Multivariate Time Series Modeling and Forecasting with Parallelized Convolution and Decomposed Sparse-Transformer

Shusen Ma, Yun-Bo Zhao*, *Senior Member, IEEE*, Yu Kang, *Senior Member, IEEE*, and Peng Bai

***Abstract*—Many real-world scenarios require accurate predictions of time series, especially in the case of long sequence time-series forecasting (LSTF), such as predicting traffic flow and electricity consumption. However, existing time series prediction models encounter certain limitations. Firstly, they struggle with mapping the multidimensional information present in each time step to high dimensions, resulting in information coupling and increased prediction difficulty. Secondly, these models fail to effectively decompose the intertwined temporal patterns within the time series, which hinders their ability to learn more predictable features. To overcome these challenges, we propose a novel end-to-end LSTF model with parallelized convolution and decomposed sparse-Transformer (PCDformer). PCDformer achieves the decoupling of input sequences by parallelizing the convolutional layers, enabling the simultaneous processing of different variables within the input sequence. To decompose distinct temporal patterns, PCDformer incorporates a temporal decomposition module within the encoder-decoder structure, effectively separating the input sequence into predictable seasonal and trend components. Additionally, to capture the correlation between variables and mitigate the impact of irrelevant information, PCDformer utilizes a sparse self-attention mechanism. Extensive experimentation conducted on five diverse datasets demonstrates the superior performance of PCDformer in LSTF tasks compared to existing approaches, particularly outperforming encoder-decoder-based models.**

***Impact Statement*—LSTF is a popular topic in time-series forecasting, which can predict the values of variables over a long period. It enables us to make decisions according to the forecasting results, which plays a significant role in various fields. However, existing models struggle with information coupling and intertwined temporal patterns, which might undermine prediction accuracy. This article proposes a new end-to-end model, PCDformer, which simultaneously considers the above problems and provides a new idea to improve the accuracy of time-series forecasting. PCDformer provides a separate convolutional layer for each variable to learn temporal dependence, avoiding the impact of information coupling. Besides, PCDformer deploys series decomposition and sparse self-attention to the canonical Transformer to disentangle series and extract the correlation between variables. The experimental results illustrate the superiority of the proposed model compared with the state-of-the-art methods, covering traffic, energy, and weather prediction.**

***Index Terms*—Decomposed sparse-Transformer, long sequence time-series forecasting, parallelized convolution**

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).

Shusen Ma is with the Institute of Advanced Technology, USTC, Shushan District, Hefei, 230031, Anhui, China.

Yun-Bo Zhao* (corresponding author) is with the Institute of Advanced Technology, USTC, Shushan District, Hefei, 230031, Anhui, China, the Department of Automation, USTC, Shushan District, Hefei, 230031, Anhui, China, and also with the Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Shushan District, Hefei, 230031, Anhui, China (e-mail: ybzhao@ustc.edu.cn).

Yu Kang is with the Institute of Advanced Technology, USTC, Shushan District, Hefei, 230031, Anhui, China, the Department of Automation, USTC, Shushan District, Hefei, 230031, Anhui, China, and also with the Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Shushan District, Hefei, 230031, Anhui, China.

Peng Bai is with the Department of Automation, USTC, Shushan District, Hefei, 230031, Anhui, China.

I. Introduction

TIME series forecasting, especially the long sequence time-series forecasting (LSTF), is extensively utilized for predicting energy consumption, economic fluctuations, weather changes, and more [1]–[7]. It enables better long-term planning, which can minimize risks or maximize benefits. Currently, time series forecasting methods can be broadly categorized into two types: supervised end-to-end learning methods [8]–[12] and representation learning methods [13]–[17]. The end-to-end learning method involves the extraction of features through the stacking of various nonlinear layers. These extracted features are then fed into a regression layer for prediction [9]. Transformer-based models [8], [9], [12], [18]–[21] have shown promising results within this category. Representation learning methods [13]–[17], on the other hand, primarily focus on learning feature representations from observed data to improve prediction performance. These methods aim to capture meaningful patterns and relationships within the data, leading to more accurate forecasts.

However, multivariate long-term forecasting tasks still present certain challenges. Firstly, it is not proper to regard the multivariate information of each time step as a token and then map it to a high-dimensional space by linear layer. It is well known that each time step contains diverse variables' state information and each variable has its unique characteristics. If we adopt the above mapping approach, models may overlook learning these characteristics, and irrelevant variables may also interfere with the prediction of the target variable, causing information coupling and making the model prone to overfitting and more sensitive to noise in the long sequence. Secondly, there are different entangled temporal patterns in the time series [14], [22], [23], such as trend part and seasonal part. Moreover, each variable has different temporal patterns, which makes it difficult to extract the temporal dependence between different variables and the temporal features of each variable directly from the complex long series. Additionally, the presence of irrelevant temporal dependence information between variables can have a detrimental impact on the accuracy of predictions.

To solve the above problems, we propose a new end-to-end LSTF model with parallelized convolution and decomposed sparse-Transformer (PCDformer). For the first problem, PCDformer tries to decouple multivariate time series [24] by parallelizing different variables of the input sequence [25]. Then it extracts the temporal dependencies of the variables separately [25] through the proposed convolutional module that consists of parallelized convolution layers. To extract more potential temporal dependencies, we convert the one-dimensional input sequence into a two-dimensional image-like data [7] and extract the temporal features by the proposed M-inception layer based on Inception [26]. For the second problem, PCDformer embeds series decomposition modules in the encoder-decoder structure to disentangle temporal patterns and then improves self-attention to sparse self-attention because of the redundant information [27]. The series decomposition module can decompose complex series into trend and seasonal parts [22] so that the model can learn more predictable features. Considering the negative interaction between variables, we propose sparse self-attention to eliminate the impact of irrelevant feature information on the prediction of the target variable and reduce the computation of the model. Not all time series are predictable, so this article focuses on data with relatively obvious trends and seasonal characteristics. PCDformer achieves state-of-the-art accuracy on multiple datasets. The main contributions of this work are summarized below:

