Applied Intelligence
https://doi.org/10.1007/s10489-023-04980-z

TCLN: A Transformer-based Conv-LSTM network for multivariate time series forecasting

Shusen Ma[1] · Tianhao Zhang[2] · Yun-Bo Zhao[1,2,3] · Yu Kang[1,2,3] · Peng Bai[2]

Accepted: 24 August 2023

Abstract
The study of multivariate time series forecasting (MTSF) problems has high significance in many areas, such as industrial forecasting and traffic flow forecasting. Traditional forecasting models pay more attention to the temporal features of variables and lack depth in extracting spatial and spatiotemporal features between variables. In this paper, a novel model based on the Transformer, convolutional neural network (CNN), and long short-term memory (LSTM) network is proposed to address the issues. The model first extracts the spatial feature vectors through the proposed Multi-kernel CNN. Then it fully extracts the temporal information by the Encoder layer that consists of the Transformer encoder layer and the LSTM network, which can also obtain the potential spatiotemporal correlation. To extract more feature information, we stack multiple Encoder layers. Finally, the output is decoded by the Decoder layer composed of the ReLU activation function and the Linear layer. To further improve the model's robustness, we also integrate an autoregressive model. In model evaluation, the proposed model achieves significant performance improvements over the current benchmark methods for MTSF tasks on four datasets. Further experiments demonstrate that the model can be used for long-horizon forecasting and achieve satisfactory results on the yield forecasting of test items (our private dataset, TIOB).

Keywords MTSF · Transformer · CNN · LSTM · Autoregressive model

1 Introduction

The application of multivariate time series forecasting (MTSF) is common in real life, such as stock price forecasting, weather forecasting, traffic flow forecasting, and so on [1–4]. It can help us take measures in advance and avoid possible risks [5]. For example, we can throw stocks just before they may fall and avoid significant economic loss. Traditional forecasting models include the vector autoregression (VAR) model, autoregressive moving average (ARMA) model, support vector regression (SVR) model, and so on. Though they can be applied to kinds of tasks [6–8], they are mainly suitable for univariate time series and cannot deal with complex nonlinear relationships of multivariate time series. In most real-world scenarios, multiple variables often exist at the same time, which makes the traditional models no longer applicable.

With the advent and development of deep learning, the strong nonlinear fitting ability of neural network models makes them able to effectively process MTSF tasks, such as the stock price prediction [9]. This work also takes full advantage of various neural networks to improve the forecasting accuracy of our goal. The focus of the MTSF tasks is to fully extract the temporal, spatial, and spatiotemporal correlations of the multivariate time series. They contain complex relationships of variables and can be used to forecast the trends of the target variable. The correlations are interpreted in Fig. 1.

The traditional recurrent neural network (RNN) can establish connections between hidden units, allowing it to store recent information. Therefore, RNN can effectively extract temporal dependence from stored historical information. However, RNN has difficulties capturing long-term dependence due to the gradient disappearing or exploding [10]. Long short-term memory (LSTM) network, a variant of

✉ Yun-Bo Zhao
ybzhao@ustc.edu.cn

1 Institute of Advanced Technology, USTC, Shushan District, Hefei 230031, Anhui, China

2 Department of Automation, USTC, Shushan District, Hefei 230031, Anhui, China

3 Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Shushan District, Hefei 230031, Anhui, China

Published online: 02 October 2023

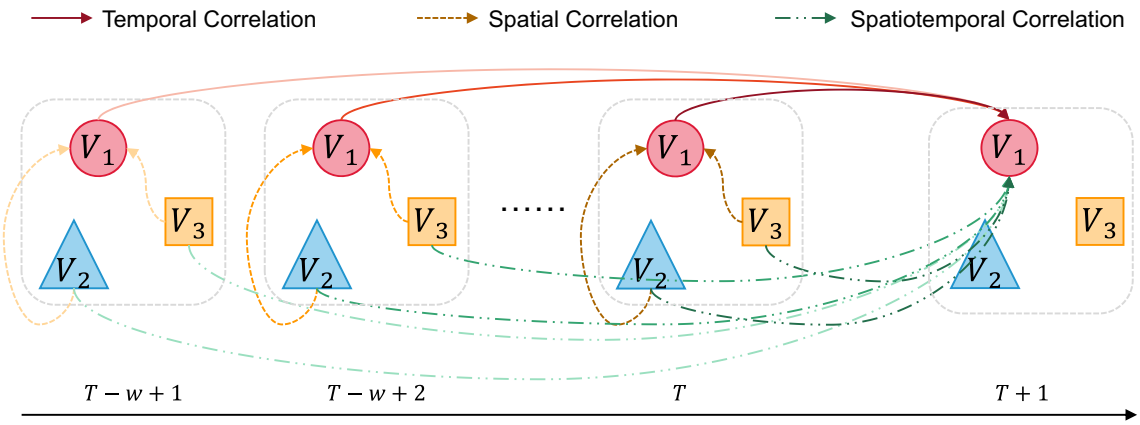


Fig. 1 Correlations of the multivariate time series. V_1 represents the target variable, V_2 and V_3 represent the covariates. The temporal correlation refers to that the value of V_1 at $T+1$ moment may be affected by the values of V_1 of the previous w moments. The spatial correlation refers to the latent relevance between V_1 and other covariates at every moment. And the spatiotemporal correlation refers to that the value of V_1 at $T+1$ moment may have an indirect relevance with the values of other covariates of the previous w moments

RNN, effectively alleviates the problem by introducing gating mechanisms and improves the ability to process long-time series input. However, it still cannot handle the input longer than a certain length. To extract long-term dependence of time series, the models based on attention mechanism [11–13] have been proposed, which improve the forecasting accuracy to a certain extent. However, their ability to extract spatial features is weak, which can be solved by an enhanced convolutional neural network (CNN) layer, such as the proposed Multi-kernel CNN layer in this paper.

With the coming of the Transformer [14] based on the self-attention mechanism, some studies [15, 16] apply canonical Transformer to MTSF tasks, which can improve the ability to process long time series. However, compared with the LSTM network, the encoder layer of the canonical Transformer is still insufficient in extracting temporal features though it has designed the Positional Encoding layer. Therefore, the combination of the Transformer and the LSTM network can exploit both strengths to enhance the performance of the model.

Simply put, the major challenge of MTSF tasks lies in adequately capturing the complex and long-term nonlinear relationship of multivariate time series, such as temporal, spatial, and spatiotemporal dependence. Motivated by the above, in this paper, an enhanced Transformer-based Conv-LSTM network (TCLN) is proposed for MTSF tasks. The main contributions of this paper are as follows:

- A new model named TCLN is proposed for MTSF tasks, which can outperform the current MTSF methods on four datasets, covering economy, climate, energy, and industrial product yield.
- The proposed Multi-kernel CNN module has a deeper receptive field than the traditional convolutional layer and can extract more spatial information between variables, while the traditional convolutional layer can only extract limited spatial features.
- The Encoder of TCLN combines the LSTM network with the self-attention mechanism, which can perform well in capturing long-term temporal information. The self-attention mechanism can handle long input series and the LSTM network can effectively obtain temporal information from the perspective of input order.

The remainder of this paper is structured as follows. Section 2 briefly analyzes related work about MTSF. Section 3 describes the TCLN in detail. And Section 4 illustrates the evaluation results of the TCLN in comparison with several baselines on four datasets. An ablation study is carried out to demonstrate the reasonableness of each module. Additionally, further experiments are conducted to show whether the TCLN can be applied to industrial data forecasting and achieve a good performance. In Section 5, some conclusions are drawn and the deficiency of the model and the future research directions are also discussed.

2 Related work

The classical time series forecasting models, such as VAR, ARMA, and SVR, are widely applied to various tasks [6–8]. Though they can extract the linear characteristics of variables, they are mainly suitable for univariate cases and cannot handle potential nonlinear relationships between multivariate time series. In addition, some models based on statistical machine learning are also applied to scenario prediction and planning [17, 18]. Fu et al. introduce statistical machine learning techniques for probabilistic power flow

calculation based on multiple scenarios, which are applied to the stochastic planning of distribution networks [18]. With the development of deep learning, deep neural network (DNN) has been widely used in time series forecasting problems [2, 3, 9, 19, 20]. Due to the strong nonlinear fitting characteristics of DNN, DNN-based models can extract the nonlinear features between variables well. Therefore, compared with the classical models, the tasks of MTSF can be completed more effectively by DNN-based models.

Deep learning models commonly used for MTSF tasks include RNN, variants of RNN (LSTM and Gated Recurrent Unit (GRU)) [21, 22], and hybrid models [23], combining traditional methods with DNN models. However, due to the problem of gradient vanishing or explosion in RNN-based models, they cannot handle long-time series effectively. To solve this problem, Qin et al. [12] propose a two-stage recurrent neural network (DA-RNN) based on the attention mechanism. The model successfully captures the long-term temporal dependence of time series by introducing the attention mechanism. Besides, a two-stage attention mechanism is adopted to avoid irrelevant features being assigned a higher attention weight. However, the DA-RNN model mainly focuses on the temporal correlation of time series and ignores the spatial correlation and the spatiotemporal correlation between variables. Considering the combination of CNN and LSTM (Conv-LSTM) network may solve the issues, Xiao et al. [11] propose a dual-stage attention-based Conv-LSTM network (DA-Conv-LSTM) based on the DA-RNN model. The model first processes the input data into the form of an image then extracts the spatial features of the input data through the convolutional layer, and finally uses the attention layer and LSTM layer to extract the temporal information of the variables. Fu et al. [13] propose a temporal self-attention-based Conv-LSTM network (TS-Conv-LSTM) by improving the structure of the DA-Conv-LSTM. However, the ability of the model to extract spatial correlation becomes weak as the spatial span of the series increases. Lai et al. [24] propose the Long- and Short-term Time-series network (LSTNet) model, which makes full use of CNN and recurrent layers to extract the spatial correlation and long-term temporal dependence, separately. Additionally, an autoregressive (AR) module is added to improve the robustness of the model.

