

TCLformer: Enhancing Multi-Scale Time Series Forecasting with Temporal Decomposition and Sparse Convolutional Attention

Xiaohe Wu

Nanjing University of Science and Technology

Kun Zhang

kunzhang1977@163.com

Nanjing University of Science and Technology

Chenxi Cai

Nanjing University of Science and Technology

Dianying Chen

Nanjing University of Science and Technology

Yaodi Liu

Nanjing University of Science and Technology

Research Article

Keywords: Time series forecasting. Multi-scale. TCN. Time series decomposition. Attention

Posted Date: October 29th, 2025

DOI: <https://doi.org/10.21203/rs.3.rs-7386157/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

TCLformer: Enhancing Multi-Scale Time Series Forecasting with Temporal Decomposition and Sparse Convolutional Attention

Xiaohe Wu¹, Kun Zhang^{2*}, Chenxi Cai², Dianying Chen², Yaodi Liu²

1. School of Cyberspace Security, Nanjing University of Science and Technology, Nanjing, 210094, China
2. School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, 210094, China

Abstract: Real-world multivariate time series (MTS) often contain complex multi-scale temporal dependencies, and accurately capturing their short-term and long-term patterns remains a challenging task. Although Transformer-based models have made significant progress in this field in recent years, they still suffer from high computational overhead and limited ability to model long-term dependencies. Convolutional-based models, while computationally efficient, struggle to capture global dependencies and tend to overlook periodic and trend information in time series. To address this, we propose a predictive model that combines multi-scale adaptive convolutions with sparse convolutional attention mechanisms, named TCLformer. The model first performs trend-seasonal decomposition on the input time series to extract key intertwined temporal patterns; then, it uses parallel multi-scale convolutional modules to adaptively capture local features at different time scales. Finally, it introduces a sparse attention module based on causal convolutions, to efficiently model long-term dependencies. Experimental results on six public datasets demonstrate that TCLformer significantly outperforms existing state-of-the-art methods in terms of prediction accuracy, validating its effectiveness and superiority in multi-variable time series prediction tasks.

Keywords: Time series forecasting. Multi-scale. TCN. Time series decomposition. Attention

1. Introduction

Time series data is a type of data related to the sequence of time. Time series forecasting techniques aim to predict future trends by modeling historical observations. Time series forecasting techniques play a crucial role in practical applications such as weather forecasting[1], traffic flow prediction[2], Energy[3], power demand forecasting[4], and financial system[5]. Real-world multivariate time series exhibit complex temporal patterns, with different scales containing distinct information, making multivariate time series forecasting a challenging task, particularly for long-term predictions where performance tends to deteriorate.

¹ E-mail address: wuxiaohe@njust.edu.cn

^{2*}Corresponding author, E-mail address: zhangkun@njust.edu.cn

Time series data often exhibit complex structures such as high-dimensionality, strong nonlinearity, interactions between variables, and multi-scale temporal characteristics. These characteristics make traditional statistical modeling methods (such as ARIMA and VAR)[6][7] and machine learning models perform poorly when dealing with complex dependencies and large-scale data, failing to meet the requirements for accuracy and generalization capabilities[8]. With the development of deep learning, sequence modeling methods based on RNN[9], LSTM[10], and CNN[11] have been widely applied to time series tasks. These models have made some progress in modeling short-term dependencies, but due to the limitations of their cyclical structure, they encounter problems such as gradient disappearance and error accumulation when dealing with long sequences, which causes the effectiveness of the model to decline as the prediction time step increases[12]. The Transformer architecture, with its self-attention mechanism, has gained significant attention in the field of time series forecasting due to its advantages in modeling long-range dependencies[13]. However, the self-attention mechanism of the standard Transformer exhibits quadratic growth in computational and memory complexity with sequence length, leading to severe efficiency bottlenecks when handling long-term sequences. Although various improvement methods have been proposed to reduce the computational overhead of the attention mechanism, such as Log-sparse and Reformer[14], multi-variable time series prediction still faces the following main challenges: (1) While Transformers possess strong long-term dependency modeling capabilities, its quadratic complexity limits its application in long sequence tasks[15]. (2) Real-world time series often exhibit complex interactions between multiple variables and dynamic temporal dependencies, increasing modeling complexity. (3) Time series data commonly contain multi-scale temporal structures, such as trend components, periodic components, and local perturbations, and how to capture multi-scale temporal information remains a challenge.

Based on the aforementioned issues, this paper proposes a time series prediction model that integrates multi-scale adaptive convolution with a lightweight convolutional attention mechanism—TCLformer. This model combines the strong local temporal feature perception capabilities of temporal convolutions with the efficient long-term dependency modeling capabilities of the Log-sparse attention mechanism, achieving a good balance in modeling multi-scale, long-term, and short-term dependency features. The main contributions of this paper are as follows:

(1) We propose a prediction framework that combines TCN, convolutional attention mechanism, and seasonal-trend decomposition module to achieve efficient modeling of multi-scale, non-stationary time series signals.

(2) By introducing a parallel variable convolutional structure, we incorporate scale information into local modeling. Combined with the trend-seasonal decomposition module, we explicitly separate the trend and periodic components in the sequence, thereby improving the interpretability of the model.

(3) This paper replaces the global self-attention mechanism in the standard Transformer with a structured convolution-guided log-sparse attention mechanism. This mechanism introduces convolution kernel priors in attention weight calculations, guiding the model to focus on local context regions, significantly enhancing its ability to model local time series structures (such as sudden changes, anomalies, and short-term fluctuations). Compared to fully connected self-attention, the computational complexity is reduced from $O((L)^2)$ to $O(L(\text{Log}L)^2)$, significantly improving the

processing efficiency for long sequences while maintaining the ability to model long-term dependencies.

(4) TCLformer was extensively tested on six publicly available real-world multivariate time series datasets, and our method achieved the best performance, fully demonstrating the effectiveness of the proposed method.

The remainder of this paper is organized as follows. In Section 2, we introduce the current state of the art in time series forecasting. In Section 3, we first present the problem and definition of multivariate time series forecasting, and then describe the model proposed in this paper in detail. Section 4 provides a detailed introduction to the dataset and experimental setup, and compares the performance of the method proposed in this paper with other benchmark time series forecasting models. Section 5 is the discussion section. Finally, Section 6 presents the conclusions and future research directions.

2. Related works

2.1 Time series forecasting

Time series forecasting plays an indispensable role in everyday life. With the rise of deep learning, time series forecasting models have shifted from traditional ARIMA-based approaches to deep learning-based methods. Recurrent neural networks (RNNs) are specifically designed for time series forecasting, but they face issues of gradient vanishing and gradient explosion in deep networks. The emergence of LSTMs and CNNs alleviated these limitations, but they still cannot capture long-term dependencies. DeepAR[16] combines recurrent neural networks and autoregressive AR to predict time series. As a classic CNN-based model, TCN[17] uses causal convolutions to simulate temporal causal relationships and dilated convolutions to expand the receptive field. It can better integrate local information in sequences and achieves competitive results in medium-to-short-term predictions[18]. However, due to the limitations of the receptive field size, TCN typically requires multiple layers to model the global relationships of time series, significantly increasing the network's complexity and the difficulty of model training. MDTNet[19] uses stacked dilated convolutions and GRUs to capture intertemporal dependencies and temporal patterns. The extracted features remain localized, making it challenging to capture long-term dependencies. LSTNet[20] uses CNNs to capture cross-dimensional dependencies, and RNNs to capture cross-temporal dependencies. DSANet[21] employs two parallel convolutional modules to capture local and global dependencies, respectively. SCINet[22] is a hierarchical framework that captures dependencies at multiple temporal resolutions through sequence segmentation, convolution, and interactive learning. TimesNet[23] captures temporal dependencies at multiple temporal resolutions.