- A new end-to-end model called PCDformer is proposed for LSTF tasks, which outperforms currently state-of-the-art methods on five datasets, covering energy, traffic, and weather.
- We parallelize the input sequences of different variables and efficiently extract the corresponding temporal dependencies through the proposed parallel processing layer, which can reduce the difficulty of temporal feature extraction caused by information coupling.
- A sparse self-attention mechanism and series decomposition modules are applied to the canonical Transformer to extract disentangled temporal patterns. The sparse self-attention mechanism enables the PCDformer to focus on the most contributive features. The series decomposition module could disentangle the temporal patterns of the input series, making the model learn useful temporal features.

II. Related Work

With the increasing adoption of LSTF in real-world scenarios [28]–[30], there has been a surge in the development of diverse models tailored for LSTF tasks. Conventional forecasting techniques like autoregressive [31] and autoregressive integrated moving average [32] models are primarily utilized for straightforward forecasting tasks involving univariate data, lacking the capability to effectively handle complex multivariate data. Due to the development of deep learning and the improvement of computing power, a large number of models based on deep learning are used in various time series prediction tasks, although they may be vulnerable to imperceptible perturbations [33]. The earliest models were based on the internal memory states of recurrent neural networks (RNNs) to remember dependencies between different time steps. However, due to the problem of vanishing gradients or explosions, RNNs-based models [34] cannot remember the temporal dependence of long-term sequences. To improve this problem, variants such as gated recurrent units [34] and long short-term memory [35] have been proposed. These variants enhance the model's ability to process long sequence inputs by adding gating units to let the model selectively memorize and forget information. VAE-GRU [11] utilizes the strengths of RNNs and Stochastic Gradient Variational Bayes, which can take advantage of the unlabeled data to promote supervised learning of RNNs. With the advent of convolutional neural networks (CNNs), many researchers have applied them to extract short-term temporal dependencies of series and local correlations between variables. TCN [10] introduces a new convolutional structure based on CNNs, which increases the receptive fields of the canonical convolution so that the long-term dependence of the series can be better captured.

Since the self-attention mechanism in the Transformer [36] learns global information well, many Transformer-based models are applied to the LSTF tasks. Here is a summary of late typical Transformer-based models. Reformer [18] substitutes locality-sensitive hashing attention for dot-product attention, which improves the efficiency of the Transformer. LogTrans [19] proposes convolutional self-attention with LogSparse design, making local context better incorporated into the attention mechanism and reducing space complexity. Informer [9] puts forward ProbSparse self-attention mechanism and self-attention distilling to extract the most vital keys and decrease the counting amount. Autoformer [8] proposes the decomposition architecture to decompose more predictable components from complex temporal patterns and the auto-correlation mechanism to implement series-wise connections. Pyraformer [20] introduces the pyramidal attention module in which inter-scale tree architecture extracts diverse resolutions' features and intra-scale neighboring connections capture various ranges' temporal dependencies. FEDformer [21] reduces the distribution difference between input and output by seasonal-trend decomposition and applies an attention mechanism in the frequency domain to increase robustness to noise. CLformer [12] deploys dilated convolutional networks to extract temporal patterns and proposes a local group auto-correlation mechanism to capture the dependencies at multiple scales.

Besides supervised learning, self-supervised learning has also been demonstrated to be efficient for downstream tasks by learning useful representations. In the time-series representations domain, there are many relevant works. TNC [16] utilizes the local smoothness of a signal's generative process to define neighborhoods and learns time series representations with a debiased contrastive objective. In a hierarchical way over augmented context views, TS2Vec [15] fulfills contrastive learning and enables a robust contextual representation for each timestamp. CoST [13] utilizes inductive biases within the model to obtain disentangled seasonal and trend feature