With the development of graph neural network (GNN) and Transformer technology [14], more and more studies have adopted them to improve the accuracy of MTSF tasks [15, 16, 25–29]. Wu et al. [29] extract the sparse image adjacency matrix through the graph structure learning layer, making the hidden connections between image nodes (variables) can be found. And the spatial dependence between variables can be extracted by the graph convolutional layer. However, GNN-based models may be more suitable for situations where there is a topological relationship between the variables or a distribution network [30]. For example, the variables of the traffic-flow forecasting task can be connected by graph structure according to the geographical position of the sensors used to collect the traffic-flow data [31]. Yang et al. [25] propose an enhanced transformer-based framework for MTSF tasks. To scale the calculation amount, explicit sparse multi-head attention is applied. The static covariates processing module is designed to capture the periodicity of the time series. Ren et al. [32] propose a Transformer-enhanced temporal convolution network (TE-TCN) that combines the transformer multi-head attention mechanism and GRU to capture long-term periodic patterns. Additionally, two parallel temporal convolution networks are utilized to address short-term periodic dependencies.

Although the above methods improve the classical MTSF models' ability to extract temporal and spatial features, they still struggle with at least one of the three problems: lacking in the extraction of spatial features of changing spatial span, ignoring the spatiotemporal features between times series and struggling with long-term feature extraction. The TCLN can solve the above problems simultaneously. The Multi-kernel CNN layer can effectively extract spatial features. Meanwhile, the Transformer encoder and LSTM network are combined to extract the long-term temporal features and finally obtain the spatiotemporal correlation of the time series.

3 Framework

This section first gives a formulaic representation of the MTSF problem, and then details the various components of TCLN (Fig. 2).

3.1 Problem formulation

Assume that the dataset can be expressed as $X_{\text{dataset}} = (X_1, X_2, \ldots, X_N)^T$, where N represents the length of the dataset. We use $X_t = (x_t^1, x_t^2, \ldots, x_t^M) \in R^M$ to represent all variables at time t, where M represents the number of the variables. And the target variable is expressed as $y_t = X_t^l$, where $l \in [1, M]$. The purpose of this paper is to predict the value of the target variable in the next different time steps (horizon = 1, 3, 6, 12, 24) by analyzing the time series of the past t moments. For example, when the input data is $X = (X_1, X_2, \ldots, X_t)^T$ and horizon is h, the forecasting value of the target variable will be $\hat{y}_{t+h}$. Based on the interpretation above, we can conclude our goal as follows:

$$\hat{y}_{t+h} = f\left(X_1, X_2, \ldots, X_t, \text{horizon} = h\right), \tag{1}$$

where $f(\cdot)$ is the mapping function that the TCLN needs to learn.

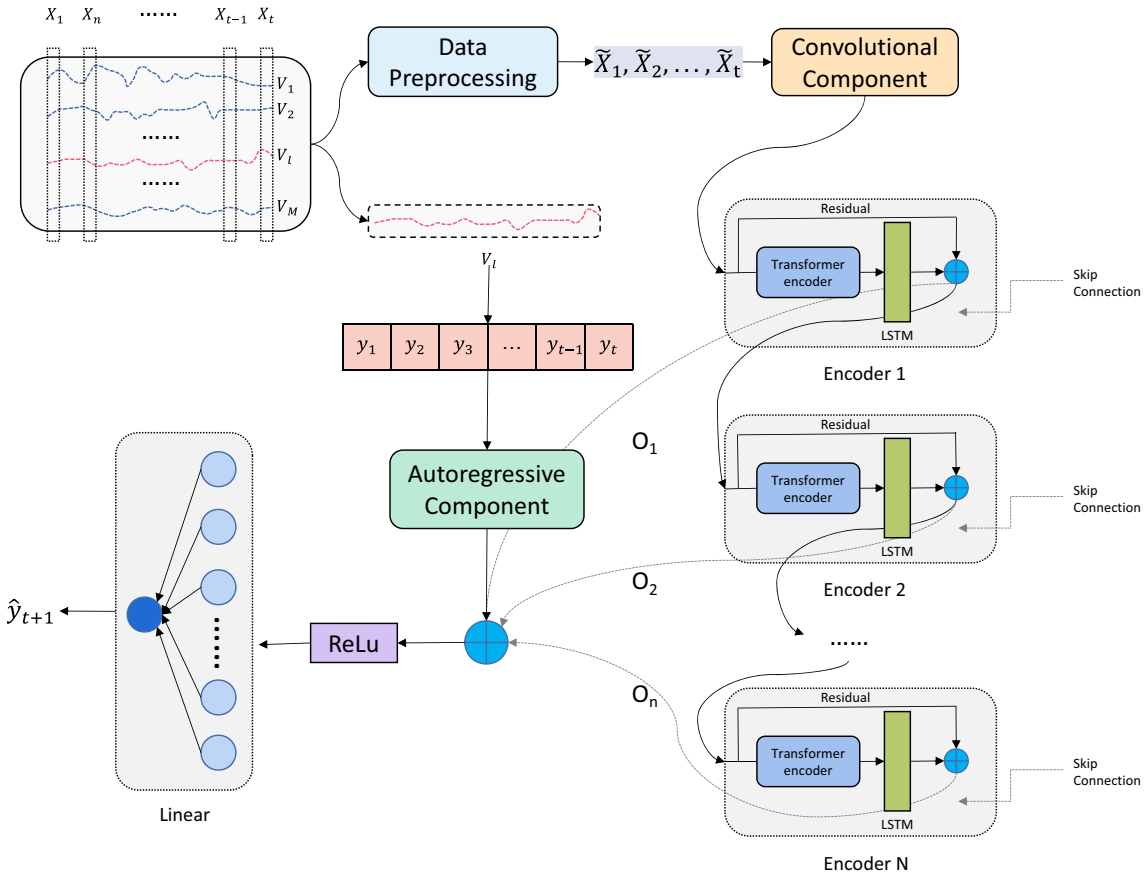


Fig. 2 The overall structure of the model

Considering that convolution operation can effectively extract the spatial features between variables, we process the data into the form of an image [11]. That is to say, the one-dimensional data at each moment is converted into two-dimensional data during data preprocessing. For example, we can convert $X_t = (x_t^1, x_t^2, \ldots, x_t^M) \in R^M$ into $\widetilde{X}_t \in R^{p \times p}$, where $p \times p \geq M$. And when $p \times p > M$, the remaining blank spaces are filled with 0. The conversion process is shown in Fig. 3 and the selection of p is shown in (2):

$$p = \begin{cases} \sqrt{M}, & \sqrt{M} \in \mathbb{N}^* \\ \lceil \sqrt{M} \rceil, & \sqrt{M} \notin \mathbb{N}^*. \end{cases} \tag{2}$$

3.2 Convolutional component

Compared with the single-kernel convolutional layer [11, 24], the receptive field of the multi-kernel convolutional layer is deeper [33], and it makes the extracted spatial features richer, which is beneficial for improving the prediction accuracy of TCLN. Therefore, we propose a module based on the Inception module, named Multi-kernel CNN. The module can make the dimension of the output of different layers consistent by configuring each layer's kernels and paddings. The output of all the convolutional layers can be added to get the final spatial feature vectors. And the structure of Multi-kernel CNN is shown in Fig. 4. When the input data is $\widetilde{X} = (\widetilde{X}_1, \widetilde{X}_2, \ldots, \widetilde{X}_{t_w})^T$, the overall input dimension can be expressed as (t_w, H, W), where t_w represents the length of the sliding window, H represents the height of $\widetilde{X}_1$ and W represents the width of $\widetilde{X}_1$. The output dimension of this convolutional component can be expressed as $(t_w, c, H-1, 1)$, where c represents the number of channels that is set to 1 in this paper. To simplify the expression, the redundant dimensions of the output are removed, that is to say, the dimension of the final output is $(t_w, H-1)$. To emphasize the temporal information, data of each time step is processed by separate Multi-kernel CNN.