Methods based on MLP have demonstrated high efficiency in modeling time series. A simple yet effective baseline model, DLinear[24] questioned the effectiveness of Transformer-based models for time series prediction, arguing that the self-attention mechanism may lead to the loss of temporal information. It proposed a linear regression model that combines sequence decomposition with a single-layer linear layer. This

simple yet effective model outperforms existing Transformer-based models. RLinear[25] systematically investigates the role of linear mapping in long-term time series forecasting, proposing that preprocessing time series data with RevIN and CI before inputting it into a linear layer can enhance model performance. TimeMixer[26] introduces a fully MLP-based multi-scale hybrid architecture, first utilizing downsampling to capture information at different scales, then uses sequence decomposition blocks at each scale to obtain seasonal and trend components, aggregates seasonal and trend information at each scale, and finally performs regression using a single linear layer. MTS-Mixers[27] proposed the MTS Mixer, which uses time and channel MLP decomposition factorization modules to capture time and channel dependencies. Recently, MLP-based architectures have achieved success in the field of computer vision [28][29]. Inspired by the MLP Mixer architecture in computer vision, TSMixer[30] proposes an architecture that alternately applies MLP in the time domain and feature domain—TSMixer, where the temporal MLP is shared across all features, and the feature-domain MLP is shared across all time steps. LightTS[31] proposes a lightweight deep learning architecture based solely on a simple MLP structure, inspired by the key fact that down sampled time series typically retain most of their information. By applying an MLP-based structure on top of two refined down sampling strategies (including interval sampling and continuous sampling), it achieves good performance. Although Transformer-based models perform well on long-term time series problems, they still have some limitations, including high time and memory complexity and error accumulation caused by the autoregressive encoder itself. Although MLP-based models have achieved outstanding results, they struggle to effectively capture complex temporal patterns across different time scales (hours, days, weeks, months) in real-world time series.

2.2 Transformer-based time series forecasting models

Transformer-based models can capture long-term remote dependencies due to their use of multi-head attention mechanisms. They have achieved surprising results not only in the fields of natural language processing (NLP)[32], computer vision (CV)[33], and audio[34], but also show great potential in long-term time series prediction tasks. However, their high computational complexity and memory consumption limit their widespread application in time series prediction. To overcome this bottleneck, various Transformer-based variants have gained popularity. Autoformer[35] first uses moving average decomposition to separate the seasonal and trend components in complex time series, then employs an autocorrelation mechanism to achieve better prediction performance. FEDformer[36] preprocesses data using multi-core moving averages, then utilizes Fourier enhancement modules and wavelet enhancement modules to obtain the final prediction results. Informer[37], LogTrans[38], and Pyraformer[39] respectively improve the self-attention mechanism to reduce temporal and spatial complexity. TDformer[40] first applies seasonal trend decomposition, then combines the MLP for predicting the trend component with the Fourier attention for predicting the seasonal component to obtain the final prediction. Inspired by Fredformer[41], FCAFormer[42] integrates frequency-domain modeling with channel attention

mechanisms, enabling the simultaneous learning of long-term temporal patterns and the dynamic importance of frequency features. FreEformer[43] first convert time series into the complex frequency domain using the Discrete Fourier Transform (DFT). The Transformer architecture is then applied to the frequency spectra to capture cross-variate dependencies, with the real and imaginary parts processed independently, and enhance the vanilla attention mechanism by introducing an additional learnable matrix to the original attention matrix, followed by row-wise L1 normalization.

TFT[44] proposed an interpretable Transformer architecture that combines prediction with interpretability of temporal dynamics, using recurrent layers to extract local features and interpretable self-attention layers to model long-term dependencies. PatchTST[45] is a multi-variable time series prediction model based on the self-supervised pre-training task of the Transformer model, which takes the original time series divided into sub-sequence-level patches as input, thereby retaining local semantic information while reducing memory and computational overhead. Crossformer[46] argues that existing transformer-based models primarily focus on modeling temporal dependencies (cross-temporal dependencies) but often overlook dependencies between different variables (cross-dimensional dependencies). Therefore, it first divides each channel's long sequence into fixed-length segments and then embeds them into feature vectors. Finally, a two-stage attention layer is used to capture both temporal and cross-dimensional dependencies. MICN[47] employs equidistant convolutions to model global correlation dependencies in a multi-scale framework, and down sampling convolutions to aggregate temporal sequence information from local contexts. Scaleformer[48] introduces a new iterative scale refinement paradigm, which can be easily adapted to various Transformer architectures. ETSformer[49] replaces the traditional self-attention in Transformers with new exponential smoothing attention and frequency-domain attention, where exponential smoothing attention obtains attention scores based on the relative lag time of the query rather than content similarity, and frequency-domain attention extracts the main seasonal patterns in the sequence by selecting the K values with the largest amplitude in the frequency domain.

3. Methodology

3.1 Problem definition and notations

Multivariate time series forecasting is the task of predicting the future values of one or more variables at future time points given a set of historical observations. Let $\mathbf{x}_t \in \mathbb{R}^m$ denote an observation of a multivariate time series at time step t and m is the dimension of observed signals. Given a look-back window $\mathbf{X}_{t-L:t} = [\mathbf{x}_{t-L}, \mathbf{x}_{t-L+1}, \dots, \mathbf{x}_{t-1}] \in \mathbb{R}^{L \times m}$, where L is the length of input time series. Given a forecasting horizon H , $\mathbf{X}_{t:t+H} = [\mathbf{x}_t, \mathbf{x}_{t+1}, \dots, \mathbf{x}_{t+H-1}]$. We denote $\hat{\mathbf{X}}_{t:t+H} \in \mathbb{R}^{H \times m}$ as the point forecast of $\mathbf{X}_{t:t+H}$. The goal of time series forecasting is to learn a map $\hat{\mathbf{X}}_{t:t+H} = f(\mathbf{X}_{t-L:t})$ by minimizing a specific loss function $\mathcal{L}: \mathbb{R}^{H \times m} \times \mathbb{R}^{H \times m} \rightarrow \mathbb{R}$. $f(\cdot)$ denotes the prediction mapping function. In short, given a sequence of one or more observed variables, the main purpose of time series forecasting is to use past observations to obtain the most likely future predictions.

3.2 Overall Architecture

As shown in Fig. 1, TCLformer adopts a residual and encoder-decoder architecture, but upgrades the traditional Transformer. The encoder consists of a TCN and a time series decomposition module, where TCN and time series decomposition module are used to extract the seasonal components of the sequence. The decoder consists of a TCN and a log-sparse cross-attention mechanism.

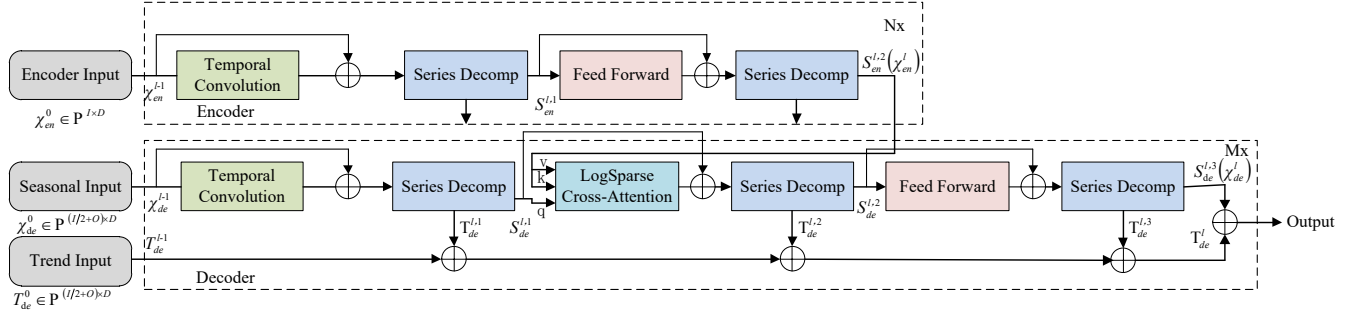


Fig. 1 The overall architecture TCLformer.

