Surveys 24 (2009) 389–401.

[8] Zhang L. Optimizing ANN training performance for chaotic time series prediction using small data size[J]. Int. J. Mach. Learn. Comput, 2018, 8(6): 606-612.

[9] Hung C, Hung C N, Lin S Y. Predicting time series using integration of moving average and support vector regression[J]. International Journal of Machine Learning and Computing, 2014, 4(6): 491.

[10] Delima A J P. Application of Time Series Analysis in Projecting Philippines Electric Consumption[J]. International Journal of Machine Learning and Computing, 2019, 9(5):694-699.

[11] Dingli A, Fournier K S. Financial time series forecasting–a deep learning approach[J]. Mach. Learn. Comput, 2017, 7(5): 118-122.

[12] Galicia. A, Talavera-Llames. R, Troncoso A, Multi-step forecasting for big data time series based on ensemble learning[J], Knowledge-Based Systems, 2019, 163: 830-841.

[13] Zhou. Y, Chang. F. J, Chang. L. C, Explore a deep learning multi-output neural network for regional multi-step-ahead air quality forecasts[J], Journal of cleaner production, 2019, 209: 134-145.

[14] Petersen. N. C, Rodrigues. F, Pereira. F. C, Multi-output bus travel time prediction with convolutional LSTM neural network[J], Expert Systems with Applications, 2019, 120: 426-435.

[15] Xingjian. S. H. I, Chen. Z, Wang. H, Convolutional LSTM network: A machine learning approach for precipitation nowcasting[C], Advances in neural information processing systems. 2015: 802-810.

[16] Bao. Y, Xiong. T, Hu Z, Multi-step-ahead time series prediction using multiple-output support vector regression[J], Neurocomputing, 2014, 129: 482-493.

[17] Hochreiter. S, Schmidhuber. J, Long short-term memory. Neural Computation, 1997, 9(8): 1735-1780.

[18] P. Baldi, P. Sadowski, The dropout learning algorithm[J], Artificial intelligence, 2014, 210: 8-122.7

[19] Hinton. G. E, Srivastava. N, Krizhevsky, Improving neural networks by preventing co-adaptation of feature detectors[J], arXiv preprint arXiv:1207.0580, 2012.

[20] Srivastava. N, Hinton. G, Krizhevsky. A, Dropout: a simple way to prevent neural networks from overfitting[J], The journal of machine learning research, 2014, 15(1): 1929-1958.

[21] Wu. Y, Yuan. M, Dong. S, Remaining useful life estimation of engineered systems using vanilla LSTM neural networks[J], Neurocomputing, 2018, 275: 167-179.