3.3 Encoder component

Considering the good performance of Transformer [14] in handling the input of long text, we combine the encoder layer of the Transformer and the LSTM network to form the Encoder component to extract temporal features for long input sequences. The skip connection between Encoder modules is mainly to prevent degradation problems caused by too

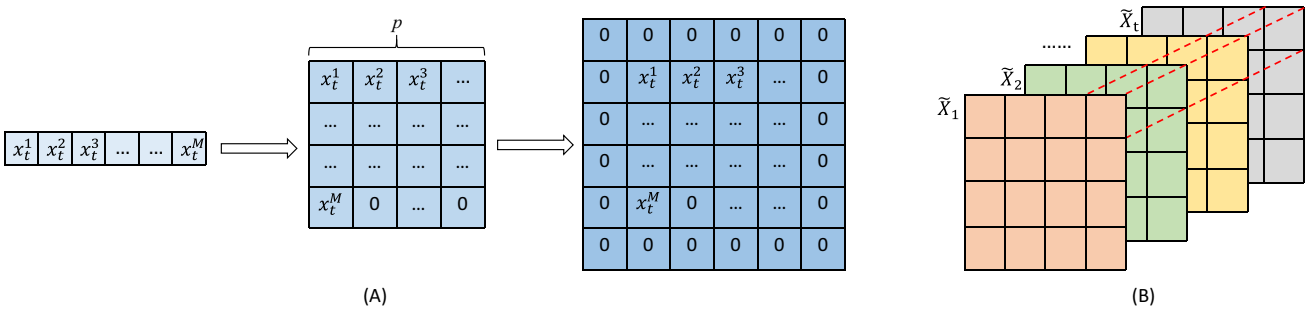


Fig. 3 Data preprocessing. (A) Describe how one-dimensional data is converted into two-dimensional data and how to pad zero value according to the padding size set to 1 in this figure. And the figures also show the spatial dependence between variables. (B) Describe the input data that we use to forecast the next different horizon's target variable. The figure also shows the spatiotemporal dependence between variables, and the red dotted lines represent the time correlation

deep a network. The residual connection in the module is to alleviate the problem of gradient disappearance. The details of the Encoder component are shown in Fig. 5.

3.3.1 Transformer encoder component

The Transformer encoder mainly consists of the Positional Encoding layer, the Dimension Transformation layer, the Multi-head Attention layer, and the Feed Forward layer. The Positional Encoding layer and the Dimension Transformation layer perform positional encoding and dimension transformation for the input data $\widetilde{X}^{t_w \times (H-1)}$, respectively. The purpose of positional encoding is to make the model remember the position information of the input data. And the dimension of the encoded data is (t_w, d_{model}). The purpose of dimension transformation is to map the low dimension into a high dimension and make the dimension of the input data consistent with that of the encoded data. And the transformation method used in this paper is linear mapping. Finally, sum the vectors encoded by the Positional Encoding layer and the vectors of the Dimensional Transformation layer. The obtained result, $\hat{X}^{t_w \times d_{\text{model}}}$, is used as the input of the Multi-head attention layer.

The Multi-head attention layer first maps $\hat{X}^{t_w \times d_{\text{model}}}$ into H independently learned $Q_h = \hat{X}^{t_w \times d_{\text{model}}} W_h^Q$, $K_h = \hat{X}^{t_w \times d_{\text{model}}} W_h^K$, $V_h = \hat{X}^{t_w \times d_{\text{model}}} W_h^V$, $h = 1, 2, \ldots, H$, through the linear transformation matrix W^Q, W^K, W^V. And then calculate the attention vector by the scaled dot-product attention algorithm. The algorithm formula is as follows:

$$\begin{aligned} \text{Context}_h &= \text{Attention}\,(Q_h, K_h, V_h) \\ &= \text{softmax}\left(\frac{Q_h {K_h}^T}{\sqrt{d_k}}\right) V_h, \end{aligned} \tag{3}$$

where W^Q, W^K, W^V are trainable parameters and $d_k = \frac{d_{\text{model}}}{H}$. The calculated H Contexts are spliced into a matrix by

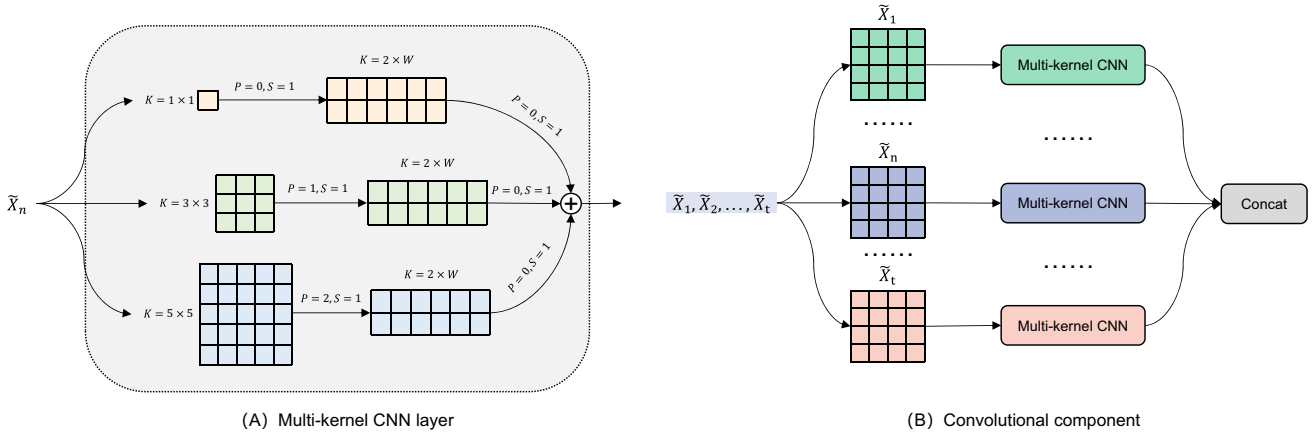


Fig. 4 (A) Multi-kernel CNN layer. S represents stride, P represents the padding size (padding value is 0), K represents the convolution kernel size, and W represents the width of the input. After processing the input by different convolutional layers, the output is three vectors of identical dimension, $(W - 1, 1)$. The final spatial feature vector can be obtained by adding the vectors. (B) Convolutional component. The results of all sublayers are concatenated to form a new feature vector. The dimension of the vector is $(t_w, W - 1)$

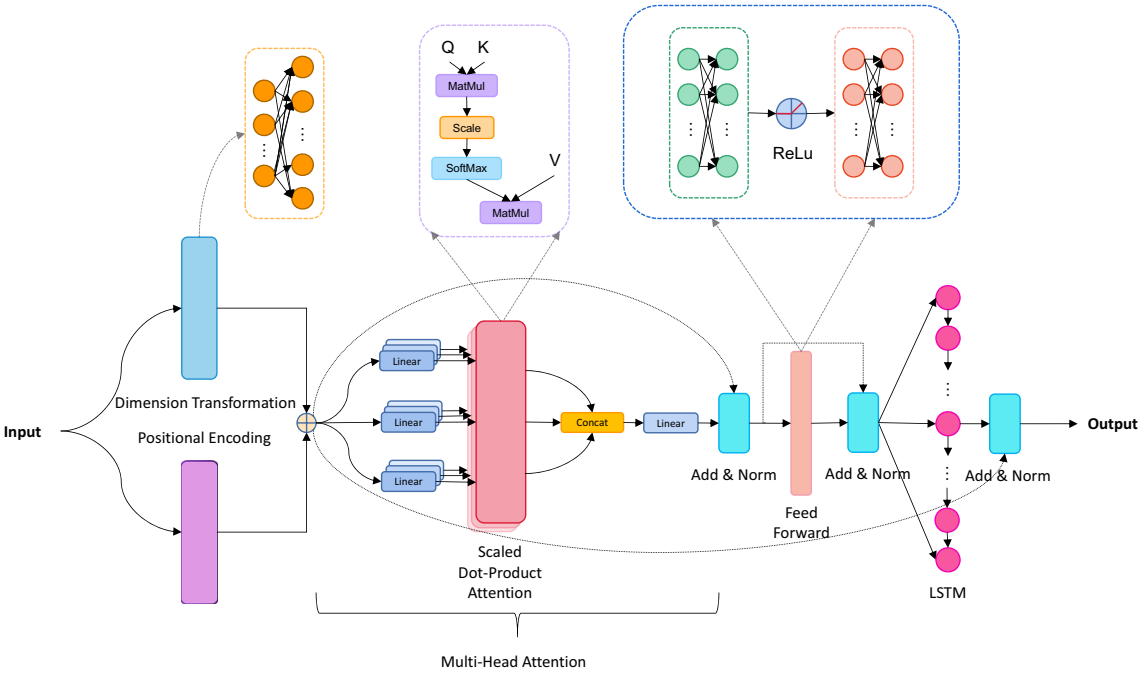


Fig. 5 Encoder structure. The input passes through the Positional Encoding layer and the Dimension Transformation layer, a linear layer shown in the orange dotted box. The output results of the two layers are added to get the input of the Multi-Head Attention, which uses the scaled dot-product attention algorithm shown in the purple dotted box to calculate the temporal dependencies. Then after the Add&Norm layer, there is a Feed Forward layer shown in the blue dotted box. Finally, the preceding result passes through two Add&Norm layers with an LSTM layer between them to obtain the output

the concatenation operation. Then the matrix is mapped into the final output vector, named X_{atten}, by multiplying linear matrix W_O. The dimension of X_{atten} is consistent with that of the initial input $\hat{X}^{t_w \times d_{\text{model}}}$. And the calculation formula is as follows:

$$X_{\text{atten}} = \text{Concat}\left(\text{Context}_1, \text{Context}_2, \cdots, \text{Context}_H\right) W_O, \tag{4}$$

where W_O is a trainable parameter.