Temporal Convolution Network. Compared to RNN, TCN has stronger parallelism and more flexible receptive fields, and requires less memory when dealing with long sequences. The benefits of using TCN are: 1) The convolutions in the architecture are causal, meaning there is no information leakage from the future to the past; 2) Residual layers and dilated convolutions can be used to construct very long effective historical scales, and the input sequence length is highly flexible, supporting both fixed-length and variable-length sequence data; 3) TCN employs convolutional operations, enabling efficient parallel computing and making it suitable for large-scale data processing. The structure of TCN is shown in Fig. 2. TCN uses dilated convolutions to exponentially increase the receptive field. By stacking convolutional layers and increasing the receptive field of the convolutional kernels, TCN can effectively capture local dependencies in sequence data. TCN can extract features at different scales by stacking multiple convolutional layers. Each convolutional layer processes the input using convolutional kernels of different sizes, thereby capturing features across different time scales in sequential data. This multi-scale information extraction capability makes TCN more sensitive to local dependencies in sequential data.

For 1-D sequence input $x \in \mathbb{R}$ and a filter $f: \{0, \dots, k-1\} \rightarrow \mathbb{R}$, the dilated convolution operation F on element s of the sequence is defined as

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \quad (1)$$

Where d is the dilation factor, k is the filter size, and $s - d \cdot i$ accounts for the direction of the past.

Using a larger expansion allows the top-layer output to represent a wider range of inputs, thereby effectively expanding the receptive field of the convolutional neural network. A larger receptive field enables the model to extract temporal patterns across multiple scales in time series.

Time Series Decomposition. The future values of a time series are determined by the interplay of multiple scales. Existing research has shown that separating the seasonal and trend components can improve forecasting performance[25]. Time series decomposition is a data processing method widely used in time series forecasting. Its purpose is to decompose an original sequence into trend and residual components to better capture the structural characteristics of the time series and thereby enhance forecasting performance. We adopt an approach similar to Autoformer[35], incorporating the decomposition module as an internal block within the model, progressively extracting the long-term stationary trend from the intermediate hidden variables of the prediction, including past sequences and intermediate prediction results. Autoformer uses a fixed sliding window kernel size for mean filtering to extract the trend component, but this approach has limitations: the fixed window may not adapt well to time series with multi-scale features, resulting in insufficient fitting capability for complex trends and potential trend-residual mixing. Therefore, we adopt a multi-scale learnable fusion decomposition module, using moving averages with multiple sliding windows to model trends at different scales separately, dynamically fusing different scale receptive fields, making it more suitable for time series with complex or multi-scale trend changes.

Set a group of different kernel sizes $\{k_1, k_2, \dots, k_n\}$, For each scale k_i , calculate its moving average trend:

$$\chi_t^{(i)} = AvgPool_{k_i}(Padding(\chi)), i = 1, 2, \dots, n \quad (2)$$

Introduce a learnable function $\alpha(\chi) \in \mathbb{R}^n$, representing the weighted weights for each scale trend, satisfying Softmax normalization:

$$\alpha = Softmax(W \cdot \chi), W \in \mathbb{R}^{1 \times n} \quad (3)$$

Ultimately, the fusion trend items are:

$$\chi_t = \sum_{i=1}^n \alpha_i \cdot \chi_t^{(i)} \quad (4)$$

Extraction of residual (seasonal) components:

$$\chi_s = \chi - \chi_t \quad (5)$$

χ_s and χ_t represent the extracted seasonal cycle component and trend component, respectively.

We represent Series Decomp Block as:

$$\chi_s, \chi_t = SD(\chi) \quad (6)$$

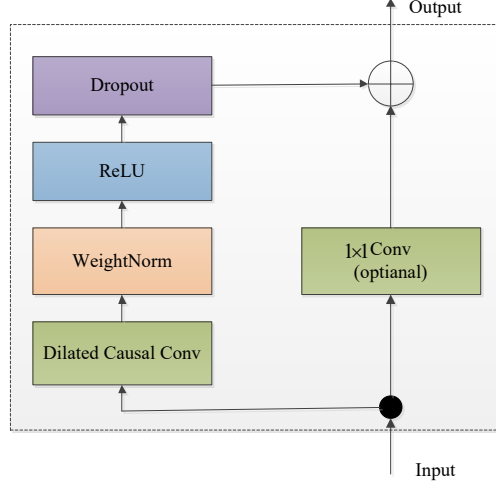


Fig. 2 Temporal convolution block architecture.

Encoder. The encoder adopts a multi-layer structure, focusing mainly on modeling seasonal components. $\chi_{en}^l = \text{Encoder}(\chi_{en}^{l-1})$, where $l \in \{1, \dots, N\}$ denotes the output of l -th encoder layer and $\chi_{en}^0 \in \mathbb{R}^{I \times D}$ is the embedded historical series. The $\text{Encoder}(\cdot)$ is formalized as:

$$\mathcal{S}_{en, -}^{l,1} = SD(TCN(\chi_{en}^{l-1}) + \chi_{en}^{l-1}) \quad (7)$$

$$\mathcal{S}_{en, -}^{l,2} = SD(\text{FeedForward}(\mathcal{S}_{en}^{l,1}) + \mathcal{S}_{en}^{l,1}) \quad (8)$$

$$\chi_{en}^l = \mathcal{S}_{en}^{l,2} \quad (9)$$

Where $\mathcal{S}_{en}^{l,i}, i \in \{1,2\}$ represents the seasonal component after the i -th decomposition block in the l -th layer respectively.

Decoder. The decoder also adopts a multilayer structure as: $\chi_{de}^l, \mathcal{T}_{de}^l = \text{Decoder}(\chi_{en}^{l-1}, \mathcal{T}_{de}^{l-1})$, where $l \in \{1, \dots, M\}$ denotes the output of l -th decoder layer. The $\text{Decoder}(\cdot)$ is formalized as:

$$\mathcal{S}_{de}^{l,1}, \mathcal{T}_{de}^{l,1} = SD(TCN(\chi_{de}^{l-1}) + \chi_{de}^{l-1}) \quad (10)$$

$$\mathcal{S}_{de}^{l,2}, \mathcal{T}_{de}^{l,2} = SD(CA(\mathcal{S}_{de}^{l,1}, \chi_{en}^N) + \mathcal{S}_{de}^{l,1}) \quad (11)$$

$$\mathcal{S}_{de}^{l,3}, \mathcal{T}_{de}^{l,3} = SD(\text{FeedForward}(\mathcal{S}_{de}^{l,2}) + \mathcal{S}_{de}^{l,2}) \quad (12)$$

$$\chi_{de}^l = \mathcal{S}_{de}^{l,3} \quad (13)$$

$$\mathcal{T}_{de}^l = \mathcal{T}_{de}^{l-1} + \mathcal{W}_{l,1} \cdot \mathcal{T}_{de}^{l,1} + \mathcal{W}_{l,2} \cdot \mathcal{T}_{de}^{l,2} + \mathcal{W}_{l,3} \cdot \mathcal{T}_{de}^{l,3} \quad (14)$$

Where $\mathcal{S}_{de}^{l,i}, \mathcal{T}_{de}^{l,i} \in \{1,2,3\}$ represents the seasonal component and the trend component after the i -th decomposition block in the l -th layer respectively. $\mathcal{W}_{l,i}, i \in \{1,2,3\}$ represents the projection for the i -th extracted $\mathcal{T}_{de}^{l,i}$.

