ACADEMIC
World Academic Union

ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 16, No. 1, 2021, *pp*.041-051

A Multi-input Time Series Prediction Model Based on CNN-BLSTM

Ting Xiao
Nanjing University of Information Science and Technology, Nanjing-210044, China
(Received January 25, 2021, accepted March 12, 2021)

Abstract. Real time series data sets are often composed of multiple variables. For the future trend of data, not only the historical value of the variables but also other implicit influence factors should be considered. In this paper, a deep neural network prediction model based on multivariable input multi-step output named CNN-BLSTM is proposed. CNN-BLSTM is mainly composed of convolutional neural network (CNN) and bi-directional long short memory network (Bi-LSTM). CNN is used to extract spatial features between variables of multivariate raw data, and Bi-LSTM is used to extract and encode features in time direction. The proposed CNN-BLSTM is able to predict the temperature based on a real-life meteorological data. The experimental results show that the prediction accuracy of the proposed CNN-BLSTM model is significantly better than several state-of-the-art baseline methods.

Keywords: time series, bi-directional long-short term memory, convolutional neural network, prediction, multivariable input

1. Introduction

Time series refers to the ordered set of some variables according to the sequence of time occurrence, which contains the general law that this variable changes with time. Such as monitoring data in meteorological management [1], household electricity consumption data [2], transaction data in the stock market [3], etc. In real life, the collected data is usually multivariate time series (MTS), and there may be complex dynamic interdependencie between different sequences. For example, road surface temperature changes are affected by factors such as temperature, rainfall and humidity. These interdependencies are important, but they are difficult to capture and analyze. The research shows that if the intrinsic correlation and historical characteristics of multivariate time series can be correctly mined, the accuracy of prediction variables can be significantly improved [3].

Researchers have come up with some ways to solve prediction problems. For example, the traditional statistical method integrated automatic regression moving average (ARIMA) model performs well in linear time series data forecasting [4], such as Kumar and Jain [5] used ARIMA to establish a model for predicting traffic noise time series, Ediger and Akar [6] used ARIMA model to predict Turkey's fuel demand and so on. ARIMA is good at predicting stationary time series data. However, for nonstationary time series data, ARIMA faces the problem of increasing smooth step size and difficulty. With the rapid development of machine learning, machine learning methods represented by artificial neural networks (ANN) have been widely used in the classification and prediction of time series because of its strong ability of nonlinear data processing and noise suppression [7,8,9,10]. However, ANN [11] is not as good as ARIMA in the low frequency range of data, and it is easy to produce overfitting in the high frequency range [12].

Recently, deep learning, an advanced machine learning in the field of artificial intelligence, is becoming more and more popular. It combines several machine learning methods to extract high-level expression from complex data by hierarchical learning process and using structure composed of multiple nonlinear transformations [13]. Due to the high computational performance, deep learning models have shown great advantages in automatically extracting and learning multivariate data features in large amounts of data, and have been successfully applied in many scientific fields, including computer vision [14], image classification [15,16], natural language processing [17,18], etc.

Long short-term memory network (LSTM) and convolutional neural networks (CNN)) are two popular deep learning models. LSTM is a recurrent neural network that collects extended sequential data for processing, representation and storage in hidden memory. It solves the problem that RNN is easy to produce gradient explosion or disappearance, which leads to its performance degradation. It becomes a state-of-the-

art predictive model to process sequence data in various applications [19], including visual recognition and description [20], continuous Speech recognition [21,22] etc. However, the actual data changes usually depend not only on the previous historical data sequence, but also on the later generated sequence. Elsheikh [23] et al. used a bidirectional LSTM (Bi-LSTM) algorithm to increase the input of LSTM by inputting one step of time series data into the network forward and backward respectively. Compared with LSTM, Bi-LSTM improves the training mechanism to capture information from the past data context while also capturing information from the future data context, so as to obtain more complete information and make predictions with higher accuracy.

CNN [24] is a biologically-inspired type of deep neural network (DNN) that has recently gained popularity due to its success in classification problems (e.g. image recognition [25] or time series classification [26]). The CNN consists of a sequence of convolutional layers, the output of which is connected only to local regions in the input. This is achieved by sliding a filter, or weight matrix, over the input and at each point computing the dot product between the two (i.e. a convolution between the input and filter). This structure allows the model to learn filters that are able to recognize specific patterns in the input data. Studies have shown that the combination of CNN and LSTM can be used to develop more accurate prediction models [27]. For example, Huang et al. [28] found that it is advantageous to use CNN for feature extraction and then input the feature value into the LSTM architecture. In the hybrid CNN-LSTM model, a one-dimensional CNN extracts the deep features of the main elements. Then, use the LSTM model to make predictions using these depth features. Such as Li et al. [29] use a joint CNN-LSTM model to predict collision risk, Barzegar R et al. [30] predict short-term water quality and so on.