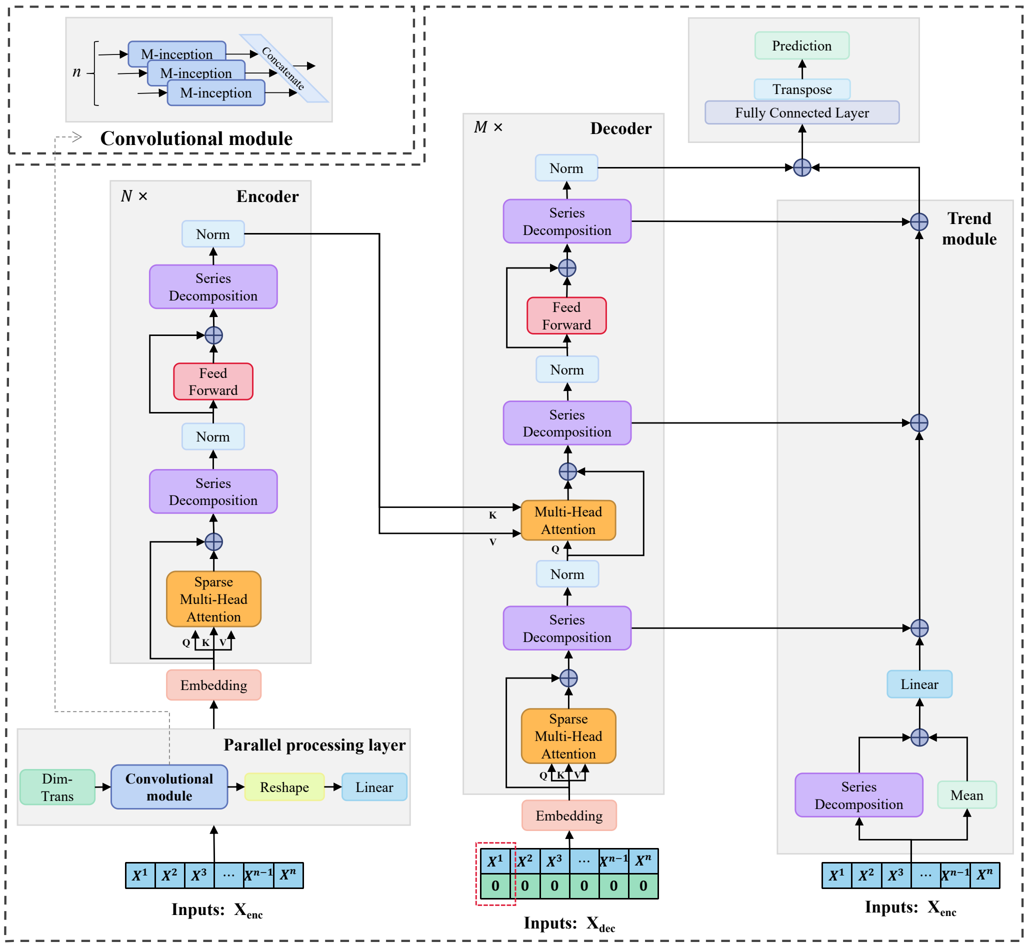


Fig. 1. PCDformer architecture.

representations. LaST [14] combines representation learning and series decomposition to learn the disentangled seasonal and trend characterization of the long sequence. SimTS [17], a contrastive representation learning method, employs a siamese structure and a simple convolutional encoder to learn features and does not depend on negative pairs. ACST [37] generates samples by an improved cycle generative adversarial data augmentation method and cuts down on the impact of noise and feature variable weights through the gated residual networks and a noise decomposition module.

III. Preliminary

Given a specific dataset, expressed as $\mathbf{X} \in \mathbb{R}^{l\times n}$ where l represents the length of the total sequence and n denotes the number of variables, and a look-back window of fixed length L. At the moment t, the task of LSTF is to forecast $\widetilde{\mathbf{X}}_{t+1:t+\tau} = \{\mathbf{x}_{t+1}, ..., \mathbf{x}_{t+\tau}\}$ taking the historical observation $\mathbf{X}_{t-L+1:t} = \{\mathbf{x}_{t-L+1}, ..., \mathbf{x}_t\}$ into account, where τ indicates the length of the prediction, called horizon, and $\mathbf{x_t} \in \mathbb{R}^n$.

When training and inference, the existing study usually takes each time step as a token, while entering consecutive L tokens, and then extracts the temporal dependency between each token. In contrast, this paper regards the values of each variable for consecutive L moments as a token, first independently learns the temporal dependence of each token, then extracts the correlation between each token and eliminates redundant feature information. The specific input and output descriptions of the model have been given in Section IV-A.

IV. Methodology

We propose a PCDformer model, shown in Fig. 1, based on Transformer and Informer's generative style encoder-decoder structure, mainly consisting of a parallel processing layer, Encoder-Decoder, and trend module. The parallel processing layer comprises a dimension transformation (Dim-Trans) layer, a Convolutional module, a Reshape layer, and a Linear layer. It is designed to capture diverse variables' temporal features independently. The Encoder mainly consists of the sparse Multi-Head Attention and the Series Decomposition modules, which can eliminate the impact of irrelevant information between variables and be conducive to the learning of predictable features, such as the trend and season characteristics. The Decoder is utilized to capture the relationship between historical and future information, possessing similar components to the Encoder. The design of the trend module is for the complete learning of the trend part. In subsequent sections, we first describe the input and output of the model, and then introduce the details of these components.

A. The descriptions of the input and output

Currently, Transformer-based LSTF models employ the mixed-channel approach for input [9]. In this method, the multivariate information at each time step of the input sequence will be regarded as a token projected into a high-dimensional space, mixing the multivariate information. For example, the input $X \in \mathbb{R}^{L\times n}$ will be mapped to $\hat{X} \in \mathbb{R}^{L\times d_{\text{model}}}$ after passing the linear layer whose output dimension is d_{model}. However, this approach may cause information coupling because each element in $\hat{X}$ is a fusion of features from all variables, rendering the model vulnerable to interference from irrelevant features and impacting the predictive accuracy of the model. In contrast, this paper will adopt the parallelized-channel input method, taking the information contained in each variable of the input sequence as a separate token.

Specifically, the input to the parallel processing layer and the trend module can be expressed as $\mathbf{X}_{enc} \in \mathbb{R}^{n\times L}$. Unlike Informer, the decoder input of PCDformer takes into account all the observed historical information. It can be expressed as $\mathbf{X}_{dec} \in \mathbb{R}^{n\times(L+\tau)}$, where τ represents the length of the prediction. Since we cannot know the future information in advance, the position to be predicted is initialized to 0. For example, the token inside the red dotted box in Fig. 1 can be represented as Fig. 2. The sum of the output of the decoder and the trend module, $\mathbf{Y} \in \mathbb{R}^{n\times d_{\text{model}}}$, becomes $\hat{\mathbf{Y}} \in \mathbb{R}^{n\times\tau}$ after passing the Fully Connected Layer. $\hat{\mathbf{Y}}$ then passes through the Transpose layer, obtaining the final predicted value, $\widetilde{\mathbf{Y}} \in \mathbb{R}^{\tau\times n}$.

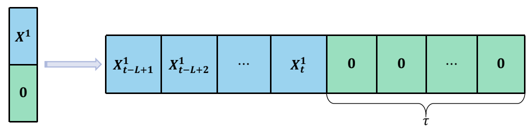


Fig. 2. The structure of one token of the decoder input.