The final prediction is the sum of the two refined decomposed components as $\mathcal{W}_S \cdot \chi_{de}^M + \mathcal{T}_{de}^M$, where $\mathcal{W}_S$ is to projection the deep transformed seasonal component χ_{de}^M to the target dimension.

3.3 Convolutional Attention

As shown in Fig. 3 (a), the similarity between the query and key in the self-attention mechanism of the standard Transformer architecture is calculated on a per-point basis, resulting in insensitivity to local context. Additionally, the spatial complexity of the traditional Transformer increases quadratically with sequence length, limiting its ability to model long sequences. To address these two issues, as shown in Fig. 3(b), we replace the self-attention mechanism with convolutional self-attention. When computing the key and query, we use a convolutional kernel with a size greater than 1 (equal to 1 corresponds to the original Transformer) and a stride of 1 for causal convolution, enabling the attention mechanism to focus on local context and match more relevant features. Causal convolution ensures that the current position can never access future information. By using causal convolution, the generated queries and keys can better understand the local context, enabling their similarity to be calculated based on their local contextual information (e.g., local shape) rather than point-wise values, thereby improving prediction accuracy. Note that when $k=1$, convolutional self-attention degrades to standard self-attention, so it can be viewed as a generalization. By introducing convolutional self-attention, on one hand, it enhances the focus on local contextual information, and on the other hand, it alleviates the issue of excessive spatial complexity in self-attention, enabling the model to handle longer time series data[38].

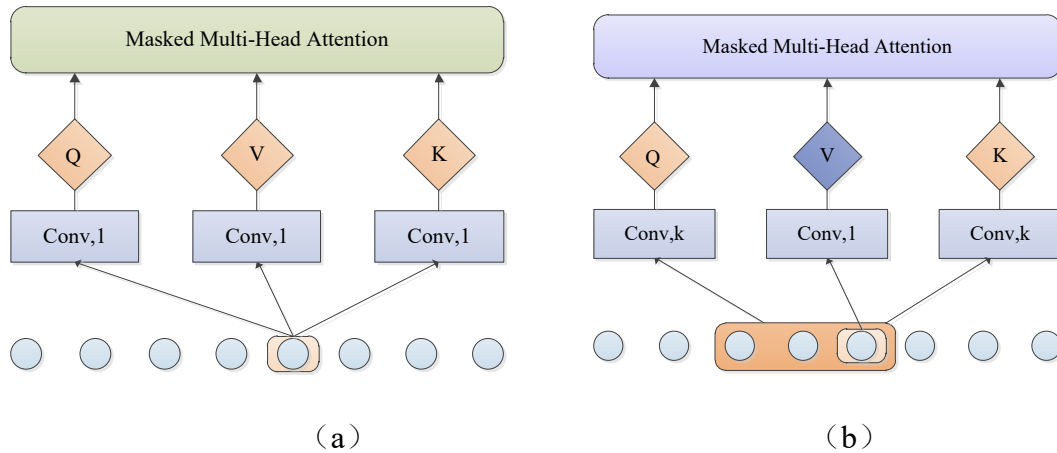


Fig. 3 The comparison between canonical and our convolutional self-attention layers. “Conv,1” and “Conv, k” mean convolution of kernel size $\{1,k\}$ with stride 1, respectively.

4. Experiments

4.1 Datasets

Our experiments were conducted on six public real-world datasets: (1) ETTh2 (Electricity Transformer Temperature)[37] records data from electricity transformer every 1 hour from July 2016 to July 2018. (2) Traffic³ contains the hourly road occupancy rates measured by 862 sensors on highways in the San Francisco area, recorded by the California Department of Transportation from 2015 to 2016. (3) Electricity⁴ contains the hourly electricity consumption of 321 clients. (4) Exchange[20] contains daily exchange rates of 8 countries daily ranging from 1990 to 2016. (5) Weather⁵ contains 21 meteorological coordinates in Germany, recorded every 10 minutes for 2020 whole year. (6) ILI⁶ contains influenza-like illness (ILI) patients data weekly from the US Center for Disease Control and Prevention from 2002 to 2021. To ensure a fair comparison, we followed the same approach as in PatchTST[45] and divided all datasets into training, validation, and test sets. Table 1 provides information on these six real-world datasets.

Table 1 Statistical information of all datasets for time series forecasting

Datasets	ETTh2	Traffic	Electricity	Exchange	Weather	ILI
Variates	7	862	321	8	21	7
Timesteps	69680	17544	26304	7588	52696	966
Granularity	5min	1hour	1hour	1day	10min	7day
Data Partition	6:2:2	7:1:2				

4.2 Experimental setup

4.2.1 Evaluation metrics

To evaluate the performance of the proposed model, we use the commonly used evaluation metrics in time series prediction: mean square error (MSE) and mean absolute error (MAE) as the model evaluation metrics. The formulas for calculating MSE and MAE are as follows:

$$MSE = \frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2 \quad (15)$$

$$MAE = \frac{1}{N} \sum_{i=1}^N |Y_i - \hat{Y}_i| \quad (16)$$

Where N is the size of the dataset, Y_i is the true value of the data, and $\hat{Y}_i$ is the predicted value of the model.

The MSE metric evaluates the predictive accuracy between the model's actual

³ <http://pems.dot.ca.gov>

⁴ <https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014>

⁵ <https://www.bgc-jena.mpg.de/wetter/>

⁶ <https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html>

values and predicted values, while the MAE assesses the average error between the model's predicted values and actual values, reflecting the model's overall predictive accuracy. Therefore, the lower the MSE and MAE values, the better the trained model.

4.2.2 Baseline

To validate the superiority of TCLformer, we selected five Transformer-based models as baseline models for comparison: LogTrans[38], Informer[37], FEDformer[36], Autoformer[35], and Reformer[14]. LogTrans, Informer, and Reformer replace the self-attention mechanism of Transformer with convolutional self-attention mechanisms, sparse versions, and locally sensitive hash attention mechanisms, reducing computational complexity. FEDformer transforms the time domain into the frequency domain and achieves the best performance across all benchmarks. Autoformer is the first model to integrate a time series decomposition module into the Transformer, with its autocorrelation mechanism enabling efficient connections at the sequence level.

4.2.3 Experimental parameter setting

TCLformer is implemented using the PyTorch framework, and All experiments were conducted on an NVIDIA RTX A6000 GPU, with a server memory size of 250GB and a high-performance server running Ubuntu 18.04 as the operating system. The PyTorch[50] version is 1.9.0. The model uses a batch size of 32, an early stopping strategy, and a training cycle of 10. The mean squared error (MSE) is used as the loss function, and the Adam[51] optimizer is used for training.