Through the above discussion, this paper proposed a multi-variable input-multi-step output time series forecasting model. The model mainly includes two parts: a spatial feature extractor composed of CNN and a temporal feature encoder composed of BLSTM. Before the temporal feature encoder, the spatial feature extractor is used to extract the horizontal relationship of multi-dimensional variables, and then the encoder is used to learn the temporal relationship features obtained by the feature extractor, and make predictions accordingly.

The rest of this article is organized as follows: The second part is mainly an introduction to related neural networks. The third part is a detailed introduction to the framework and algorithm of the proposed CNN-BLSTM model. The fourth part is experiment and result analysis. The fifth part is summary and outlook.

2. Related Networks

2.1 Convolutional Neural Network

CNN is a neural network proposed by LeCun et al. [31] to effectively extract the original image features by simulating the perception mechanism of human natural vision. As shown in Figure 1, each node of the CNN is only connected to a local area in the input. The spatial extent of this connectivity is called the receptive domain of the node. Local connectivity is achieved by replacing the weighted sum of neural networks with convolution. In each layer of the convolutional neural network, the input is convolved with the weight matrix (also called the filter) to create a feature map. Unlike conventional neural networks, all values in the output feature map have the same weight. This means that all nodes in the output detect exactly the same pattern. The local connectivity and shared weights of neural networks reduce the total number of learnable parameters and improve training efficiency. Therefore, the essence behind the convolutional neural network is to learn a weight matrix in each layer so that it can extract the necessary translation invariant features from the input.

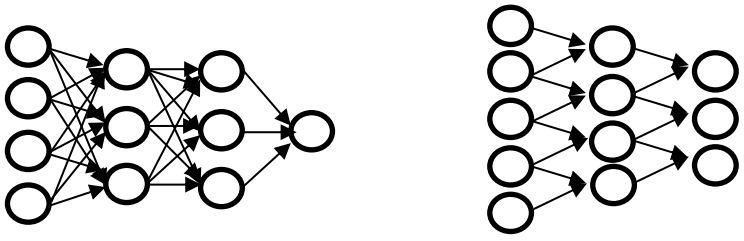


Fig. 1. A feedforward neural network with three layers (L) and a convolutional neural network with two layers and filter size 1 × 2(R).

The input of the convolutional layer is usually three-dimensional: the height, weight and number of channels. In the first layer this input is convolved with a set of M_1 three-dimensional filters applied over all the input channels to create the feature output map. Consider now a one-dimensional input $x = (x_t)_{t=0}^{N-1}$ of size N with no zero padding. The output feature map from the first layer is then given by convolving each filter w_h^1 for $h = 1, \dots, M_1$ with the input:

$$a^1(i,h) = w_h^1 * x(i) = \sum_{j=-\infty}^{\infty} w_h^1(j)x(i-j), \quad (1)$$

Where $w_h^1 \in R^{1\times k\times 1}$ and $a^1 \in R^{1\times N-k+1\times M_1}$. Since the number of input channels is one at this time, the weight matrix also has only one channel. Similar to the feedforward neural network, this output is then passed through the non-linearity $h(\cdot)$ to give $f^1 = h(a^1)$.

In each subsequent layer $l = 2, \dots, L$ the input feature map, $f^{l-1} \in R^{1\times N_{l-1}\times M_{l-1}}$, where $1 \times N_{l-1} \times M_{l-1}$ is the size of the output filter map from the previous convolution with $N_{l-1} = N_{l-2} - k + 1$, is convolved with a set of M_l filters $w_h^1 \in R^{1\times k\times M_{l-1}}$, $h = 1, \cdots, M_l$ to create a feature map $a^1 \in R^{1\times N_l - k\times M_l}$:

$$a^l(i,h) = (w_h^l * f^{l-1})(i) = \sum_{j=-\infty}^{\infty}\sum_{m=1}^{M_{l-1}} w_h^l(j,m)f^{l-1}(i-j,m). \quad (2)$$

Then through nonlinear output, we get $f^l = h(a^l)$. The filter size parameter k controls the receiving field of each output node. In the absence of zero padding, the convolution output in every layer has width $N_l = N_{l-1} - k + 1$ for $l = 1, \dots, L$. All elements in the feature map share the same weight. The output of the network after the L convolutional layer is a matrix f^L. Its size depends on the size of the filter used in the last layer and the number of filters. The weights in the model are trained according to requirements to minimize the error between the output f^L of the network and the real output [32].

2.1 From RNN to LSTM

Recurrent Neural Network (RNN) [33] is a neural network that establishes a directional loop connection between neurons. As shown in Figure 2, on the left is a simple RNN structure, x represents the vector of the input layer, s is the hidden layer memory unit of the network, U is the weight matrix from the input layer to the hidden layer, V is the weight matrix from the hidden layer to the output layer, W is the weight matrix of the previous value of the hidden layer as the input of this time, and o represents the value of the output layer; On the right is the expanded RNN structure at the time step, s_t、 x_t and o_t are the hidden layer state, input and output of the t-th step, respectively. Unlike general neural networks that can only map from input data to target vectors, the RNN can map the target vector from the entire history entered before, allowing the previously entered memory to be stored in the form of the internal state of the network, and training is performed through backward propagation according to time and the ownership value is shared. However, for time series with long-term correlation, the RNN model is prone to the problem of gradient explosion or disappearance during training[34], which seriously hinders its performance.