B. Parallel processing layer

Existing works often use convolution to extract temporal features [38], [39], which cannot effectively perceive global and long-term information because of the limited receptive field. Inspired by the former work [7], we first insert the Dim-Trans layer into the parallel processing layer to convert each token of the input sequence into an image form. The process is shown in Fig. 3, where the selection of P is shown as follows:

$$P = \begin{cases} \sqrt{L}, & \sqrt{L} \in \mathbb{N}^* \\ \lceil\sqrt{L}\rceil, & \sqrt{L} \notin \mathbb{N}^*. \end{cases} \tag{1}$$

Then we extract the local and global features of each token through two-dimensional convolution. Considering that too deep a network can cause model overfitting or degradation, we take the idea of the Inception network and establish a module called M-inception, shown in Fig. 4, to extract the temporal dependencies of each token. Each convolutional layer of M-inception takes a different convolution kernel and the number in parentheses indicates the number of convolution kernels and the size of padding. The output results of different convolutional layers are summed, and the feature fusion values are averaged after the Relu activation function to obtain the time series features contained in each token.

The convolutional module shown in Fig. 1 contains n parallelized M-inception layers. It denotes that different variables will be handled independently by different M-inception modules, which can avoid the problem of information coupling. The output of all M-inception layers will be concatenated. To conform to the dimensional need of the subsequent algorithm, we reshape the last two dimensions of the output result of the convolutional module to one dimension and then map it to high-dimensional space through the linear layer, whose representation dimension can be expressed as d_{model}.

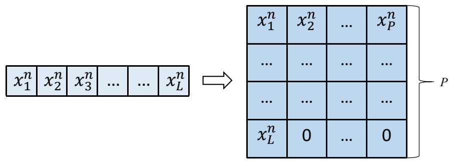


Fig. 3. The illustration of the Dim-Trans layer. We take the token n of $\mathbf{X}_{enc} \in \mathbb{R}^{n\times L}$ as an example to illustrate the process of dimension transformation. When $\sqrt{L} \notin \mathbb{N}^*$, the remaining blank spaces are filled with 0.

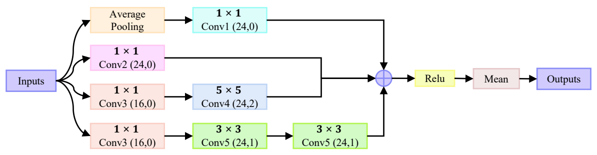


Fig. 4. The structure of the M-inception. The Mean layer indicates the operation of averaging.

Specifically, the dimension format of the parallel processing layer's input, $\mathbb{R}^{n\times L}$, becomes $\mathbb{R}^{n\times P\times P}$ after going through Dim-Trans layer. Then the convolutional module extracts the temporal features while keeping the overall dimension unchanged. The features' dimension will be reshaped to $\mathbb{R}^{n\times L}$ by the Reshape layer and the reshaped result will be mapped to $\mathbb{R}^{n\times d_{\text{model}}}$ by the linear layer.

C. Series decomposition

Due to the entangled temporal patterns in time series, it is difficult to learn the temporal dependence of variables directly from long input sequences. The survey by Cleveland et al. [22] offers a significantly versatile and robust time-series decomposition method, which could be deployed in the LSTF task. Wu et al. [8] successfully embed the series decomposition module into their model, making the model learn more predictable features. Inspired by them, we try to embed the series decomposition module into our Transformer-based model, which can break down the input sequence into a trend part and a seasonal part. The trend part represents the long-term trend of the sequence, while the seasonal part represents the periodic characteristics of the sequence. Specifically, we use the moving average algorithm to calculate the long-term trend of the input series, and then subtract the trend part from the original input to obtain the seasonal part of the sequence. The specific series decomposition process is as follows:

$$\begin{aligned} X_t &= \mathrm{AvePool}\left(\mathrm{Padding}\left(X_{\text{series}}\right)\right) \\ X_s &= X_{\text{series}} - X_t, \end{aligned} \tag{2}$$

where $X_{\text{series}} \in \mathbb{R}^{n \times d_{\text{model}}}$ represents the input of the series decomposition module. The purpose of $\mathrm{Padding}(\cdot)$ is to keep the length of the input series unchanged. $\mathrm{AvePool}(\cdot)$ is 1D average pooling whose pooling window size is expressed as Kernel size. X_t and X_s represent the trend part and seasonal part, respectively.

D. Sparse self-attention

The self-attention mechanism of the canonical Transformer is defined based on the inputs (query, key, and value), which conducts the scaled dot-product as shown in (3):

$$\mathcal{A} = \mathrm{Softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{model}}}}\right)V, \tag{3}$$

where the query $Q = \mathcal{X}W_Q \in \mathbb{R}^{n \times d_{\text{model}}}$, the key $K = \mathcal{X}W_K \in \mathbb{R}^{n \times d_{\text{model}}}$, and the value $V = \mathcal{X}W_V \in \mathbb{R}^{n \times d_{\text{model}}}$. $\mathcal{X}$ is the input of the encoder-decoder and n is the number of variables. To better illustrate the self-attention mechanism, let q_i represent the i_{th} row of Q. The corresponding attention output of the q_i can be defined as:

$$\mathcal{A}_{q_i} = \mathrm{Softmax}\left(\frac{q_i K^\top}{\sqrt{d_{\text{model}}}}\right)V. \tag{4}$$

As we can see from (4), the q_i can attend to all rows in K. However, not all of the rows can give the q_i useful information. For a specific variable, it is usually impacted by some of the other variables. Take the traffic flow prediction as an example: when there is a traffic accident at a certain intersection, the traffic flow at intersections that are nearby may increase; in contrast, the traffic flow at intersections that are far away will probably not be affected.