4.3 Comparisons with the State of the Art

The performance of TCLformer and other models was compared across six datasets. To ensure a fair comparison, we followed the experimental setup outlined in FEDformer. For the Electricity, Weather, Exchange, and Traffic datasets, the model's input length was fixed at 96, with training and evaluation prediction lengths set to 96, 192, 336, and 720, respectively. For the ILI datasets, the input length was fixed at 36, with training and evaluation prediction lengths set to 24, 36, 48, and 60. As shown in Table 2, TCLformer achieves the best performance in terms of average MSE and MAE across all six benchmark datasets, and nearly optimal performance across all prediction lengths on all datasets.

Table 2. Multivariate long-term forecasting errors in terms of MSE and MAE, the lower the better. Among them, ILI dataset is with forecasting horizon $T \in \{24, 36, 48, 60\}$. For the others, $T \in \{96, 192, 336, 720\}$. The best results are highlighted in bold and the second one is underline.

Model		Our		LogTrans		Informer		FEDformer		Autoformer		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	92	0.337	0.381	2.116	1.197	3.755	1.525	<u>0.346</u>	<u>0.388</u>	0.358	0.397	2.626	1.317
	196	0.416	0.429	4.315	1.635	5.602	1.931	<u>0.429</u>	<u>0.439</u>	0.456	0.452	11.12	2.979
	336	0.452	0.464	1.124	1.604	4.721	1.835	0.496	0.487	<u>0.482</u>	<u>0.486</u>	9.323	2.769
	720	0.445	0.473	3.188	1.540	3.647	1.625	<u>0.463</u>	<u>0.474</u>	0.515	0.511	3.874	1.697
	Avg.	0.412	0.436	2.685	1.494	4.431	1.729	<u>0.433</u>	<u>0.447</u>	0.452	0.461	6.735	2.190

Electricity	96	0.188	0.300	0.258	0.357	0.274	0.368	<u>0.193</u>	<u>0.308</u>	0.201	0.317	0.312	0.402
	192	0.193	0.303	0.266	0.368	0.296	0.386	<u>0.201</u>	<u>0.315</u>	0.222	0.334	0.348	0.433
	336	0.201	0.313	0.280	0.380	0.300	0.394	<u>0.214</u>	<u>0.329</u>	0.231	0.338	0.350	0.433
	720	0.240	0.345	0.283	0.376	0.373	0.439	<u>0.246</u>	<u>0.355</u>	0.254	0.361	0.340	0.420
	Avg.	0.205	0.315	0.271	0.370	0.310	0.396	<u>0.213</u>	<u>0.326</u>	0.227	0.337	0.337	0.422
Weather	96	0.216	0.280	0.458	0.490	0.300	0.384	<u>0.217</u>	<u>0.296</u>	0.266	0.336	0.689	0.596
	192	0.266	0.314	0.658	0.589	0.598	0.544	<u>0.276</u>	<u>0.336</u>	0.307	0.367	0.752	0.638
	336	0.314	0.347	0.797	0.652	0.578	0.523	<u>0.339</u>	<u>0.380</u>	0.359	0.395	0.639	0.596
	720	0.391	0.391	0.869	0.675	1.509	0.741	<u>0.403</u>	<u>0.428</u>	0.419	0.428	1.130	0.792
	Avg.	0.296	0.333	0.695	0.601	0.7462	0.548	<u>0.308</u>	<u>0.360</u>	0.337	0.381	0.802	0.296
Exchange	96	<u>0.157</u>	<u>0.288</u>	0.968	0.812	0.847	0.752	0.148	0.278	0.197	0.323	1.065	0.829
	192	0.265	0.376	1.040	0.851	1.204	0.895	<u>0.271</u>	<u>0.380</u>	0.300	0.369	1.188	0.906
	336	0.450	0.492	1.659	1.081	1.672	1.036	<u>0.460</u>	<u>0.500</u>	0.509	0.524	1.357	0.976
	720	1.184	0.833	1.941	1.127	2.478	1.310	<u>1.195</u>	<u>0.841</u>	1.447	0.941	1.510	1.016
	Avg.	0.514	0.497	1.402	0.967	1.550	0.998	<u>0.518</u>	<u>0.499</u>	0.613	0.539	1.280	0.514
Traffic	96	0.553	0.339	0.684	0.384	0.719	0.391	<u>0.587</u>	<u>0.366</u>	0.613	0.388	0.732	0.423
	192	0.557	0.340	0.685	0.390	0.696	0.379	<u>0.604</u>	<u>0.373</u>	0.616	0.382	0.733	0.420
	336	0.562	0.341	0.774	0.408	0.777	0.420	<u>0.621</u>	0.383	0.622	<u>0.337</u>	0.742	0.420
	720	0.583	0.363	0.717	0.396	0.864	0.472	<u>0.626</u>	<u>0.382</u>	0.660	0.408	0.755	0.423
	Avg.	0.563	0.345	0.715	0.394	0.764	0.415	<u>0.609</u>	<u>0.376</u>	0.627	0.378	0.740	0.580
ILI	24	2.948	1.105	4.480	1.444	5.764	1.677	<u>3.228</u>	<u>1.260</u>	3.483	1.287	4.400	1.382
	36	2.549	0.968	4.799	1.467	4.755	1.467	<u>2.697</u>	<u>1.080</u>	3.103	1.148	4.783	1.448
	48	2.592	0.990	4.800	1.468	4.763	1.469	<u>2.622</u>	<u>1.078</u>	2.669	1.085	4.832	1.465
	60	2.751	1.040	5.278	1.560	5.264	1.564	2.857	1.157	<u>2.770</u>	<u>1.125</u>	4.882	1.483
	Avg.	2.710	1.025	4.839	1.484	5.136	1.544	<u>2.851</u>	<u>1.143</u>	3.006	1.161	4.724	2.71

-Other results are from FEDformer[36].

On the ETTh2 dataset, compared to FEDformer, TCLformer achieved the most significant performance improvement (8.87%) at a prediction length of 336, with an average MSE performance improvement of 4.85% across four prediction lengths. On the Electricity dataset, when the prediction length is 336, TCLformer achieves a performance improvement of 6.07% compared to FEDformer, with an average MSE performance improvement of 3.76% across the four prediction lengths. On the Weather dataset, the most significant performance improvement is achieved when the prediction length reaches 336, at 7.37%, with an average performance improvement of 3.9%. On the Exchange dataset, the maximum performance improvement (2.22%) was achieved when the prediction length was 192, with an average performance improvement of 0.77%. The Exchange dataset did not exhibit obvious periodicity, but TCLformer still achieved better performance than FEDformer, demonstrating the effectiveness of our model. On the Traffic dataset, TCLformer achieved the maximum performance improvement (9.5%) when the prediction length was 336, with an average performance improvement of 7.55%. On the ILI dataset, TCLformer achieved the maximum performance improvement (8.67%) at a prediction length of 24, with an average performance improvement of 4.95%.

4.4 Visual analysis of model predictions

In Fig. 4 and Fig. 5, we compare the predicted values (336 and 720 steps) generated by TCLformer and FEDformer on the ECL dataset with the true values. The blue line represents the true values, and the orange line represents the predicted values. As observed in the sequence segments highlighted in the red boxes, our model outperforms FEDformer in both long-term temporal patterns and local dynamics when fitting true values and predicted values, demonstrating the effectiveness and superiority of our model.