In addition to the residual connection and standardized operation, the output of the multi-head attention layer, X_{atten}, also needs to go through the Feed Forward layer. The Feed Forward layer mainly performs a full connection operation on the input and then uses the ReLU activation function to calculate the output of the full connection. The relevant calculation formula is as follows:

$$FF = AN\left(FC_2\left(\text{ReLU}\left(FC_1\left(AN\left(X_{\text{atten}}\right)\right)\right)\right)\right) \tag{5}$$

$$\text{ReLU}\left(x\right) = \max\left(0, x\right), \tag{6}$$

where $AN\left(\cdot\right)$ represents the Add&Norm layer, $FC_1\left(\cdot\right)$ and $FC_2\left(\cdot\right)$ represent the full connection layers. Finally, FF passes through the LSTM layer and the Add&Norm layer to get the output of the Encoder layer. The calculation formula is as follows:

$$\text{Output} = AN\left(\text{LSTM}\left(FF\right)\right), \tag{7}$$

where the theory of LSTM$(\cdot)$ will be described in Section 3.3.2.

3.3.2 LSTM component

Although the Transformer encoder can obtain position information through the Positional Encoding layer, there might still be a certain information loss between words' positions. Compared with obtaining position information through functions, the LSTM network directly gets it from the perspective of the input order, which may enhance the model's perception of temporal information. The temporal information can be stored by the gating mechanism, making it more effective to obtain the temporal dependence of variables. Therefore, we add an LSTM network layer based on the Transformer encoder layer to jointly extract the temporal features.

The LSTM network structure mainly includes a forgetting gate, an input gate, and an output gate. The forgetting gate f_t is used to control the internal state c_{t-1} of the previous moment to forget how much information. The input gate i_t is

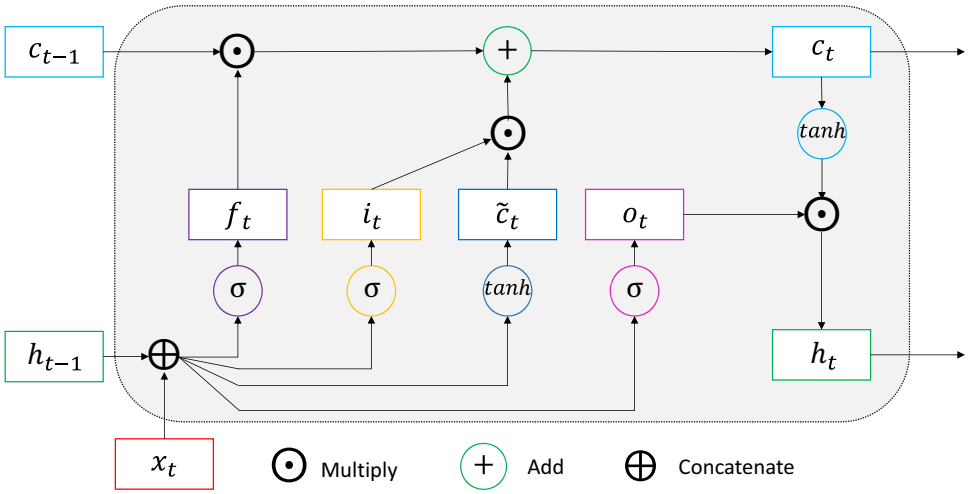


Fig. 6 LSTM network structure

used to control the candidate state $\widetilde{c}_t$ of the current moment to retain how much information. The output gate o_t is used to control the internal state c_t of the current moment to output how much information to the external state h_t. The structure of the LSTM network is shown in Fig. 6.

The forgetting gate, input gate, and output gate are expressed as follows:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{8}$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{9}$$
$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right). \tag{10}$$

The current candidate state $\widetilde{c}_t$, memory unit c_t, and external state h_t are calculated as follows:

$$\widetilde{c}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{11}$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \widetilde{c}_t \tag{12}$$
$$h_t = o_t \cdot \tanh\left(c_t\right). \tag{13}$$

$W_f, b_f, W_i, b_i, W_o, b_o, W_c, b_c$ are trainable parameters. $\sigma(\cdot)$ and $\tanh(\cdot)$ are all activation functions. The formulas are as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{14}$$
$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \tag{15}$$

3.4 Decoder component

This component firstly adds the output of the previous N encoder layers and the AR component and then decodes the sum through the ReLU function and the Linear layer to obtain the final forecasting result $\hat{y}_{t+1}$. The calculation formula for the decoder layer is as follows:

$$\hat{y}_{t+1} = \text{Linear}\left(\text{ReLU}\left(\Sigma\left(O_1, O_2, \cdots, O_n, O_{AR}\right)\right)\right), \tag{16}$$

where O_{AR} represents the output of the AR model, described in Section 3.5.

3.5 Autoregressive component

Due to the non-linear nature of the CNN layer, Transformer encoder, and LSTM network, the scale of the model output is not sensitive to that of the input. Therefore, we consider the final forecasting output of the TCLN as a mixture of a non-linear part and a linear part. In this paper, the linear part adopts the AR model. It can extract the linear correlation of the past series of the target variable to improve prediction accuracy. The AR model is formulated as follows:

$$\begin{aligned} O_{AR} = \widetilde{y}_{t+1} &= w_1 y_1 + w_2 y_2 + \cdots + w_t y_t + \varepsilon_{t+1} \\ &= \sum_{k=1}^{t} w_k y_k + \varepsilon_{t+1}, \end{aligned} \tag{17}$$

where w_k, ε_{t+1} are trainable parameters. w_k represents the weight of the variable, ε_{t+1} represents the random noise and y_k represents the past value of the target variable.

4 Evaluation

We mainly evaluate recent models and TCLN on four datasets for MTSF tasks and then show the performance of the TCLN in the long horizon forecasting task.

4.1 Methods for comparison

In this paper, the models in our comparative evaluation are as follows.

- CNN is the one-dimensional CNN model with a dense layer and one hidden unit.
- LSTM is the typical sequential model with a dense layer and one hidden unit.
- GRU is the improved version of the LSTM model with fewer parameters.
- StemGNN, based on the discrete Fourier transform, stands for the spectral temporal graph neural network.
- DA-RNN stands for the dual-stage attention-based recurrent neural network.
- TPA-LSTM stands for the model with a new temporal pattern attention mechanism.
- LSTNet stands for the Long- and Short-term time series forecasting model with a one-dimensional CNN and RNN to capture short-term and long-term dependence, respectively.
- DA-Conv-LSTM stands for the dual-stage attention-based Conv-LSTM network with two attention layers and a convolutional layer to capture the temporal and spatial dependence, respectively.
- TS-Conv-LSTM stands for the temporal self-attention-based Conv-LSTM network.

4.2 Metrics

In this paper, we use three conventional evaluation metrics (RMSE, MAE, and MAPE) to evaluate the model. The relevant definitions are as follows:

- Mean Absolute Error(MAE):

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{18}$$

- Root Mean Squard Error(RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(y_t - \hat{y}_t\right)^2} \tag{19}$$

- Mean Absolute Percentage Error(MAPE):

$$\mathrm{MAPE} = 100 \times \frac{1}{n}\sum_{t=1}^{n}|\frac{y_t - \hat{y}_t}{y_t}|, \tag{20}$$

where y_t and $\hat{y}_t$ represent ground truth and forecasting value, respectively. For the three metrics, a lower value is better.

4.3 Data

Three public datasets and one private dataset are adopted to verify the forecasting validity of the TCLN. The relevant information of the datasets is as follows:

- NASDAQ100: The NASDAQ100 dataset includes 82 stock price data from July 26, 2016 to December 22, 2016. Each day contains 390 data points, except for November 25 and December 22, which contain 210 and 180 data points, respectively.
- SML2010: The SML2010 dataset is collected from 22 sensors in a room for about 40 days, with an average sampling time of 15 minutes.
- Solar Energy: The production records of solar power in 2006, which are sampled every 10 minutes from 137 PV plants in Alabama State.
- TIOB: To verify the performance of the TCLN on the industrial forecasting tasks, we also adopt an industrial dataset, called TIOB, applied by a laptop manufacturing factory. The data is collected from the board-level functional test, including the detection results of eight test items. Each test item contains 9676 data points, whose values are 1 or 0. Considering the yield of test items is significant for the board-level functional test, we will forecast the yield of the test items. We utilize the method of sliding windows to calculate the yield, where the window size is 100 and the stride is 1. The yield of the test items is shown in Fig. 7.

The details of the four datasets in the experiment are shown in Table 1.

4.4 Experimental details

In this paper, all datasets are divided into the training set, validation set, and test set, where the division ratio is 8:1:1. The training set is used to train model parameters, the validation set is used to select the hyperparameters that are most suitable for the model, and the test set is used to evaluate the forecasting performance of the model. To make the experimental results more scientific and credible, several random seeds are set up to repeat the experiment, and the experimental results are averaged to reduce the experimental error.