To eliminate the impact of the irrelevant elements of attention, we deploy sparse self-attention to replace the canonical attention mechanism and the specific process can be seen in Fig. 5. The corresponding sparse attention output of the q_i can be defined as:

$$\mathcal{A}_{q_i} = \mathrm{Softmax}\left(\mathrm{Mask}\left(\mathrm{Top}_k\left(\frac{q_i K^\top}{\sqrt{d_{\text{model}}}}\right)\right)\right)V, \tag{5}$$

where $\mathrm{Top}_k(\cdot)$ denotes selecting top k values, and $\mathrm{Mask}(\cdot)$ represents that the values unselected will be masked by $-\infty$, making the selected information get more attention. k represents the number of variables that are the most relevant to the target variable. Because the function of other variables is auxiliary to the prediction of the target variable, it is enough to take a few other variables into account. Therefore, for the dataset having numerous other variables, we just list a small range of k values. Taking the Electricity dataset as an example, whose number of other variables is 320, we just take the value of k from 1 to 10. However, for the dataset having a few other variables, we take the number of total variables as the maximum value of k. Taking the ETTh1 dataset as an example, whose number of variables is 7, we take the value of k from 1 to 7 and then choose the optimal k that makes the model's error the smallest.

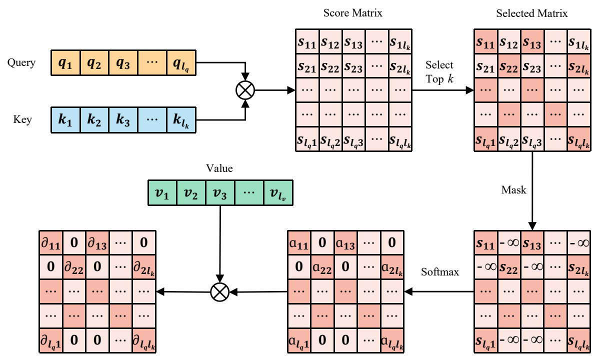


Fig. 5. The specific process of sparse self-attention. In each row of the Selected Matrix, the darker squares, whose total number is k, indicate that the values are retained while others will be masked.

E. Encoder-Decoder and trend module

As shown in Fig. 1, the encoder layer of PCDformer is primarily responsible for extracting the seasonal features of the input sequence and applying them to cross attention that can refine prediction results. Assuming there are N encoder layers, the input and output of the i^{th} encoder layer can be summarized as $X_{enc}^{i} = \mathrm{Encoder}(X_{enc}^{i-1})$. The specific calculation process is as follows:

$$\begin{aligned} S_{enc}^{i,1}, \_ &= \mathrm{SD}\left(\mathrm{SMHA}\left(X_{enc}^{i-1}\right) + X_{enc}^{i-1}\right) \\ S_{enc}^{i,2}, \_ &= \mathrm{SD}\left(\mathrm{FF}\left(\mathrm{Norm}\left(S_{enc}^{i,1}\right)\right) + \mathrm{Norm}\left(S_{enc}^{i,1}\right)\right), \end{aligned} \tag{6}$$

where _ represents the eliminated trend part, SD$(\cdot)$ denotes the series decomposition operation, SMHA$(\cdot)$ is the Sparse Multi-Head Attention layer and FF$(\cdot)$ expresses the Feed Forward layer. $X_{enc}^{i} = \mathrm{Norm}(S_{enc}^{i,2}), i \in \{1, \cdots, N\}$ represents the output of the i^{th} encoder layer. In particular, X_{enc}^{0} is the output of the Embedding layer. $S_{enc}^{i,j}, j \in \{1, 2\}$ denotes the seasonal part of the j^{th} series decomposition module in the i^{th} encoder layer.

Similarly, assuming that PCDformer has M decoder layers, the output and input of the p^{th} decoder layer can be summarized as $X_{dec}^{p} = \mathrm{Decoder}(X_{dec}^{p-1}, X_{enc}^{N})$, where X_{enc}^{N} represents the output of the N^{th} encoder layer. The specific calculation process is as follows:

$$\begin{aligned} S_{dec}^{p,1}, T_{dec}^{p,1} &= \mathrm{SD}\left(\mathrm{SMHA}\left(X_{dec}^{p-1}\right) + X_{dec}^{p-1}\right) \\ S_{dec}^{p,2}, T_{dec}^{p,2} &= \mathrm{SD}\left(\mathrm{MHA}\left(\mathrm{Norm}\left(S_{dec}^{p,1}\right), X_{enc}^{N}\right) \right. \\ &\quad \left. + \mathrm{Norm}\left(S_{dec}^{p,1}\right)\right) \\ S_{dec}^{p,3}, T_{dec}^{p,3} &= \mathrm{SD}\left(\mathrm{FF}\left(\mathrm{Norm}\left(S_{dec}^{p,2}\right)\right) \right. \\ &\quad \left. + \mathrm{Norm}\left(S_{dec}^{p,2}\right)\right), \end{aligned} \tag{7}$$

where MHA$(\cdot)$ is the Multi-Head Attention layer. $X_{dec}^{p} = \mathrm{Norm}(S_{dec}^{p,3}), p \in \{1, \cdots, M\}$ represents the output of the p^{th} decoder layer. In particular, X_{dec}^{0} is the output of the

Embedding layer. $\{S_{dec}^{p,q}, T_{dec}^{p,q}\}, q \in \{1,2,3\}$ denote seasonal part and trend part of the q^{th} series decomposition module in the p^{th} decoder layer, respectively.