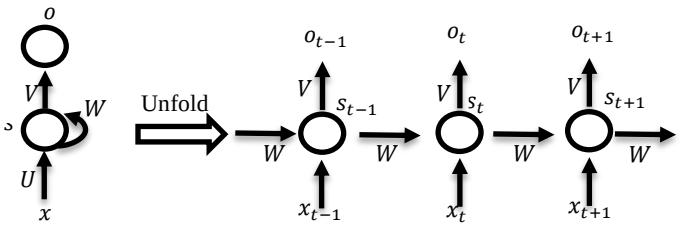


Fig.2. A simple RNN neuron, the left side is the unexpanded structure, and the right is the expanded structure

In order to solve the problem of disappearance or explosion of backward propagation errors during RNN model training, a network called LSTM [35] was proposed. It controls the input and output of information by adding three gate structures, so that the internal state of the unit can be adjusted. Figure 3 shows the internal condition of an LSTM "neuron", in which the forget gate f controls the input ratio of previous information, avoiding long-term dependence, and more effectively controlling the utilization of information in the unit state; The input gate i controls the input ratio of the current information; the output gate o controls the output state. The specific calculation formula of LSTM neural network is as follows:

$$i^t = \sigma\left(W^i x^t + V^i h^{t-1} + b^i\right)$$
$$f^t = \sigma\left(W^f x^t + V^f h^{t-1} + b^f\right)$$

$$o^t = \sigma\left(W^o x^t + V^o h^{t-1} + b^o\right) \quad (3)$$
$$c^t = f^t \odot c^{t-1} + i^t \odot tanh(W^c x^t + V^c h^{t-1} + b^c)$$
$$h^t = o^t \odot tanh(c^t)$$

Where $W \in R^{m\times n}$、$U \in R^{m\times m}$ and $b \in R^m$ are the parameters that need to be learned, m is the number of neurons in the hidden layer, n is the dimension of the input vector, "⊗" is the element-wise multiplication operator, $\sigma$is the activation function Use the forget gate f^t and the input gate i^t to control the influence of the previous state and the influence of the candidate state, thereby obtaining the final memory state c^t, and the memory state c^t and the output gate o^t together determine the output $h^t$of the hidden layer.

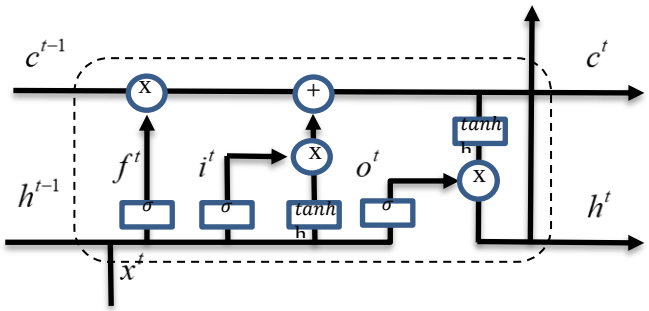


Fig.3. The internal situation of an LSTM "neuron". Each neuron has three inputs and three outputs, x^t is the newly added information at the moment, h^{t-1}, c^{t-1} is the previous information.

3. CNN-BLSTM Model Architecture

For the problem of multi-variable nonlinear time series forecasting, this paper proposed a multi-variable input multi-step output forecast model named CNN-BLSTM. The specific structure of the proposed model is shown in Figure 4. The model mainly contains two parts: a spatial feature extractor composed of CNN and a temporal feature encoder composed of Bi-LSTM. After applying CNN on the original input sequence to extract the spatial features and discriminative features of the multivariate input data, the results are input into Bi-LSTM to encode the temporal characteristics. In addition, after CNN and Bi-LSTM, dropout layer and batch-nonmalization layer were added to prevent overfitting and maintain the stability of the model. Finally, the output of Bi-LSTM is processed and predicted by two fully connected layers stacked together.

The following part mainly introduces the two main sub-modules that constitute the CNN-BLSTM model: spatial feature extractor and temporal feature encoder.