To make the model obtain a better solution in a short time and stabilize in the later stage of the training, we use the method, named exponential decay of the learning rate, to train the model. The number of training iterations is set to 200 epochs. The calculation formula for the learning rate is as follows:

$$lr = lr_0 \times g^{\frac{i}{I}}, \tag{21}$$

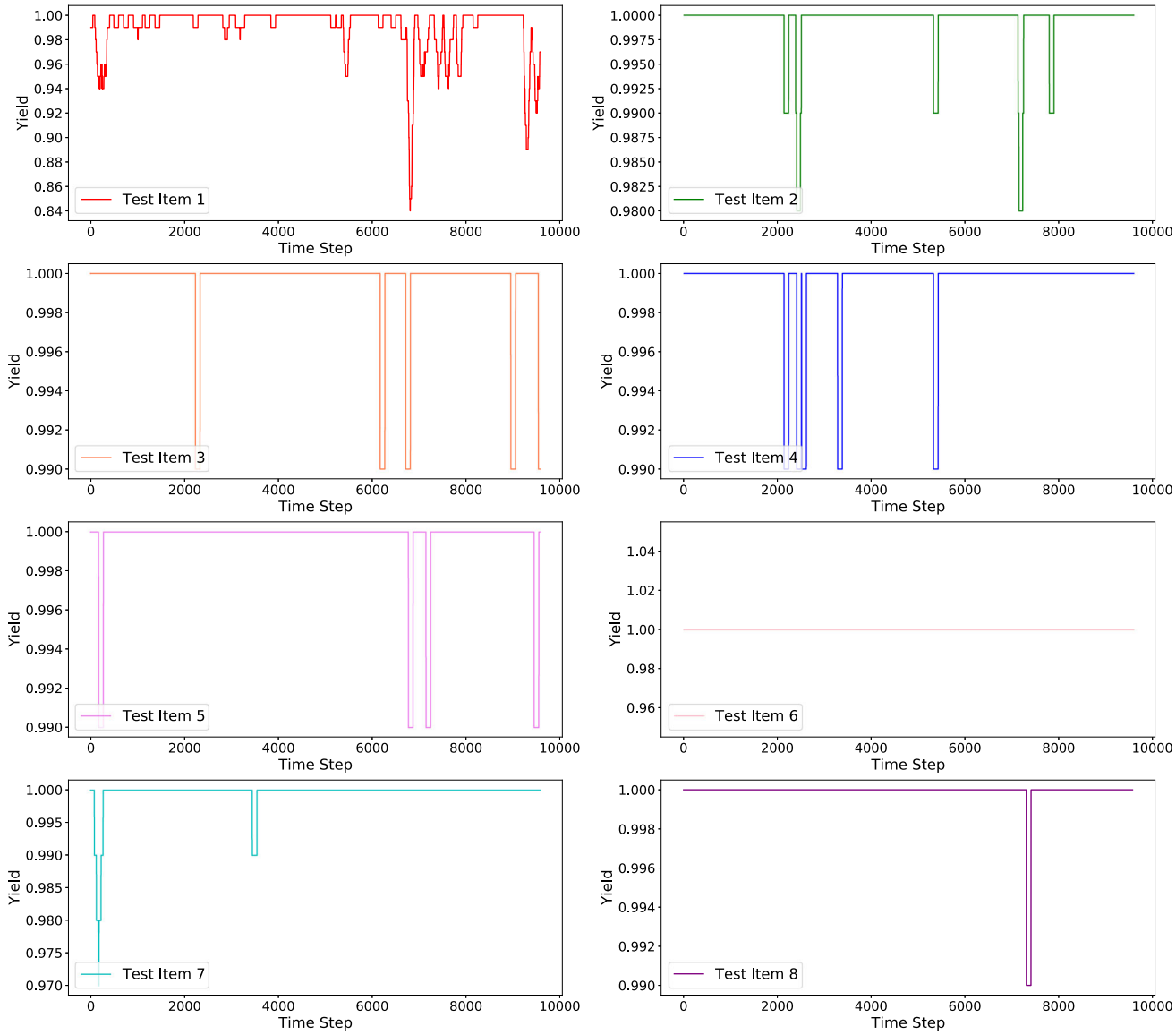


Fig. 7 The yield of the test items

where lr_0 represents the initial learning rate, g represents the decay rate of learning rate, i indicates that the current training epoch is the ith epoch, and t indicates that lr_0 decays every t epochs. This paper sets lr_0, g, and t to 0.001, 0.98, and 1, respectively.

In the following experiments, our goal is to obtain the optimal Window-Size, number of Encoder layers, and Hidden-Size. The horizon is 1 and the batch size is set to 32. To obtain the optimal sliding Window-Size, we first set the number of Encoder layers and the Hidden-Size to 4 and 256, and then calculate the evaluation metrics (RMSE, MAE, and MAPE) on the validation set using different Window-Size (10, 15, 20, 25, and 30), respectively. The experimental results obtained from different datasets are shown in Fig. 8.

Table 1 Dataset statistics, where T is the length of time series, D is the number of variables, L is the sample rate and V is the target variable

Datasets	T	D	L	V
NASDAQ100	40560	82	390 points per day	AAL
SML2010	4137	22	15 min	Indoor Temperature
Solar Energy	52560	137	10 min	The first plant's production
TIOB	9577	8	100 pieces	Test Item 1

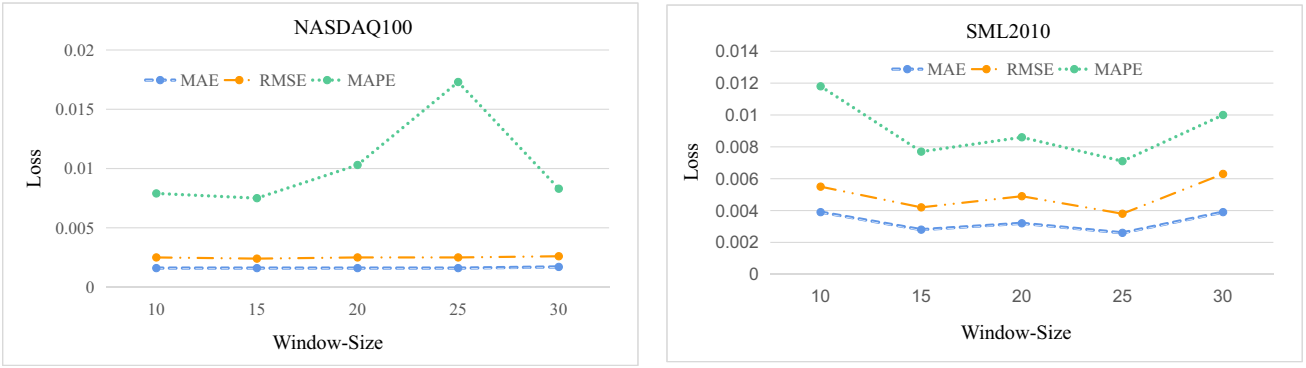


Fig. 8 The relationship between Window-Size and evaluation metrics for different datasets

It can be seen from Fig. 8 that for the NASDAQ100 dataset when the Window-Size equals 15, evaluation metrics largely reach the lowest value. When Window-Size is more than 15, MAPE begins to gradually rise, while RMSE and MAE keep steady. Therefore, the optimal Window-Size of the NASDAQ100 dataset is 15. For the SML2010 dataset, when the Window-Size equals 25, evaluation metrics all reach the lowest value. Therefore, the optimal Window-Size of the SML2010 dataset is 25.

To obtain the optimal number of Encoder layers, we keep the Window-Size unchanged and set the number of Encoder layers to 2, 4, 6, and 8, respectively. From the results of the validation set, we find that when the number of Encoder layers changes, the model performance is relatively stable. Eventually, we set the number of Encoder layers on both datasets to 6.

From the above experiments, the optimal Window-Size and the optimal number of Encoder layers corresponding to each dataset can be determined. With other conditions kept unchanged, the value of Hidden-Size is set to 64, 128, 256, 512, and 1024, respectively. The same validation set is used to calculate evaluation metrics to obtain the optimal Hidden-Size size. The experimental results of different datasets are shown in Fig. 9.

As can be seen from Fig. 9, for the NASDAQ100 dataset, the performance of the model keeps stable when the Hidden-Size is less than 1024. To make the model perform well on the test set, the Hidden-Size of the NASDAQ100 dataset is set to 512, which can learn more feature information. For the SML2010 dataset, when Hidden-Size equals 128, evaluation metrics obtain the minimum values. Therefore, the Hidden-Size of the SML2010 dataset is set to 128. In general, the optimal values of the Hidden-Size corresponding to each dataset can be set to 512 and 128, respectively.

To verify the generalization performance of the model, we fix the input length of all models to 25 when we experiment on the Solar Energy and TIOB datasets. The main hyperparameters of TCLN above the four datasets are shown in Table 2.

Our model is trained by the RMSE loss function and uses the ADAM optimizer to calculate and update the parameters of the TCLN. The date process and model training are shown in Algorithm 1. All experiments are implemented in PyTorch and conducted on a single NVIDIA GeForce RTX 3090 GPU.

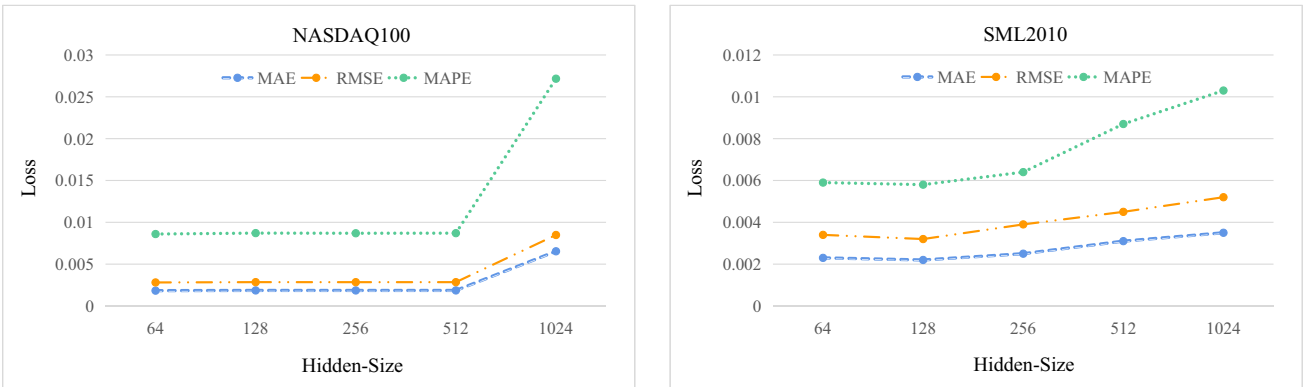


Fig. 9 The relationship between Hidden-Size and evaluation metrics for different datasets

Table 2 The hyper-parameters of TCLN on all datasets

	Window-Size	Number of Encoder layers	Hidden-Size
NASDAQ100	15	6	512
SML2010	25	6	128
Solar Energy	25	2	128
TIOB	25	6	512

Algorithm 1 Date process and model training.