Since the encoder layer discards the trend part, to complete the trend part of the sequence, we add the trend module in Fig. 1 to learn the trend characteristics of the input sequence. The specific calculation process is as follows:

$$\begin{aligned}\mathcal{T} &= \mathrm{FC}\left(\mathrm{SD}\left(X_{enc}\right) + \mathrm{Mean}\left(X_{enc}\right)\right) \\ &+ \sum_{p=1}^{M}\left(T_{dec}^{p,1} + T_{dec}^{p,2} + T_{dec}^{p,3}\right),\end{aligned} \tag{8}$$

where FC$(\cdot)$ is the linear layer.

V. Experiments

A. Datasets and baselines

Experiments are mainly conducted on **5** common time series datasets [8]: (1) ETT (ETTh1, ETTm1) dataset contains load and oil temperature collected from electricity transformers, which is recorded every 15 minutes between July 2016 and July 2018; (2) Electricity dataset contains the hourly electricity consumption of 321 customers from 2012 to 2014; (3) Traffic dataset contains data collected from the California Department of Transportation every hour, describing the road occupancy rates measured by different sensors on San Francisco Bay area freeways; (4) Weather dataset is collected every 10 minutes for the 2020 whole year, containing 21 meteorological indicators.

PCDformer is mainly compared with the **16** latest state-of-the-art methods from two categories: (1) representation learning techniques, including SimTS, LaST, ACST, CoST, TS2Vec, and TNC; (2) end-to-end forecasting models: (i) Transformer-based models, including CLformer, FEDformer, Autoformer, Pyraformer, Informer, Transformer, LogTrans and Reformer; (ii) others: VAE-GRU and TCN.

B. Experimental details

Following the previous work, we set the LSTF into two categories, univariate and multivariate forecasting. In univariate forecasting, the input and output of the model only consider a specific variate. In multivariate forecasting, the model receives and predicts all variables. For the split of the datasets, we follow the standard method, dividing all datasets into training, validation, and testing set by the ratio of 6:2:2. Our model is trained by L2 loss and uses the ADAM optimizer to calculate and update the parameters of the network. The total number of the training epochs is 10 with suitable early stopping. PCDformer consists of 2 Encoder layers and 1 Decoder layer. All experiments are implemented in PyTorch and conducted on a single NVIDIA GeForce RTX 2080ti GPU. Algorithms 1 and 2 illustrate the learning algorithm of PCDformer and the steps of forecasting calculation, respectively. Mean Absolute Errors (MAE) and Mean Squared Errors (MSE) are used to evaluate the performance of all models, shown as follows:

$$\begin{aligned}MAE &= \frac{1}{\tau}\sum_{i=t_0}^{t_0+\tau-1}\left|\hat{x}_i - x_i\right| \\ MSE &= \frac{1}{\tau}\sum_{i=t_0}^{t_0+\tau-1}\left(\hat{x}_i - x_i\right)^2,\end{aligned} \tag{9}$$

where $\hat{x}_i$ denotes the prediction value of the model, x_i represents the ground-truth, and τ is the length of the prediction steps.

TABLE I
The overall information of the five datasets.

Datasets	ETTh1	ETTm1	Traffic	Electricity	Weather
Variants	7	7	862	321	21
Timesteps	17,420	69,680	17,544	26,304	52,696
Granularity	1hour	15min	1hour	1hour	10min
Task type	Multi-step	Multi-step	Multi-step	Multi-step	Multi-step
Data partition	Training/Validation/Testing: 6/2/2				

C. Results and analyses

The results of the long-term time series forecasting of all methods on five datasets are shown in Table II, Table IV, and Table VI, which indicate the performance of univariate forecasting and multivariate forecasting for various time steps of future. Table III, Table V, and Table VII represent the corresponding hype-parameters.

Algorithm 1 The learning algorithm of PCDformer for LSTF.

```
Input:
1: $data$: the dataset;
2: $L$: the input length of PCDformer;
3: $n$: the number of total variables;
4: $bs$: the batch size;
5: $lr$: the learning rate;
6: $loss(\cdot)$: the L2 loss function;
7: $f(\cdot)$: the initialized PCDformer with $\theta$;
8: $epochs$: the number of training epochs;
9: $g$: the decay rate of the learning rate;
Output: the optimal model parameters $\theta$;
1: for $i$ in $epochs$ do:
2:     for each batch ($\mathrm{X_{enc}} \in \mathbb{R}^{bs\times n\times L}, \mathrm{X_{dec}} \in \mathbb{R}^{bs\times n\times(L+\tau)}; Y \in \mathbb{R}^{bs\times\tau\times n}$) from $data$ do:
3:         compute $\hat{Y} = f(\mathrm{X_{enc}} \in \mathbb{R}^{bs\times n\times L}, \mathrm{X_{dec}} \in \mathbb{R}^{bs\times n\times(L+\tau)}; \theta)$
4:         compute loss value $Loss = loss(\hat{Y}, Y)$
5:         update the parameters of PCDformer according to gradients and $lr$
6:     end for
7:     adjust learning rate: $lr = lr \times g$
8: end for
```

Univariate Time-series Forecasting: From Table II, we can observe that: (1) The proposed model PCDformer achieves better performance overall compared with the most advanced representation learning baseline LaST on