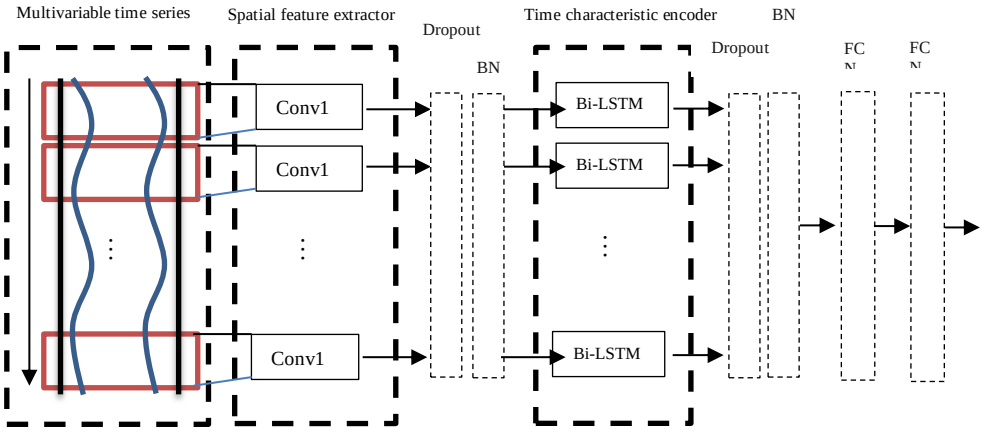


Fig.4. CNN-BLSTM model structure

3.1 Spatial Feature Extractor: CNN

Generally speaking, the data in real life is mostly multi-variable and there are very complex relationships between variables. Bi-LSTM does not have the ability to learn such complex relationships. Therefore, this article introduces CNN to extract spatial features of multivariate data before using Bi-LSTM

for temporal feature coding. One-dimensional convolution can extract two-dimensional features from multivariate time series data[36]. As shown in Figure 5, the CNN used in this article is composed of a convolutional layer and a pooling layer, directly processing the input original multivariable time series. Suppose $x_i = [x_i^1, x_i^2, \ldots, x_i^l]$ is the i-th dimension feature vector, $x_i^t \in R$ represents the data at time step t, and l is the length of the data. The convolutional layer slides the filter over the entire input sequence to generate feature maps. Each feature map can be regarded as the convolution activation of the corresponding filter on the entire sequence. It is assumed here that the convolutional layer uses k filters with a window size of m. The convolution operation is defined as a multiplication operation between a filter vector $u \in R^m$and a concatenation vector representation $x_{i:i+m-1}$ given by:

$$x_{i:i+m-1} = x_i \oplus x_{i+1} \oplus \cdots \oplus x_{i+m-1} \tag{4}$$

where $x_{i:i+m-1}$ represents a window of m continuous time steps starting from the i-th time step. In addition, a bias term b is also added into the convolution operation, so that the final operation is given as:

$$a_i = g(u^T x_{i:i+m-1} + b) \tag{5}$$

Where $*^T$ denotes the transpose of a matrix $*$ and g is a non-linear activation function that is set to Rectified Linear Units (ReLu) in our model[37].

Each vector $u$can be regarded as a filter, and the single value a_i can be regarded as the activation of the window. Then, the pooling layer is used to reduce the feature vector output by the convolutional layer, simplify the computational complexity of the network, and at the same time compress the generated feature map to generate salient features. The convolution operation of the entire sequence is realized by moving the filtering window from the start time step to the end time step. At this time, the feature map is a vector, expressed as:

$$a_j = [a_1, a_2, \cdots, a_{l-m+1}] \tag{6}$$

where the index j denotes the j-th filter. The pooling layer can reduce the length of feature maps, thereby further reducing the number of model parameters. The hyperparameter of the pooling layer is the pooling length, denoted as s. The max operation is taking a max over the s consecutive values in feature map a_j. Then, the compressed feature vector can be obtained as:

$$h = [h_1, h_2, \cdots h_{\frac{l-m}{s}+1}] \tag{7}$$

where $h_j = max(a_{(j-1)s}, a_{(j-1)s+1}, \cdots, a_{js-1})$. Usually multiple filters with different initial weights are used to derive the output of the CNN layer.

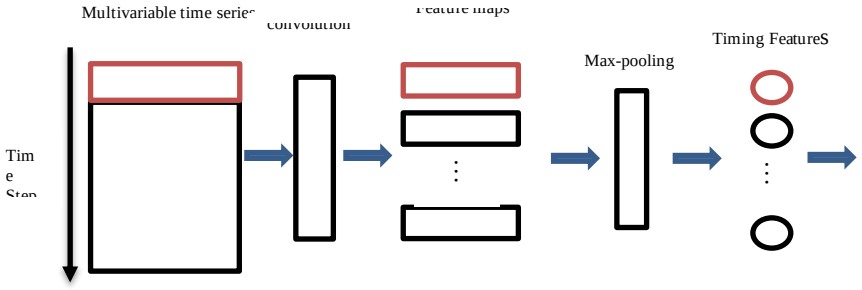


Fig.5. Spatial feature extractor

3.2 Temporal Feature Encoder: Bi-Directional LSTM

Considering that in the time series forecasting problem, the data generated in the near future has an important influence on the future forecast. This article used two-way LSTM to process temporal feature coding. The gating unit of the bidirectional LSTM is the same as the LSTM. It is a combination of the forward LSTM and the backward LSTM. The forward LSTM captures the characteristic information of the past data, and the backward LSTM captures the characteristic information of the currently generated data. Then the captured past feature information and current feature information are fused to obtain global data information, that is, a data step is the result of simultaneous input of forward and reverse (past and current) information. See Figure 6 for a bidirectional LSTM network. The following formula is the corresponding hidden layer function, → and ← represent the forward and reverse processes respectively.