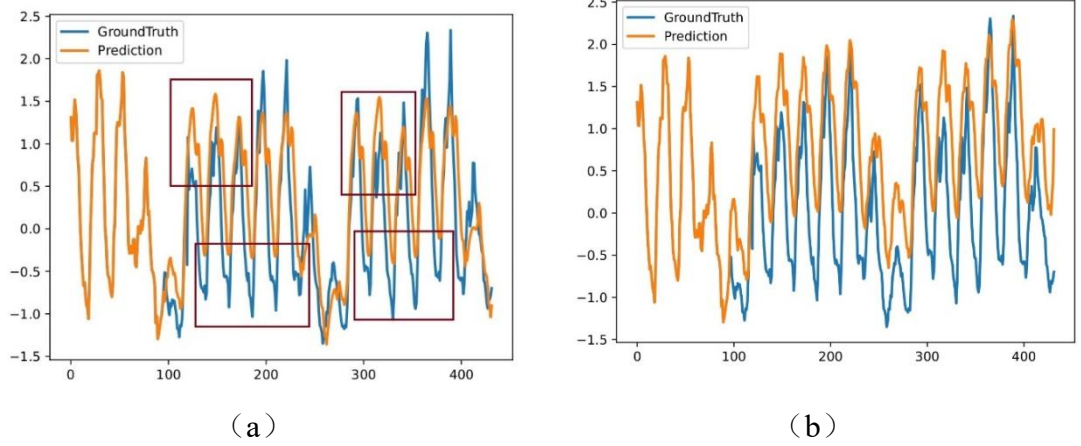


Fig. 4 Comparison of prediction results and ground-truth values of the TCLformer and FEDformer on the Electricity dataset. (a) TCLformer with prediction length of 336; (b) FEDformer with prediction length of 336.

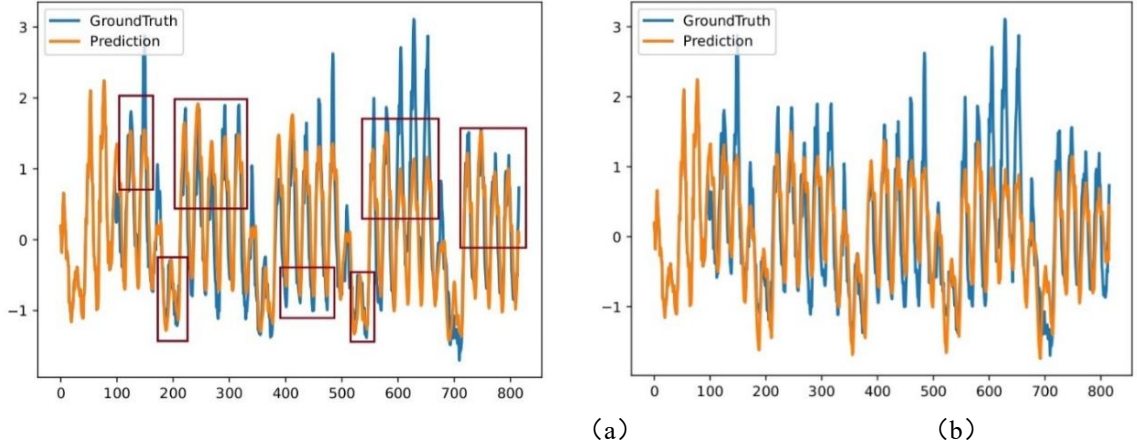


Fig. 5 Comparison of prediction results and ground-truth values of the TCLformer and FEDformer on the Electricity dataset. (a) TCLformer with prediction length of 720; (b) FEDformer with prediction length of 720.

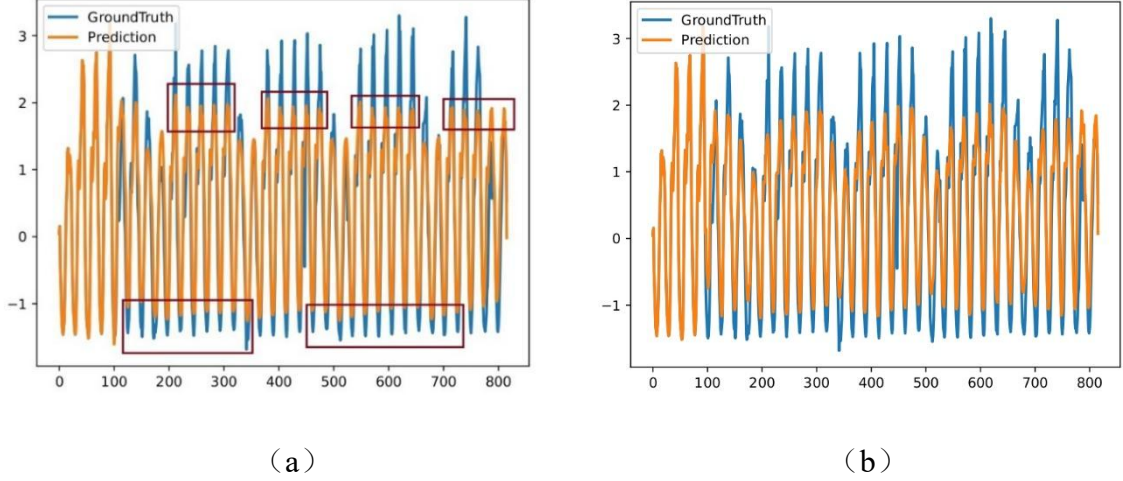


Fig. 6 Comparison of prediction results and ground-truth values of the TCLformer and FEDformer on the Electricity dataset. (a) TCLformer with prediction length of 720; (b) FEDformer with prediction length of 720.

Fig. 6 shows a comparison between TCLformer and FEDformer in predicting 720 steps of true values on the Traffic dataset. As can be seen from the figure, when predicting a length of 720, TCLformer better captures the complex temporal dependencies in the data, especially in terms of local dynamic changes. We believe that this is because the TCN module effectively captures short-term dependencies.

4.5 Ablation studies

To validate the effectiveness of the proposed TCN in the decoder, we conducted an ablation study by replacing TCN with several well-known attention mechanisms, including Fourier Attention, AutoCorrelation, and ProbSparse Attention, on the Weather and Electricity datasets. The experimental results are shown in Table 3

On the Weather dataset, the model with TCN achieves the best performance with an average MSE of 0.296, while replacing it with FourierAttention, AutoCorrelation, and ProbAttention results in increased MSEs of 0.319, 0.301, and 0.298, respectively. These represent 1.7%、7.8% and 17.6% relative performance drops, indicating that TCN contributes significantly to capturing temporal dependencies in meteorological data. Similarly, on the Electricity dataset, TCN again demonstrates superior performance with the lowest MSE of 0.205, outperforming, FourierAttention (0.235), AutoCorrelation (0.233), and ProbAttention (0.246). The respective increases of 14.6%, 13.6%, and 20% in MSE further validate the robustness and generalization ability of the proposed TCN module across domains.

These results confirm that the TCN module, through its multi-scale convolutional structure, enhances temporal feature extraction and outperforms conventional attention mechanisms in both accuracy and efficiency for multivariate time series forecasting.

Table 3. Ablation studies of TCN module on Weather and Electricity dataset.

Model		TCN		FourierAttention		AutoCorrelation		ProbAttention	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.216	0.280	0.243	0.319	0.223	0.291	<u>0.222</u>	<u>0.289</u>
	192	0.266	0.314	0.290	0.347	0.274	0.327	<u>0.268</u>	<u>0.322</u>

	336	0.314	0.347	0.339	0.380	0.318	<u>0.349</u>	<u>0.316</u>	0.347
	720	<u>0.391</u>	0.391	0.406	0.418	<u>0.391</u>	0.391	0.388	0.391
	Avg.	0.296	0.333	0.319	0.366	0.301	0.339	<u>0.298</u>	<u>0.337</u>
Electricity	96	0.188	0.300	<u>0.212</u>	<u>0.320</u>	0.218	0.328	0.221	0.334
	192	0.193	0.303	<u>0.224</u>	<u>0.331</u>	0.236	0.344	0.237	0.349
	336	0.201	0.313	0.235	0.340	<u>0.226</u>	<u>0.336</u>	0.248	0.357
	720	0.240	0.345	0.269	0.365	<u>0.253</u>	<u>0.358</u>	0.278	0.377
	Avg.	0.205	0.315	0.235	<u>0.339</u>	<u>0.233</u>	0.341	0.246	0.354

To demonstrate the effectiveness of the proposed Convolutional Attention mechanism in the decoder, we replace it with several popular attention mechanisms including Fourier Attention, ProbSparse Attention, AutoCorrelation, and Full Attention, respectively, and conducted comparative experiments on the Weather and Electricity benchmark datasets. The experimental results are shown in Table 3.