Algorithm 2 The steps of forecasting calculation of PCDformer for LSTF.

```
Input:
1: $data$: the dataset;
2: $\mathrm{X}_{\mathrm{enc}} \in \mathbb{R}^{n \times L}$: the input for parallel processing layer and trend module;
3: $\mathrm{X}_{\mathrm{dec}} \in \mathbb{R}^{n \times (L+\tau)}$: the input for the embedding layer of Decoder;
4: $ppl(\cdot)$: the parallel processing layer with parameters $\theta_{ppl}$;
5: $Enc(\cdot)$: the Encoder with parameters $\theta_{enc}$;
6: $Dec(\cdot)$: the Decoder with parameters $\theta_{dec}$;
7: $Tm(\cdot)$: the trend module with parameters $\theta_{tm}$;
8: $fcl(\cdot)$: the Fully Connected Layer with parameters $\theta_{fcl}$;
9: $T(\cdot)$: the Transpose operation;
Output: the forecasting values $\hat{Y}$;
1: for each batch $(\mathrm{X}_{\mathrm{enc}}, \mathrm{X}_{\mathrm{dec}})$ from $data$ do:
2:     feed $\mathrm{X}_{\mathrm{enc}}$ into parallel processing layer: $out_{ppl} = ppl(\mathrm{X}_{\mathrm{enc}}; \theta_{ppl})$
3:     compute Encoder output: $out_{enc} = Enc(out_{ppl}^{embed}; \theta_{enc})$
4:     feed $\mathrm{X}_{\mathrm{dec}}$ into Embedding layer: $out_{embed} = \mathrm{Embedding}(\mathrm{X}_{\mathrm{dec}})$
5:     compute Decoder output: $out_{dec} = Dec(out_{embed}, out_{enc}; \theta_{dec})$
6:     feed $\mathrm{X}_{\mathrm{enc}}$ into trend module layer: $out_{tm} = Tm(\mathrm{X}_{\mathrm{enc}}; \theta_{tm})$
7:     sum the value of extracted feature: $out_{sum} = out_{dec} + out_{tm}$
8:     compute the final forecasting results: $\hat{Y} = T(fcl(out_{sum}; \theta_{fcl}))$
9: end for
```

three real-world datasets (average MSE: 0.121→0.117, average MAE: 0.236→0.235). Especially, under the input-201-predict-720 and input-201-predict-672 settings for ETTh1 and ETTm1, PCDformer achieves **23.9%** (0.138→0.105) and **18.0%** (0.100→0.082) MSE reduction respectively. (2) When compared to models of the same category, end-to-end learning, PCDformer achieves state-of-the-art performance in all prediction horizon settings except for the Horizon=24 of VAE-GRU in ETTm1. Overall, PCDformer yields a **41.8%** (0.201→0.117) averaged MSE reduction and a **23.2%** (0.306→0.235) averaged MAE reduction.

Multivariate Time-series Forecasting: Under the multivariate settings, PCDformer achieves the following performance improvement: (1) Compared with the most advanced representation baseline LaST, our model PCDformer achieves better performance overall on three real-world datasets (average MSE: 0.334→0.313, average MAE: 0.369→0.364). In particular, for the input-201-predict-336 of ETTh1 and the input-201-predict-672 of ETTm1, PCDformer achieves **14.3%** (0.566→0.485) and **8.6%** (0.491→0.449) MSE reduction respectively. (2) When compared to the most advanced end-to-end learning model, PCDformer achieves the best performance in all prediction horizon settings except for the Horizon=720 of Autoformer in ETTh1. Overall, PCDformer yields a **19.7%** (0.390→0.313) averaged MSE reduction and a **13.3%** (0.420→0.364) averaged MAE reduction. Especially, under the input-201-predict-288 setting for ETTm1, PCDformer achieves **41.0%** (0.634→0.374) MSE reduction and **22.5%** (0.528→0.409) MAE reduction.

Since Transformer-based prediction models are more popular and the model PCDformer is also based on Transformer, this paper selects more Transformer-based methods in recent years for further performance comparison with this solution. As we can see from Table VI, PCDformer achieves state-of-the-art performance in most horizon settings of all baselines. Specifically, PCDformer achieves better performance overall compared with CLformer on three real-world datasets (average MSE: 0.371→0.356, average MAE: 0.350→0.330). Especially, under the input-96-predict-96 and input-96-predict-192 settings for Weather, PCDformer achieves **20.3%** (0.217→0.173) and **12.0%** (0.276→0.243) MSE reduction respectively compared with FEDformer. We attribute it to that: (1) The proposed parallel processing layer can extract the temporal dependence of each variable, including local features and global features. (2) The series decomposition module can disentangle complex temporal patterns, allowing the model to learn more predictable parts. Besides, the sparse self-attention mechanism can capture the correlation between variables and mitigate the impact of irrelevant information, enhancing the prediction performance of PCDformer. Fig. 6 presents a comparison of randomly selected sequences and variates from the Weather dataset. It vividly demonstrates the exceptional performance of PCDformer in addressing long sequence prediction problems, surpassing other Transformer-based models.

D. Parameter sensitivity

To analyze the parameter sensitivity of the model, we perform experiments on ETTh1 under multivariate settings. **Input length**: When predicting a specific horizon, keep hyper-parameters of the other input lengths consistent with that of input length 201. In Fig. 7 (a), we can see that when predicting not too long a sequence (like 24, 48, or 168), the MSE slowly decreases and then slowly rises as the input length increases. However, the MSE drops rapidly with longer inputs in predicting longer sequences (like 336 or 720). We can conclude that the length of the input sequence is proportional to the prediction horizon to some extent. That is, to predict longer future information, we can provide more historical information to obtain the best results, which is also in line with our conventional wisdom. **Top *k***: To analyze the sensitivity of k to the model, we keep the horizon (168) and other parameters unchanged. In Fig. 7 (b), the general performance drops a little with the k starting from 1 to 3 and keeps relatively stable at last with the k starting from 5 to 7. Therefore, we can assume that only some variables have useful relevant information between them, and the rest of the redundant information will interfere with the model prediction and weaken the model performance. $\boldsymbol{d}_{\mathbf{model}}$: In this experiment, the horizon is fixed length (168). To show the influence of different d_{model} on a specific input, we only

TABLE II
UNIVARIATE TIME-SERIES FORECASTING RESULTS ON ETTH1, ETTM1 AND ELECTRICITY.