$$\vec{i}^t = \sigma\left(\vec{W}^i \vec{x}^t + \vec{V}^i \vec{h}^{t-1} + \vec{b}^i\right)$$

$$\vec{f}^t = \sigma\left(\vec{W}^f \vec{x}^t + \vec{V}^f \vec{h}^{t-1} + \vec{b}^f\right)$$

$$\vec{o}^t = \sigma\left(\vec{W}^o\vec{x}^t + \vec{V}^o\vec{h}^{t-1} + \vec{b}^o\right) \tag{8}$$

$$\vec{c}^t = \vec{f}^t \odot \vec{c}^{t-1} + \vec{i}^t \odot tanh(\vec{W}^c\vec{x}^t + \vec{V}^c\vec{h}^{t-1} + \vec{b}^c)$$

$$\vec{h}^t = \vec{o}^t \odot tanh(\vec{c}^t)$$

$$\overleftarrow{i}^t = \sigma(\overleftarrow{W}^i\overleftarrow{x}^t + \overleftarrow{V}^i\overleftarrow{h}^{t+1} + \overleftarrow{b}^i)$$

$$\overleftarrow{f}^t = \sigma(\overleftarrow{W}^f\overleftarrow{x}^t + \overleftarrow{V}^f\overleftarrow{h}^{t+1} + \overleftarrow{b}^f)$$

$$\overleftarrow{o}^t = \sigma\left(\overleftarrow{W}^o\overleftarrow{x}^t + \overleftarrow{V}^o\overleftarrow{h}^{t+1} + \overleftarrow{b}^o\right) \tag{9}$$

$$\overleftarrow{c}^t = \overleftarrow{f}^t \odot \overleftarrow{c}^{t+1} + \overleftarrow{i}^t \odot tanh(\overleftarrow{W}^c\overleftarrow{x}^t + \overleftarrow{V}^c\overleftarrow{h}^{t+1} + \overleftarrow{b}^c)$$

$$\overleftarrow{h}^t = \overleftarrow{o}^t \odot tanh(\overleftarrow{c}^t)$$

where model parameters including all W, V and b are shared by all time steps and learned during model training, σ is the sigmoid activation function, $\odot$ denotes the element-wise product and k is a hyper-parameter that represents the dimensionality of hidden vectors.

The complete BLSTM hidden element representation h^t is the concatenated vector of the outputs of forward and backward processes as follows:

$$h^t = \vec{h}^t \oplus \overleftarrow{h}^t. \tag{10}$$

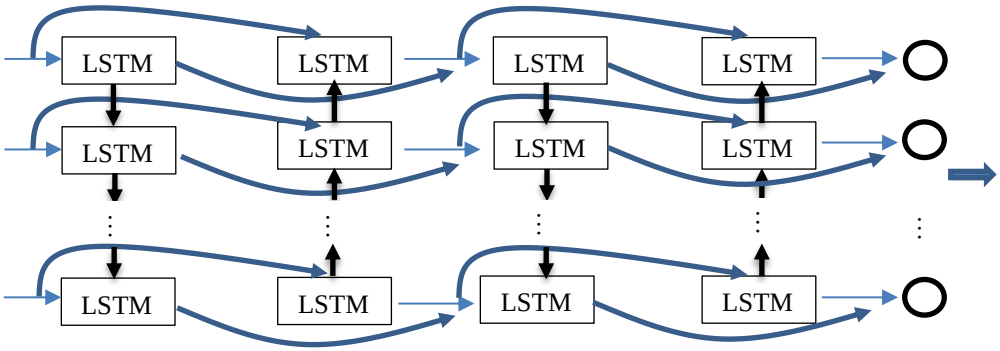


Fig.6. Stacked Bidirectional LSTMs

3.3 Dropout and Batch-Normalization Layers

After CNN and Bi-LSTM, add dropout layer batch-normalization layers layer. Among them, the dropout layer reduces the correlation between neurons to prevent overfitting[38]. Figure 7 shows the sparse network generated by applying dropout to a standard neural network, and some of the crossover units have been discarded.