On the Weather dataset, ConvAttention achieves the lowest MSE of 0.296, outperforming all other attention variants by a significant margin. Specifically, replacing it with ProbAttention, FourierAttention, AutoCorrelation, and FullAttention results in 17.6%, 23%, 28.4%, and 45.9% relative increases in MSE respectively, highlighting the superiority of the proposed convolutional attention in modeling complex temporal patterns. On the Electricity dataset, ConvAttention also delivers the best performance with an MSE of 0.205. While the performance gap is narrower in this dataset, ConvAttention still consistently outperforms the other attention mechanisms, indicating its robustness across different domains.

These results confirm that the proposed convolutional attention, by leveraging local context through causal convolution, captures both short- and long-range dependencies more effectively and efficiently compared to standard attention variants.

Table 4. Ablation studies of the ConvAttention module on Weather and Electricity dataset.

Model		ConvAttention		ProbAttention		FourierAttention		AutoCorrelation		FullAttention	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.216	0.280	0.234	0.310	<u>0.224</u>	<u>0.303</u>	0.226	0.307	0.304	0.376
	192	0.266	0.314	0.306	0.367	0.343	0.405	0.458	0.496	<u>0.300</u>	<u>0.362</u>
	336	0.314	0.347	<u>0.342</u>	<u>0.393</u>	0.410	0.452	0.391	0.426	0.382	0.427
	720	0.391	0.391	0.513	0.494	0.482	0.488	<u>0.447</u>	<u>0.455</u>	0.744	0.639
	Avg.	0.296	0.333	<u>0.348</u>	<u>0.391</u>	0.364	0.412	0.380	0.421	0.432	0.451
Electricity	96	0.188	0.300	0.202	0.317	<u>0.192</u>	0.304	0.199	<u>0.309</u>	0.196	0.310
	192	0.193	0.303	0.200	0.316	0.207	0.320	0.204	<u>0.315</u>	0.204	0.316
	336	0.201	0.313	<u>0.211</u>	0.329	0.219	0.330	0.219	<u>0.325</u>	0.221	0.331
	720	0.240	0.345	0.221	0.334	0.239	0.349	0.256	0.358	0.234	0.343
	Avg.	0.205	0.315	<u>0.208</u>	<u>0.324</u>	0.214	0.325	0.219	0.326	0.213	0.325

5. Discussion

To address the limitation of Transformer's self-attention operation, which exhibits quadratic growth with respect to the input sequence length, some models have been improved to reduce computational costs, including LogTrans, Informer, and Reformer.

As shown in Table 2, these models still lack global modeling capabilities and perform poorly in long-term predictions. Autoformer decomposes sequences into multiple temporal sub-sequences for feature extraction, with its self-correlation mechanism focusing on sub-sequence similarity to extract periodic patterns. However, a simple moving average of the input signal is insufficient to model complex trend patterns. FEDformer builds upon the Autoformer architecture by using frequency transformation to decompose the sequence into multiple frequency-domain modes for feature extraction, selecting a subset of frequency domains randomly rather than retaining only low-frequency components. However, the random selection method may fail to effectively capture multi-scale features. TCLformer adopts the Autoformer time series decomposition framework, separating seasonal and trend components. Multi-scale variable convolutions can perceive temporal local patterns at different scales, enabling adaptive modeling of multi-scale features. Subsequently, a lower-complexity convolutional attention mechanism is used to capture long-term complex temporal patterns. Ablation experiments also demonstrate the effectiveness of the TCN module and convolutional attention mechanism module. We conducted extensive experiments on six different real-world datasets. TCLformer achieved better performance than FEDformer on the Exchange and ILI datasets, which lack obvious periodicity, demonstrating the effectiveness and superiority of the proposed model.

6. Conclusion

This paper proposes a deep learning model for multi-scale time series forecasting, TCLformer, to address the issue of insufficient modeling capabilities for local and global dependencies in long-term forecasting of multi-variable time series. The model integrates seasonal-trend decomposition mechanisms, multi-scale adaptive convolutions, and sparse convolutional attention mechanisms. Through the time series decomposition module, it effectively unravels the complex intertwined patterns in time series, and utilizes convolutional structures to efficiently model the interactions between local features and global dynamics. Experimental results on six publicly available real-world datasets demonstrate that TCLformer outperforms existing state-of-the-art methods across various metrics. On the Traffic dataset, the model achieved a performance improvement of up to 9.5%, with a relative improvement of up to 8.87% in terms of average prediction error. Additionally, on non-periodic datasets (such as Exchange and ILI), TCLformer demonstrated superior predictive performance compared to the advanced FEDformer model, validating its strong generalization capability and modeling potential in non-periodic, complex time series environments.

Future work will focus on further exploring the modeling of cross-dimensional dependencies in multi-dimensional time series and enhancing the model's robustness and generalization performance in data-sparse challenging scenarios to enhance the practical value of TCLformer in real-world applications.

Declaration of competing interest

The authors declare that they have to know competing financial interests or personal relationships that could have appeared to influence the work reported in this

paper.

Acknowledgments

This research has been supported by the Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX24_0724). The authors are grateful to anonymous reviewers whose constructive and valuable comments greatly helped us to improve the paper.

Reference

- [1] Rubhasy A, Andrianingsih, Rayyan (2023) Abc-optimized deep learning approach to rainfall forecasting in serang city. In: 2023 Eighth International Conference on Informatics and Computing, pp 1–6
- [2] Li Z, Zhou J, Lin Z, Zhou T (2024) Dynamic spatial aware graph transformer for spatiotemporal traffic flow forecasting. *Knowledge-based systems* 297:111946
- [3] Ma X, Wang J, Huang J, Ke Y (2025) Energy consumption and carbon emission modeling and forecasting study with novel deep learning methods. *Expert Systems with Applications* 290:128314
- [4] Antoniadis A, Gaucher S, Goude Y (2024) Hierarchical transfer learning with applications to electricity load forecasting. *International Journal of Forecasting* 40(2):641-60
- [5] Wang X, Zhang X, Liu J (2025) MA-MSTNet: mixed attention-based multi-scale temporal network for stock trend prediction. *J Supercomput* 81, 1207. <https://doi.org/10.1007/s11227-025-07700-1>
- [6] Khashei M, Bijari M, Hejazi S R (2012) Combining seasonal ARIMA models with computational intelligence techniques for time series forecasting. *Soft Computing* 16:1091-1105
- [7] Kaur J, Parmar K S, Singh S (2023) Autoregressive models in environmental forecasting time series: A theoretical and application review. *Environmental Science and Pollution Research* 30(8): 19617-19641
- [8] Wang X, Liu H, Yang Z, Du J, Dong X (2023) CNformer: A Convolutional Transformer with Decomposition for Long-Term Multivariate Time Series Forecasting. *Appl Intell* 53 (17):20191–20205. <https://doi.org/10.1007/s10489-023-04496-6>
- [9] Torres J F, Hadjout D, Sebaa A, Martínez-Álvarez F, Troncoso A (2021) Deep learning for time series forecasting: a survey. *Big data* 9(1):3-21
- [10] Kong Y, Wang Z, Nie Y, Zhou T, Zohren S, Liang (2025) Unlocking the power of lstm for long term time series forecasting. In: *Proceedings of the AAAI Conference on Artificial Intelligence* pp 11968-11976
- [11] Kirisci M, Cagcag Yolcu O (2022) A New CNN-Based Model for Financial Time Series: TAIEX and FTSE Stocks Forecasting. *Neural Processing Letters* 54(4): 3357–3374
- [12] Song X, Deng L, Wang H (2024) Deep learning-based time series forecasting. *Artificial Intelligence Review* 58(1):23
- [13] Vaswani A, Shazeer N M, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L,