Require: Epochs, the number of training epochs. Criterion, the loss function. Lr, the learning rate. Series, the input series. Visualize(·), the function of converting series into image form. TargetSeries, the past series of the target variable. Label, the ground truth. g, the decay rate of the learning rate.

1: NormSeries $\leftarrow$ (Series $-$ Series$_{\min}$)/(Series$_{\max}$ $-$ Series$_{\min}$)
2: GraphSeries $\leftarrow$ Visualize(NormSeries)
3: **for** $i \leftarrow 1$ **to** Epochs **do** :
4: **for** each batch **from** GraphSeries **do** :
5: Prediction $\leftarrow$ TCLN.forward(GraphSeries, TargetSeries)
6: Loss $\leftarrow$ Criterion(Prediction, Label)
7: Update the parameters of TCLN according to gradients and Lr
8: **end for**
9: Lr $\leftarrow$ Lr $\times$ g
10: **end for**

4.5 Main results

This section first illustrates the performance comparison of different methods and visualizes the forecasting results of two public datasets. Then the validity of the TCLN is demonstrated and the forecasting results of different horizons are also visualized. Next, the TCLN is applied to the board-level functional test to forecast the yield of the test items. Finally, an ablation study is conducted to verify the plausibility and validity of each component.

4.5.1 Performance comparison

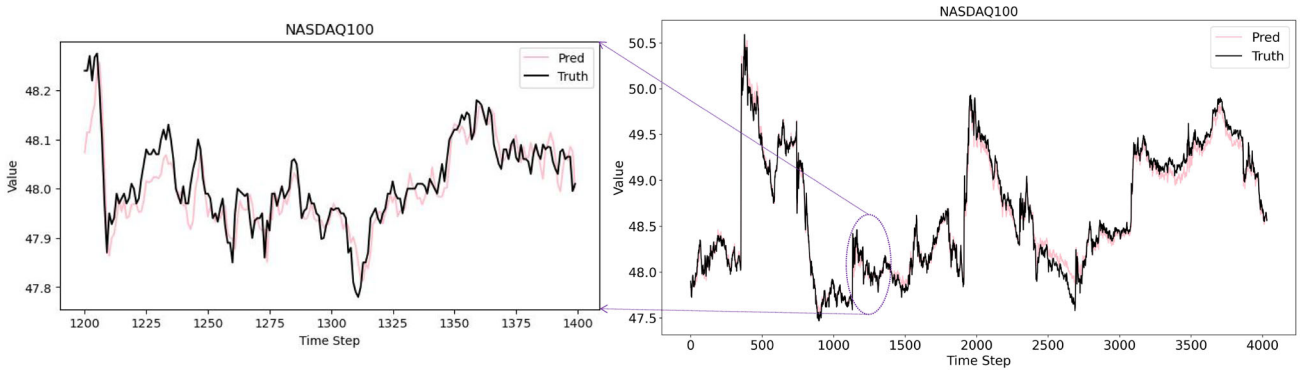
Through the previous experiments, we can get the optimal hyperparameters of the TCLN on each dataset. To verify the generalization ability of the TCLN, we calculate the evaluation metrics based on the optimal hyperparameters by using the test set. The experimental results are shown in Table 3.

From Table 3, we can find that the proposed method has achieved significant performance improvement over that of several state-of-the-art baseline methods. To observe the forecasting performance of the TCLN more intuitively, we display the forecasting results of the TCLN on NASDAQ100 and SML2010, which are shown in Fig. 10. In addition, we also compare the training time and model parameters of all

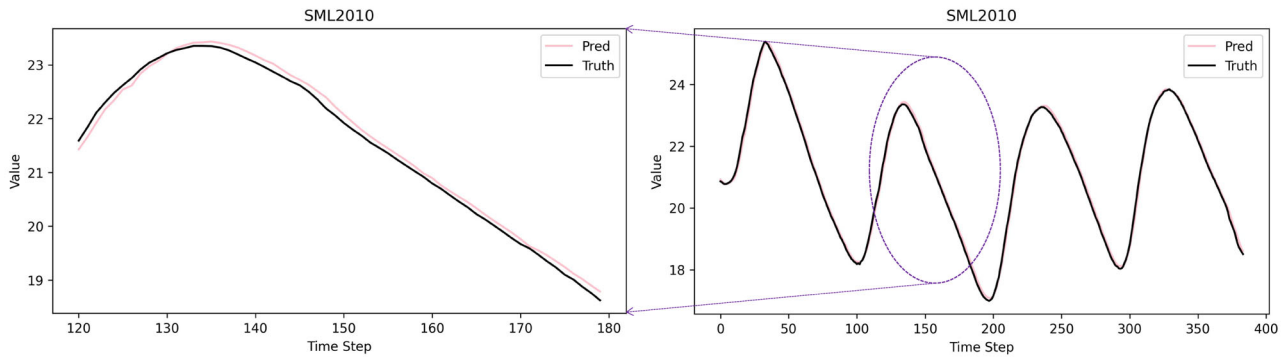
Table 3 Experimental performance comparison of all methods on four datasets

Datasets Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE
	NASDAQ100			SML2010		
CNN	1.9393429	2.0489585	4.005401	0.39808293	0.57038052	1.73886103
LSTM	0.5582465	0.6936417	1.1372567	0.11310571	0.18036349	0.52288972
GRU	0.2749340	0.3630726	0.5592619	0.05753389	0.07251797	0.26757324
StemGNN [27]	0.2152459	0.2449491	0.4400986	0.16526753	0.20664410	0.78252601
DA-RNN [12]	0.1482995	0.2175774	0.3010716	0.02423037	0.03356070	0.11546358
LSTNet [24]	0.1132649	0.1614985	0.2324591	0.32219078	0.33023251	1.55345080
DA-Conv-LSTM [11]	0.0962047	0.1415814	0.1957612	0.01934626	0.02908188	0.10361318
TS-Conv-LSTM [13]	0.0762847	0.1216078	0.1548931	0.01736006	0.02322524	0.08120701
TCLN	**0.0210528**	**0.0245163**	**0.0769483**	**0.01216436**	**0.01332673**	**0.04768813**
	Solar Energy*			TIOB*		
CNN	0.0310	0.0690	2.0379	0.0188	0.0367	0.0203
LSTM	0.0180	0.0412	1.2594	0.0201	0.0370	0.0216
GRU	0.0164	0.0390	1.0075	0.0331	0.0358	0.0340
TPA-LSTM[34]	0.0446	0.0663	0.9329	0.0242	0.0416	0.0261
DA-RNN [12]	0.0148	0.0374	0.7562	0.0180	0.0357	0.0194
LSTNet[24]	0.0153	0.0374	0.9731	0.0194	0.0379	0.0209
DA-Conv-LSTM[11]	0.0154	0.0383	0.6757	0.0193	0.0378	0.0208
TS-Conv-LSTM[13]	0.0145	0.0384	0.6837	0.0018	0.0031	0.0019
TCLN	**0.0143**	**0.0369**	**0.5853**	**0.0010**	**0.0014**	**0.0011**

Horizon is set to 1 and boldface represents the best results. The result of DA-RNN refers to [11], and other baselines' results refer to [13]. * denotes re-implementation for all methods on this dataset



(a) Prediction result for the TCLN on NASDAQ100.



(b) Prediction result for the TCLN on SML2010.