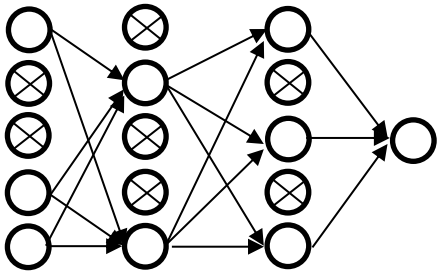


Fig.7. After applying dropout

The proposed model has a multilayer network structure. During the training process, the model parameters are constantly changing, leading to continuous changes in the input distribution of subsequent layers. The learning process must adapt each layer to the new input distribution, so the learning rate must be reduced, resulting in slow model convergence. In order to accelerate the convergence of the network and improve the stability and learning accuracy of the network, the proposed model adds batch-normalization layers[39] after the dropout layer. Batch-normalization makes networks robust to bad initialization of weights; reduces covariance shift by normalizing and scaling inputs and scale and shift parameters to avoid losing stability of the network. The output of the Batch-normalization layer can be calculated as:

$$y_i = \gamma x_i^{'} + \beta$$

$$x_i^{'} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$
$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$
$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \quad (11)$$

Where μ_B is the mean of the input x_i, σ_B^2 is the variance of x_i, ε is a small constant, γ and β are the parameters to be learned.

Finally, two fully connected dense layers are stacked together, the output of the previous fully connected layer is used as the input of the next connected layer, and the prediction result is obtained from the latter connected layer. The specific calculation is:

$$o^i = g(W_i h^i + b_i) \quad (12)$$

where o^i and h^i denote the output and input of the i-th fully-connected layer, respectively. W_i and b_i represent the transformation matrix and the bias term in the i-th fully-connected layer, respectively. The function $g()$ is also set to be ReLu.

4. Experimental Analysis

4.1 Dataset

The data set for model verification in this paper is a meteorological data set collected by a provincial expressway monitoring station, including six meteorological factors including visibility, temperature, humidity, precipitation, instantaneous wind speed, and road surface temperature. These data are recorded every 10 minutes, and one year of data is used for the experiment, the total number of samples is 52,560. Use the proposed model to make a multi-dimensional input-single output forecast of the temperature, the entire historical data is divided into three parts: training set, validation set, and test set. The training set is 70% (36792), the validation set is 20% (10512), and the test set is 10% (5256).

4.2 Parameter Setting and Evaluation Index

The proposed model uses a total of 36 sets of historical data in the first 6 hours to forecast the temperature at 36 points in the next 6 hours, that is, the input of the model is (36, 6) and the output is (36, 1). The TensorFlow framework[40] is used to implement the CNN-BLSTM model. Considering that the neural network is used to build the model, the size of the network layer and the number of neurons are not strictly defined. After many experiments and adjustments, we finally determined the parameters of the model. Among them, the number of convolution kernels of the convolution layer is set to 64, the largest pool is selected in the pool layer, and the activation function is Relu; the number of neurons in the two layers of Bi-LSTM is set to 200; The number of neurons in the fully connected layer is set to 100 and 1, respectively, and its nonlinear activation function is set to ReLu. The specific parameter settings of the model are shown in Table 1.

Tab. 1. Specific parameters of the model

CNN	Convolution Max-pooling	filters =64; kernel_size=3; stride=1 kernel_size=2; stride=2	Epoch= 100 Batch-size=600
BLSTM	Units1 Units2	Units1= 200 Units2=200	
Dropout	dropout = 0.1		
Fully connected	neurons = 100 neurons = 1		

In order to quantitatively evaluate the performance of the proposed CNN-BLSTM model, combined with the actual needs of the application, in this paper, Mean Absolute Error (MAE), Mean Relative Error (MRE) and Root Mean Squared Error (RMSE) are used as evaluation indicators. MAE is the mean value of the absolute value of all real and forecast errors. Compared with the average error, the deviation will not be offset due to the absolute value. It can better represent the overall error margin. The smaller the value, the better the forecast effect. The specific calculation formula is:

$$E_{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - p_i| \quad (13)$$

MRE is the mean value of the absolute value of all real and forecast errors divided by the real value. It is a relative quantity used to measure deviation. The specific calculation formula is:

$$E_{MRE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - p_i}{y_i}\right| \times 100\% \quad (14)$$

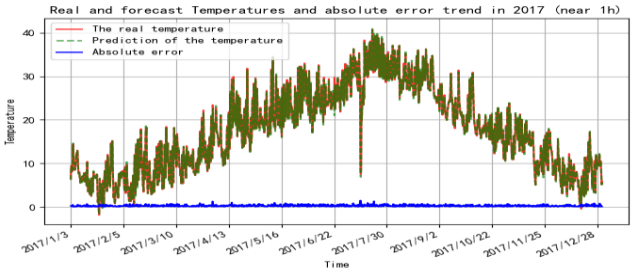
RMSE is the square root of the mean of all real values and the square of the error squares of the forecast value, and is used to measure the deviation between the forecast value and the real value. The specific calculation formula is:

$$E_{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2} \quad (15)$$

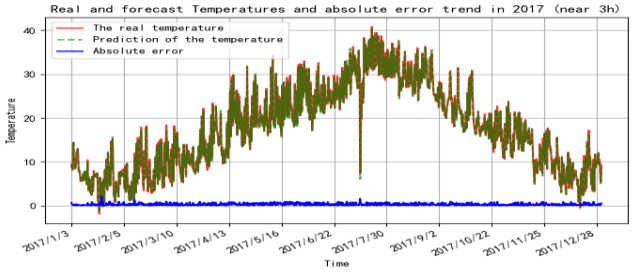
where n is the number of test samples, y_i is the true value, and p_i is the forecast value.