- Polosukhin I (2017) Attention is all you need. *Advances in Neural Information Processing Systems*
- [14] Kitaev N, Kaiser Ł, Levskaya A (2020) Reformer: The Efficient Transformer. *ArXiv*, abs/2001.04451
- [15] Wen Q, Zhou T, Zhang C, Chen W, Ma Z, Yan J, Sun L (2023) Transformers in Time Series: A Survey. In: *International Joint Conference on Artificial Intelligence*, pp 6778–6786. <https://doi.org/10.24963/ijcai.2023/759>
- [16] Salinas D, Flunkert V, Gasthaus J, Januschowski T (2020) DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36(3):1181–1191. <https://doi.org/10.1016/j.ijforecast.2019.07.001>
- [17] Bai Shao J, Kolter J Z, Koltun Vladlen (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*
- [18] Sen R, Yu H F, Dhillon I S (2019) Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in Neural Information Processing Systems*
- [19] Song W, Fujimura S (2021) Capturing combination patterns of long and short-term dependencies in multivariate time series forecasting. *Neuro computing* 464:72–82
- [20] Lai G K, Chang W C, Yang Y M, Liu H X (2018) Modeling long- and short-term temporal patterns with deep neural networks. In: *International ACM SIGIR Conference on Research & Development in Information Retrieval*
- [21] Huang S, Wang D, Wu X, Tang A (2019) DSANet: Dual self-attention network for multivariate time series forecasting. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp 2129–2132
- [22] Liu M, Zeng A, Chen M, Xu Z, Lai Q, Ma L, Xu Q (2022) SCINet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828
- [23] Wu H, Hu T, Liu Y, Zhou H, Wang J, Long M (2022) TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. *International Conference on Learning Representations*. <https://doi.org/10.48550/arxiv.2210.02186>
- [24] Zeng A, Chen M, Zhang L, Xu Q (2023) Are transformers effective for time series forecasting?. In: *Proceedings of the AAAI conference on artificial intelligence* 37:11121–11128
- [25] Li Z, Qi S Y, Li Y D, Xu Z L (2023) Revisiting Long-Term Time Series Forecasting: An Investigation on Linear Mapping. <https://arxiv.org/abs/2305.10721>
- [26] Wang S Y, Wu H X, Shi X M, Hu T G, Luo H K, Ma L T, Zhang James Y, Zhou J (2024) TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. <https://doi.org/10.48550/arXiv.2405.14616>
- [27] Li Z, Rao Z, Pan L, Xu Z (2023) MTS-Mixers: Multivariate time series forecasting via factorized temporal and channel mixing. *arxiv preprint arxiv:2302.04501*
- [28] Tolstikhin I O, Houlsby N, Kolesnikov A, Beyer L, Zhai X, Unterthiner T, Yung J, Steiner A, Keysers D, Uszkoreit J, Lucic M (2021) MLP-Mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*,

- [29]Hugo T, Bojanowski P, Caron M, Cord M, El-Nouby A, Grave E, Izacard G (2022) ResMLP: Feedforward networks for image classification with data-efficient training. *IEEE transactions on pattern analysis and machine intelligence* 45(4): 5314-5321
- [30]Chen S A, Li C L, Yoder N, Arik S O, Pfister T (2023) TSMixer: An all-mlp architecture for time series forecasting. *arxiv preprint arxiv:2303.06053*
- [31]Zhang T, Zhang Y, Cao W, Bian J, Yi X, Zheng S, Li J (2022) Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. *arxiv preprint arxiv:2207.01186*
- [32]Devlin J, Chang M W, Lee K, Toutanova K (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Annual Conference of the North American Chapter of the Association for Computational Linguistics*
- [33]Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X H, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N (2021) An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*
- [34]Dong L H, Xu S, Xu B (2018) Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*
- [35]Wu H, Xu J, Wang J, Long M (2021) Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Advances in Neural Information Processing Systems*, pp 22419–22430
- [36]Zhou T, Ma Z, Wen Q, Wang X, Sun L, Jin R (2022) FEDformer: Frequency Enhanced Decomposed Transformer for Long-Term Series Forecasting. In: *International Conference on Machine Learning*
- [37]Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021) Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *AAAI Conference on Artificial Intelligence* 35:11106–11115. <https://doi.org/10.1609/aaai.v35i12.17325>.
- [38]Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang Y X, Yan X (2019) Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *Advances in Neural Information Processing Systems*
- [39]Liu S, Yu H, Liao C, Li J, Lin W, Liu A X, Dustdar S (2022) Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. In: *International Conference on Learning Representations*
- [40]Zhang X, Jin X, Gopalswamy K, Gupta G, Park Y, Shi X, Wang H, Maddix D C, Wang Y (2022) First De-Trend then Attend: Rethinking Attention for Time-Series Forecasting. *arxiv preprint arxiv:2212.08151*
- [41]Piao X, Chen Z, Murayama T (2024) Fredformer: Frequency Debaised Transformer for Time Series Forecasting. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp 2400–2410
- [42]Xiao B, Ge Z, Zhang X (2025) FCAFormer: multivariate time series forecasting combining channel attention and transformer in the frequency domain . *The Journal*

of Supercomputing 81(11)

- [43] Yue, W Z, Liu Y, Ying X H, Xing B W, Guo R H, and Shi J (2025) Freeformer: Frequency enhanced transformer for multivariate time series forecasting. arxiv preprint arxiv:2501.13989
- [44] Lim B, Arik SÖ, Loeff N, Pfister T (2021) Temporal fusion transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting 37(4):1748-64
- [45] Nie Y, Nguyen N H, Sinthong P, Kalagnanam J (2022) A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. International Conference on Learning Representations. [arxiv preprint arxiv:2211.14730](#)
- [46] Zhang Y H, Yan J C (2023) Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: The eleventh international conference on Learning Representations
- [47] Wang H, Peng J, Huang F, Wang J, Chen J, Xiao Y (2023) MICN: Multi-scale local and global context modeling for long-term series forecasting. In: The eleventh international conference on Learning Representations
- [48] Shabani A, Abdi A, Meng L, Sylvain T (2022) Scaleformer: Iterative Multi-scale Refining Transformers for Time Series Forecasting. In: International Conference on Learning Representations. arxiv preprint arxiv:2206.04038
- [49] Woo G, Liu C H, Sahoo D, Kumar A, Hoi S (2022) ETSformer: Exponential Smoothing Transformers for Time-Series Forecasting. arxiv preprint arxiv:2202.01381
- [50] Paszke A (2019) Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703
- [51] Kingma D P, Ba J (2017) Adam: A Method for Stochastic Optimization. arxiv preprint arxiv:1412.6980