Fig. 10 The prediction results for the TCLN on different datasets when the horizon is 1

Table 4 The Complexity Analysis of all methods on the NASDAQ100 and SML2010 datasets

Datasets	NASDAQ100		SML2010	
Models	CC (s/epoch)	TNP	CC (s/epoch)	TNP
CNN	66.64	10743	6.751	13941
LSTM	4.80	14881	0.486	22081
GRU	4.73	11169	0.477	16577
DA-RNN	126.75	67549	11.802	51678
LSTNet	27.81	277252	1.472	74850
DA-Conv-LSTM	178.12	49813	16.711	48107
TS-Conv-LSTM	92.69	4772793	8.933	4754190
TCLN	24.12	5266940	2.39	334474

CC and TNP represent the computational cost and the total number of parameters, respectively

Table 5 Experimental performance of the TCLN when horizon takes different value

Datasets	NASDAQ100			SML2010		
Metrics	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Horizon=3	0.0290799	0.0347272	0.1051761	0.01565577	0.01801029	0.05569418
Horizon=6	0.0344071	0.0413187	0.1259621	0.03072104	0.03372397	0.09532821
Horizon=12	0.0623123	0.0686196	0.2010246	0.06789811	0.07307794	0.17646145
Horizon=24	0.0793681	0.0861951	0.2512565	0.14370426	0.15414449	0.51441608

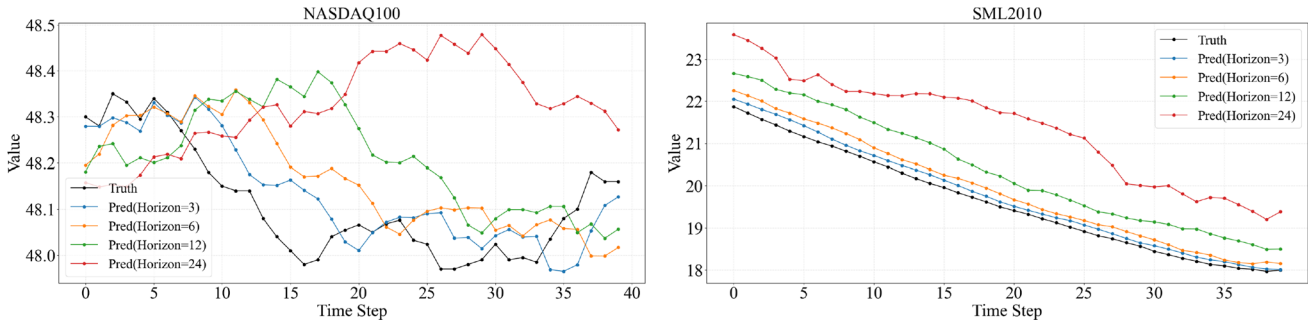


Fig. 11 The forecasting results of the TCLN on NASDAQ100 and SML2010 when horizon takes different value

models on these two datasets, which are shown in Table 4. It can be found that although TCLN has more parameters, in addition to simple models such as LSTM and GRU, the training time of TCLN per round is shorter than that of other complex models. Considering that the actual performance of TCLN is optimal, more parameters are acceptable.

4.5.2 Forecasting of long horizon

In practice, however, to have enough time to respond, we often need to predict the value of the target variable for a longer period. For example, when we can predict that the price of a stock will fluctuate over a longer period in the future, we can fully develop a corresponding strategy during this period. In this paper, to verify the performance of the long-term forecasting of the TCLN, the horizon is set to 3, 6, 12, and 24, respectively. Other parameters are determined by the results of the previous experiments. The experimental results are shown in Table 5.

According to the current research results [35, 36], when the evaluation metrics (RMSE, MAE, and MAPE) are all below 0.5, we can think that the forecasting results of the model are acceptable. As we can see from Table 5, the evaluation metrics are almost well below 0.5 for both datasets, except for the last one. Therefore, we can think that the long-term forecasting ability of the TCLN is still satisfactory.

Additionally, Fig. 11 illustrates the comparison between the forecasting value of the TCLN and the ground truth in different horizons. It can verify the effectiveness of the long-term forecasting performance of the TCLN intuitively.

4.5.3 Application to the yield forecasting of the test items

In addition to the public datasets, our model is also applied to the realistic yield forecasting of the test items. The yield of the test items is highly significant for the board-level functional test. Accurate yield forecasting can help the factory develop a test strategy in advance and improve the economic benefits of the factory. The forecasting result is shown in Fig. 12. As we can see from Fig. 7, the data of all test items is irregular. Especially for Test Item 1, the target variable, the changing trend is more complex than others. However, the forecasting result of Test Item 1 is still satisfactory though existing some deviations.

4.6 Ablation study

To demonstrate the effectiveness of the proposed components, a detailed ablation study is conducted. We remove or replace one of the components of the TCLN at a time. The TCLN without different components is named as follows.

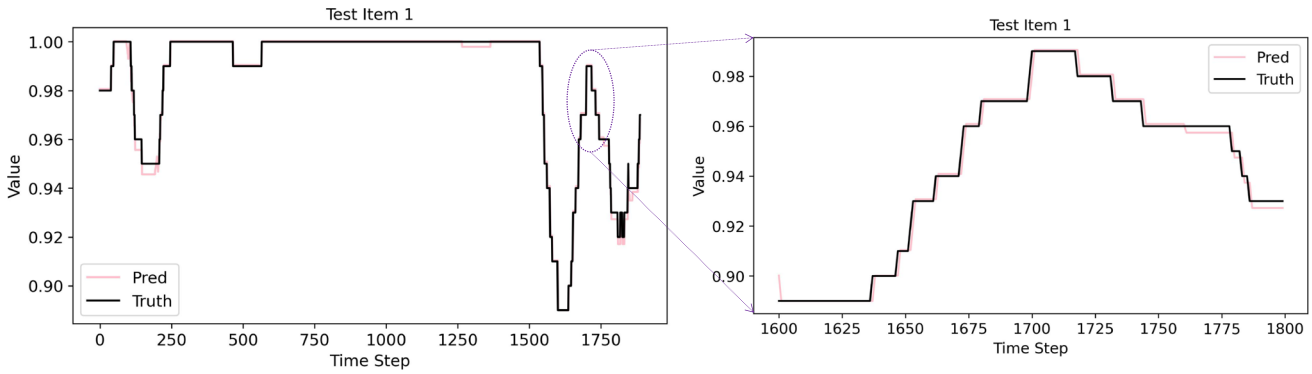


Fig. 12 The forecasting result of the TCLN for the Test Item 1 in Fig. 7

Table 6 Experimental performance of the TCLN in the ablation tests on NASDAQ100, SML2010, Solar Energy, and TIOB datasets

Ablation component	Metrics	NASDAQ100				SML2010				Solar Energy	TIOB
		h=1	h=3	h=6	h=12	h=1	h=3	h=6	h=12	h=1	h=1
TCLN	MAE	**0.0210528**	**0.0290799**	**0.0344071**	0.0623123	**0.01216436**	**0.01565577**	**0.03072104**	**0.06789811**	**0.0143**	0.0010
	RMSE	**0.0245163**	**0.0347272**	**0.0413187**	0.0686196	**0.01332673**	**0.01801029**	**0.03372397**	**0.07307794**	**0.0369**	0.0014
W/O LSTM	MAE	0.0648495	0.0572783	0.0454827	0.0965757	0.01252362	0.02029516	0.03537263	0.07415464	0.0164	0.0044
	RMSE	0.0695560	0.0636373	0.0523035	0.1046964	0.01420108	0.02287424	0.03907330	0.08075999	0.0393	0.0049
W/O Multi-kernel CNN	MAE	0.0961951	0.0959401	0.1087325	0.1360371	0.01468151	0.02061666	0.03944000	0.07362786	0.0154	0.0011
	RMSE	0.0993479	0.0999969	0.1135126	0.1408359	0.01610725	0.02347574	0.04418524	0.08149867	0.0390	0.0015
W/O AR	MAE	0.0300477	0.0382990	0.0466742	**0.0513629**	0.05483499	0.03443695	0.04558615	0.08447679	0.0143	**0.0009**
	RMSE	0.0326451	0.0432589	0.0528502	**0.0582448**	0.05912446	0.03798469	0.05069406	0.09280195	0.0381	**0.0013**
W/O Transformer	MAE	0.0361703	0.0756884	0.0632802	0.1005499	0.01552314	0.02101535	0.03579759	0.08000793	0.0148	0.0029
	RMSE	0.0389381	0.0792061	0.0685053	0.1061691	0.01682099	0.02307915	0.03871193	0.08803792	0.0374	0.0031

The boldface represents the best results and the second best results are highlighted with underlined font

- W/O LSTM: The TCLN without the LSTM network.
- W/O Multi-kernel CNN: The TCLN with a simple CNN layer.
- W/O AR: The TCLN without AR component.
- W/O Transformer: The TCLN without the Transformer encoder.

The test results measured using MAE, RMSE, and MAPE are shown in Table 6, where several observations are worth highlighting: (1) The best results on different horizons are almost obtained by the TCLN, verifying the effectiveness of all the components; (2) The performance of W/O AR significantly drops, showing the significance of the AR component. According to [24], the reason for the result is that the AR is generally robust to the scale changing in data; (3) W/O LSTM and W/O Transformer all get a performance loss, which indicates that the LSTM and Transformer encoder are both beneficial to the performance improvement of the TCLN; (4) The poor performance of W/O Multi-kernel CNN illustrates that the Multi-kernel CNN can learn more features compared with the simple CNN layer.

5 Conclusion

In this paper, we propose a novel model named TCLN for MTSF tasks. The TCLN mainly consists of the Multi-kernel CNN layer, Transformer encoder layer, LSTM network, and AR component. By combining the advantages of the self-attention mechanism and the LSTM network, the TCLN can effectively capture long-term temporal information of the time series. The Multi-kernel CNN layer can extract richer spatial information compared with the simple CNN and the AR component can enhance the robustness of the TCLN. The experimental results on the four datasets demonstrate that the TCLN outperforms the benchmark methods. Additionally, further experiments show that the TCLN can achieve a good performance in long-horizon forecasting and can also be applied to the task of industrial forecasting.

However, there are still some problems in the TCLN that can be improved:

1. The Encoder component still relies on the LSTM units, making the TCLN cannot be fully parallelized.
2. The Decoder component may be too simple to fully decode the results of the Encoder layers.
3. Due to the high computational complexity of the self-attention mechanism and the consistent attention to the input sequence, it will lead to a large amount of computation for the model. The extraneous features obtained may also affect the performance of the model.