4.3 Experimental Results and Analysis

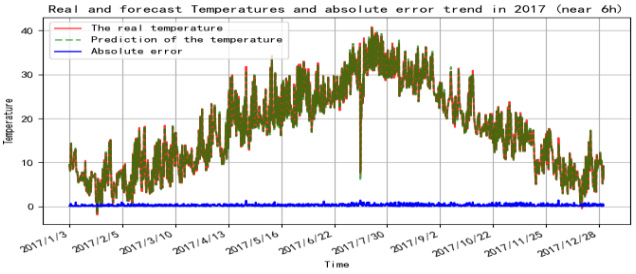
Figure 8 depicts the true temperature, forecast temperature and the error between the trend curves of the 1h, 3h and 6h monitoring stations in 2017. The red curve in the figure represents the true temperature, the green curve represents the temperature predicted by the CNN-BLSTM model, and the blue curve represents the absolute error between the true temperature and the predicted temperature. It can be seen from the figure that at each time of 1h, 3h and 6h of nowcasting, the forecasted temperature is consistent with the real temperature. In addition, the error curve between the predicted temperature and the true temperature changes smoothly, with only slight fluctuations around the x-axis, and there is no accelerated decline in forecast accuracy as the forecast time is delayed. However, in the forecast between June and July, there may be large fluctuations in the error at a certain moment, which may be caused by sudden weather changes. It can be seen that the use of the CNN-BLSTM model for temperature forecasting in the next 6 hours has very high accuracy and very stable forecasting performance.



(a) Real and forecast temperature and absolute error trend near 1h.



(b) Real and forecast temperature and absolute error trend near 3h.



(c) Real and forecast temperature and absolute error trend near 6h.

Fig.8. The trend curve of the true temperature, the forecast temperature and the error between 1h, 3h and 6h in 2017

4.4 Comparative Experiment

In order to verify the performance of the proposed CNN-BLSTM model, three benchmark models (LSTM, CNN, CNN-LSTM) are selected for comparison with the evaluation indicators. The parameter settings of each model are the same as the model settings proposed in this article.

Tab.2. Comparison results of different model forecasts

Models	MAE	MRE	RMSE
LSTM	0.627	0.375	1.352
CNN	0.583	0.126	1.359
CNN-LSTM	0.375	0.055	0.893
CNN-BLSTM	0.108	0.009	0.483

Table 2 shows the prediction results of different models. It can be seen from the table that the mixed models CNN-LSTM and CNN-BLSTM are better than the single model LSTM and CNN. It is not feasible to use a single LSTM model to predict non-stationary multivariate time series data. The MAE value of the CNN-LSTM model in the hybrid model is 0.357, the MRE value is 0.055, and the RMSE value is 0.893. For the two single models, compared with the CNN-LSTM model, the CNN model with better prediction results has a 55% increase in MAE value, a 129% increase in MRE value, and a 52% increase in RMSE value. Compared with the MAE value of the CNN-LSTM model, the MAE value of the proposed CNN-BLSTM model is reduced by 44%, the MRE is reduced by 84%, and the RMSE value is reduced by 35%. In summary, it can be concluded that the proposed CNN-BLSTM model is better than other comparative models to find out the changing law of the forecast target.

5 Conclusion

In this paper, a deep neural network prediction model based on CNN and Bi-LSTM is proposed. In order to verify the performance of the model, temperature prediction experiments are carried out on a real meteorological data set of a province, and compared with three popular models. Experimental results show that the proposed CNN-BLSTM model can effectively mine the features of multivariate non-stationary time series data. The current model has a large error in the prediction of sudden changes in temperature. Future exploration based on the error relearning model, by integrating the error characteristics of the previous time period into the model, the model is corrected to facilitate the establishment of a more advanced and reliable forecast model.

References

[1] Soares, E., Costa, P., Costa, B., et al.: Ensemble of evolving data clouds and fuzzy models for weather time series prediction. Appl. Soft Comput. (2017). S1568494617307573.

[2] Kim T Y , Cho S B . Predicting the Household Power Consumption Using CNN-LSTM Hybrid Networks[J]. 2018.

[3] Yang, Y., Guizhong, L.: Multivariate time series prediction based on neural networks applied to stock market. In: 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236), vol. 4, IEEE.

[4] Zhang, G.P.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing 50, 159–175 (2003).

[5] K. Kumar, V.K. Jain, Autoregressive integrated moving averages (ARIMA) modelling of a traffic noise time series, Applied Acoustics 58 (3) (1999) 283–294.

[6] V.S. Ediger, S. Akar, ARIMA forecasting of primary energy demand by fuel in Turkey, Energy Policy 35 (3) (2007) 1701–1708.

[7] Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting, pp. 35–62.