In future work, the following research directions can be considered to obtain performance improvements:

1. To reduce the calculation of useless attention, we can screen the input of the self-attention layer by a specific algorithm.
2. In some MTSF tasks, we can try to add the graph structure into the model to obtain more spatial features.
3. For some datasets, there may be short and long periodicity ignored easily. Therefore, we can design a module to extract the periodicity features of the time series to improve forecasting accuracy.

Author Contributions Shusen Ma establishes the model, conducts the most of the experiments, and completes the writing of this paper. Tianhao Zhang mainly realizes the initial model and conducts partial supplementary experiments. Yun-Bo Zhao mainly supervises the work. Yu Kang and Peng Bai participate in the discussion of the work.

Funding This work was supported by the National Natural Science Foundation of China (No. 62173317) and the Key Research and Development Program of Anhui (No. 202104a05020064).

Data Availability The NASDAQ100 Stock dataset and SML2010 dataset can be obtained from [12]. The Solar Energy dataset can be obtained from [24]. The TIOB dataset will not be shared for it involves the privacy of the factory.

Declarations

Ethics approval Not applicable.

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Prakhar K, Sountharrajan S, Suganya E, Karthiga M, Kumar S (2022) Effective stock price prediction using time series forecasting. In: 6th International Conference on Trends in Electronics and Informatics (ICOEI) pp 1636–1640
2. Venkatachalam K, Trojovský P, Pamucar D, Bacanin N, Simic V (2023) DWFH: An improved data-driven deep weather forecasting hybrid model using transductive long short term memory (*T*-LSTM). Expert Syst Appl 213 (Part), 119270. https://doi.org/10.1016/j.eswa.2022.119270
3. Guo S, Lin Y, Feng N, Song C, Wan H (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: The thirty-third AAAI conference on artificial intelligence, pp 922–929. https://doi.org/10.1609/aaai.v33i01.3301922
4. Gao H, Su H, Cai Y, Wu R, Hao Z, Xu Y, Wu W, Wang J, Li Z, Kan Z (2021) Trajectory prediction of cyclist based on dynamic bayesian network and long short-term memory model at unsignalized intersections. Science China Information Sciences 64(7):172207. https://doi.org/10.1007/s11432-020-3071-8

5. Shi H, Zhu J, Kuang M, Yuan X (2021) Cooperative prediction guidance law in target-attacker-defender scenario. Sci China Inf Sci 64(4):149201. https://doi.org/10.1007/s11432-018-9806-7
6. Gefang D, Koop G, Poon A (2023) Forecasting using variational Bayesian inference in large vector autoregressions with hierarchical shrinkage. Int J Forecast 39(1):346–363
7. Zhang B, Chan JCC, Cross JL (2020) Stochastic volatility models with ARMA innovations: An application to G7 inflation forecasts. Int J Forecast 36(4):1318–1328
8. Khajavi H, Rastgoo A (2023) Improving the prediction of heating energy consumed at residential buildings using a combination of support vector regression and meta-heuristic algorithms. Energy 272:127069. https://doi.org/10.1016/j.energy.2023.127069
9. Swathi T, Kasiviswanath N, Rao AA (2022) An optimal deep learning-based LSTM for stock price prediction using twitter sentiment analysis. Appl Intell 52(12):13675–13688. https://doi.org/10.1007/s10489-022-03175-2
10. Bengio Y, Simard PY, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5(2):157–166. https://doi.org/10.1109/72.279181
11. Xiao Y, Yin H, Zhang Y, Qi H, Zhang Y, Liu Z (2021) A dual-stage attention-based Conv-LSTM network for spatio-temporal correlation and multivariate time series prediction. Int J Intell Syst 36(5):2036–2057. https://doi.org/10.1002/int.22370
12. Qin Y, Song D, Chen H, Cheng W, Jiang G, Cottrell GW (2017) A dual-stage attention-based recurrent neural network for time series prediction. In: IJCAI, pp 2627–2633. https://doi.org/10.24963/ijcai.2017/366
13. Fu E, Zhang Y, Yang F, Wang S (2022) Temporal self-attention-based Conv-LSTM network for multivariate time series prediction. Neurocomput 501:162–173. https://doi.org/10.1016/j.neucom.2022.06.014
14. Vaswani A,Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Proc Syst 30
15. Nascimento EGS, de Melo TAC, Moreira DM (2023) A transformer-based deep neural network with wavelet transform for forecasting wind speed and wind energy. Energy 278:127678. https://doi.org/10.1016/j.energy.2023.127678
16. Zerveas G, Jayaraman S, Patel D, Bhamidipaty A, Eickhoff C (2021) A transformer-based framework for multivariate time series representation learning. In: KDD '21: The 27th ACM SIGKDD conference on knowledge discovery and data mining pp 2114–2124. https://doi.org/10.1145/3447548.3467401
17. Fu X, Guo Q, Sun H (2020) Statistical machine learning model for stochastic optimal planning of distribution networks considering a dynamic correlation and dimension reduction. IEEE Transactions on Smart Grid 11(4):2904–2917. https://doi.org/10.1109/TSG.2020.2974021
18. Fu X (2022) Statistical machine learning model for capacitor planning considering uncertainties in photovoltaic power. Protect Contr Mod Power Syst 7(1):5. https://doi.org/10.1186/s41601-022-00228-z
19. Pan S, Long S, Wang Y, Xie Y (2023) Nonlinear asset pricing in Chinese stock market: A deep learning approach. Int Rev Fin Anal 87:102627. https://doi.org/10.1016/j.irfa.2023.102627
20. Mohimont L, Chemchem A, Alin F, Krajecki M, Steffenel LA (2021) Convolutional neural networks and temporal CNNs for COVID-19 forecasting in France. Appl Intell 51(12):8784–8809. https://doi.org/10.1007/s10489-021-02359-6
21. Banerjee T, Sinha S, Choudhury P (2022) Long term and short term forecasting of horticultural produce based on the LSTM network model. Appl Intell 52(8):9117–9147. https://doi.org/10.1007/s10489-021-02845-x
22. Li G, Zhong X (2023) Parking demand forecasting based on improved complete ensemble empirical mode decomposition and GRU model. Eng Appl Artif Intell 119:105717. https://doi.org/10.1016/j.engappai.2022.105717
23. Xu W, Peng H, Zeng X, Zhou F, Tian X, Peng X (2019) A hybrid modelling method for time series forecasting based on a linear regression model and deep learning. Appl Intell 49(8):3002–3015. https://doi.org/10.1007/s10489-019-01426-3
24. Lai G, Chang W, Yang Y, Liu H (2018) Modeling long- and short-term temporal patterns with deep neural networks. In: The 41st international ACM SIGIR conference on research & development in information retrieval pp 95–104. https://doi.org/10.1145/3209978.3210006
25. Yang Y, Lu J (2022) Foreformer: an enhanced transformer-based framework for multivariate time series forecasting. Appl Intell 1–20
26. Chen Z, Chen D, Zhang X, Yuan Z, Cheng X (2022) Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. IEEE internet things J 9(12):9179–9189. https://doi.org/10.1109/JIOT.2021.3100509
27. Cao D, Wang Y, Duan J, Zhang C, Zhu X, Huang C, Tong Y, Xu B, Bai J, Tong J et al (2020) Spectral temporal graph neural network for multivariate time-series forecasting. Adv Neural Inf Proc Sys 33:17766–17778
28. Shang C, Chen J, Bi J (2021) Discrete graph structure learning for forecasting multiple time series. In: 9th international conference on learning representations. https://openreview.net/forum?id=WEHSlH5mOk
29. Wu Z, Pan S, Long G, Jiang J, Chang X, Zhang C (2020) Connecting the dots: multivariate time series forecasting with graph neural networks. In: KDD '20: The 26th ACM SIGKDD conference on knowledge discovery and data mining pp 753–763. https://doi.org/10.1145/3394486.3403118
30. Fu X, Zhou Y (2023) Collaborative optimization of PV greenhouses and clean energy systems in rural areas. IEEE transactions on sustainable energy 14(1):642–656. https://doi.org/10.1109/TSTE.2022.3223684
31. Huang X, Tang J, Yang X, Xiong L (2022) A time-dependent attention convolutional LSTM method for traffic flow prediction. Appl Intell 52(15):17371–17386. https://doi.org/10.1007/s10489-022-03324-7
32. Ren Q, Li Y, Liu Y (2023) Transformer-enhanced periodic temporal convolution network for long short-term traffic flow forecasting. Expert Syst Appl 227:120203. https://doi.org/10.1016/j.eswa.2023.120203
33. Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: IEEE conference on computer vision and pattern recognition pp 1–9. https://doi.org/10.1109/CVPR.2015.7298594
34. Shih S-Y, Sun F-K, Lee H-y (2019) Temporal pattern attention for multivariate time series forecasting. Mach Learn 108:1421–1441
35. Cheng Q, Chen Y, Xiao Y, Yin H, Liu W (2022) A dual-stage attention-based Bi-LSTM network for multivariate time series prediction. J Supercomput 78(14):16214–16235. https://doi.org/10.1007/s11227-022-04506-3
36. Wang Q, Chen L, Zhao J, Wang W (2020) A deep granular network with adaptive unequal-length granulation strategy for long-term time series forecasting and its industrial applications. Artif Intell Rev 53(7):5353–5381. https://doi.org/10.1007/s10462-020-09822-9