[8] Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, pp. 159–175.

[9] Lai, G., Chang, W. C., Yang, Y., & Liu, H. (2018). Modeling long- and short-term temporal patterns with deep neural networks. SIGIR, pp. 95–104.

[10] Qin, Y., Song, D., Cheng, H., Cheng,W. Jiang, G., & Cottrell, G.W. (2017). A dual-stage attention-based recurrent

neural network for time series prediction. In IJCAI'17, pp. 2627–2633. http://dl.acm.org/citation.cfmid=3172077.3172254.

[11] Popescu, I., Nikitopoulos, D., Constantinou, P., Nafornita, I.: ANN prediction models for outdoor environment. In: 2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5. IEEE (2006).

[12] Sapankevych, N.I., Sankar, R.: Time series prediction using support vector machines: a survey. IEEE Comput. Intell. Mag. 4(2), 24–38 (2009).

[13] Zuo R, Xiong Y, Wang J, Carranza EJM (2019) Deep learning and its application in geochemical mapping. Earth Sci Rev 192:1–14. https://doi.org/10.1016/j.earscirev.2019.02.023.

[14] Fang W, Zhong B, Zhao N, Love PE, Luo H, Xue J, Xu S (2019) A deep learning-based approach for mitigating falls from height with computer vision: convolutional neural network. Adv Eng Inf 39:170–177.

[15] Affonso C, Rossi ALD, Vieira FHA, de Leon Ferreira ACP (2017) Deep learning for biological image classification. Expert Syst Appl 85:114–122.

[16] Fan L, Zhang T, Zhao X, Wang H, Zheng M (2019) Deep topology network: a framework based on feedback adjustment learning rate for image classification. Adv Eng Inf 42:100935.

[17] Plappert M, Mandery C, Asfour T (2018) Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks. Rob Auton Syst 109:13–26.

[18] Shin HC, Lu L, Summers RM (2017) Natural language processing for large-scale medical image analysis using deep learning. In: Zhou SK, Greenspan H, Shen D (eds) Deep learning for medical image analysis. Academic Press, Cambridge, pp 405–421.

[19] HSU D. Time Series Forecasting Based on Augmented Long Short-Term Memory[J]. 2017.

[20] DONAHUE J, HENDRICKS L A, GUADARRAMA S, et al. Long-term recurrent convolutional networks for visual recognition and description[C]// 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015.

[21] ECK DOUGLAS, GRAVES A, SCHMIDHUBER JUERGEN. A New Approach to Continuous Speech Recognition Using {LSTM} Recurrent Neural Networks[J]. dynamic network, 2003.

[22] BERINGER N. Human language acquisition methods in a machine learning task[C]// International Conference on Interspeech -icslp. DBLP, 2004.

[23] Elsheikh, Ahmed, Yacout, Soumaya, et al.: Bidirectional handshaking LSTM for remaining useful life prediction. Neurocomputing 323, 148–156 (2019).

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems 25, (2012), pp. 1097–1105.

[26] Z. Wang, W. Yan, and T. Oates, Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline, CoRR, abs/1611.06455 (2016).

[27] Kim TY, Cho SB (2019) Predicting residential energy consumption using CNN-LSTM neural networks. Energy 182:72–81.

[28] Huang M, Tian D, Liu H, Zhang C, Yi X, Cai J, Ruan J, Zhang T, Kong S, Ying G (2018) A hybrid fuzzy wavelet neural network model with self-adapted fuzzy-means clustering and genetic algorithm for water quality prediction in rivers. Complexity. https://doi.org/10.1155/2018/8241342.

[29] Li P, Abdel-Aty M, Yuan J (2020) Real-time crash risk prediction on arterials based on LSTM-CNN. Accid Anal Prev 135:105371. https://doi.org/10.1016/j.aap.2019.105371.

[30] Barzegar R , Aalami M T , Adamowski J . Short-term water quality variable prediction using a hybrid CNN–LSTM deep learning model[J]. Stochastic Environmental Research and Risk Assessment, 2020, 34(8):1-19.

[31] Le Cun, B.B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Handwritten digit recognition with a back-propagation network. In Advances in Neural Information Processing Systems; Denver, CO, USA, 26–29 November 1990.

[32] Borovykh A , Bohte S , Oosterlee C W . Conditional Time Series Forecasting with Convolutional Neural Networks[J]. arXiv, 2017.

[33] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555, (2014).

[34] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.

[35] HOCHREITER, SEPP, SCHMIDHUBER, et al. Long short-term memory[J]. Neural Computation, 1997.

[36] Suzuki, J., Isozaki, H., Maeda, E.: Convolution kernels with feature selection for natural language processing tasks. In: Meeting of the Association for Computational Linguistics (2004).

[37] Nair, V.; Hinton, G.E. Rectifified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.

[38] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting, vol 15.

[39] Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift.

[40] MARTIN ABADI, ASHISH AGARWAL, PAUL BARHAM, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. Eprint Arxiv, 2015.