Methods		Ours		Representation Learning										Ours		End-to-end Forecasting							
		PCDformer		SimTS [17]		LaST [14]		CoST [13]		TS2Vec [15]		TNC [16]		**PCDformer**		VAE-GRU [11]		Autoformer [8]		Informer [9]		TCN [10]	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	0.038	0.150	0.036	0.143	**0.030**	**0.131**	0.040	0.152	0.039	0.151	0.057	0.184	**0.038**	**0.150**	0.042	0.155	0.057	0.189	0.098	0.247	0.104	0.254
	48	**0.050**	0.171	0.054	0.176	0.051	**0.169**	0.060	0.186	0.062	0.189	0.094	0.239	**0.050**	**0.171**	0.077	0.218	0.070	0.207	0.158	0.319	0.206	0.366
	168	**0.075**	**0.208**	0.084	0.216	0.078	0.211	0.097	0.236	0.142	0.291	0.171	0.329	**0.075**	**0.208**	0.172	0.344	0.108	0.260	0.183	0.346	0.462	0.586
	336	**0.084**	**0.222**	0.100	0.239	0.100	0.246	0.112	0.258	0.160	0.316	0.179	0.345	**0.084**	**0.222**	0.140	0.301	0.119	0.281	0.222	0.387	0.422	0.564
	720	**0.105**	**0.254**	0.126	0.277	0.138	0.298	0.148	0.306	0.179	0.345	0.235	0.408	**0.105**	**0.254**	0.204	0.381	0.109	0.264	0.269	0.435	0.438	0.578
ETTm1	24	0.013	0.086	0.013	0.084	**0.011**	**0.077**	0.015	0.088	0.016	0.093	0.019	0.103	**0.013**	0.086	0.013	**0.082**	0.022	0.115	0.030	0.137	0.027	0.127
	48	**0.021**	0.111	0.024	0.112	**0.021**	**0.108**	0.025	0.117	0.028	0.126	0.045	0.162	**0.021**	**0.111**	0.026	0.120	0.032	0.138	0.069	0.203	0.040	0.154
	96	**0.032**	0.138	0.041	0.143	0.033	**0.134**	0.038	0.147	0.045	0.162	0.054	0.178	**0.032**	**0.138**	0.046	0.164	0.045	0.168	0.194	0.372	0.097	0.246
	288	**0.065**	**0.193**	0.098	0.207	0.069	0.197	0.077	0.209	0.095	0.235	0.142	0.290	**0.065**	**0.193**	0.127	0.294	0.071	0.207	0.401	0.554	0.305	0.455
	672	**0.082**	**0.216**	0.117	0.242	0.100	0.239	0.113	0.257	0.142	0.290	0.136	0.290	**0.082**	**0.216**	0.217	0.399	0.102	0.254	0.512	0.644	0.445	0.576
Electricity	24	**0.149**	0.285	–	–	0.151	**0.277**	0.243	0.264	0.260	0.288	0.252	0.278	**0.149**	**0.285**	0.330	0.406	0.290	0.411	0.251	0.275	0.243	0.367
	48	**0.184**	0.314	–	–	0.186	**0.307**	0.292	0.300	0.313	0.321	0.300	0.308	**0.184**	**0.314**	0.437	0.481	0.310	0.408	0.346	0.339	0.283	0.397
	168	0.250	0.358	–	–	0.243	**0.346**	0.405	0.375	0.429	0.392	0.412	0.384	**0.250**	**0.358**	0.433	0.476	0.435	0.490	0.544	0.424	0.357	0.449
	336	**0.286**	0.389	–	–	**0.286**	**0.379**	0.560	0.473	0.565	0.478	0.548	0.466	**0.286**	**0.389**	0.472	0.504	0.646	0.606	0.713	0.512	0.355	0.446
	720	**0.320**	0.427	–	–	0.322	**0.422**	0.889	0.645	0.863	0.651	0.859	0.651	**0.320**	**0.427**	0.543	0.563	0.609	0.587	1.182	0.806	0.387	0.477
Avg.		**0.117**	**0.235**	N/A	N/A	0.121	0.236	0.208	0.268	0.222	0.289	0.234	0.328	**0.117**	**0.235**	0.219	0.326	0.201	0.306	0.345	0.400	0.278	0.403

[1] The best results are represented in bold font, and the second-best results are represented with an underline. The other models' results can refer to [14], [17].
[2] - denotes that the methods are not implemented on this dataset. N/A indicates that the average is not available because of the existence of missing values.

TABLE III
THE HYPER-PARAMETERS OF PCDFORMER ON ETTH1, ETTM1 AND ELECTRICITY DATASETS FOR UNIVARIATE TIME-SERIES FORECASTING.

Model configurations		ETTh1					ETTm1					Electricity				
Hyper-parameter	Horizon	24	48	168	336	720	24	48	96	288	672	24	48	168	336	720
	Look-back window	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201
	Batch size	32	32	32	32	32	32	32	32	32	256	64	128	128	256	256
	Learning rate	8e-5	8e-5	2.5e-5	2.5e-5	2.5e-5	5e-5	5e-5	8e-5	8e-5	5e-5	1.2e-4	1.5e-4	8e-5	5.5e-5	8e-5
	d_{model}	256	256	512	512	1024	256	256	256	256	512	256	256	512	1024	1024
	Dropout	0.4	0.4	0.5	0.5	0.5	0.2	0.45	0.55	0.55	0.6	0.12	0.15	0.15	0.005	0.01
	Kernel size	65	65	45	45	45	65	45	45	45	45	45	45	45	45	45

TABLE IV
MULTIVARIATE TIME-SERIES FORECASTING RESULTS ON ETTH1, ETTM1 AND ELECTRICITY.

Methods		Ours		Representation Learning												Ours		End-to-end Forecasting							
		PCDformer		SimTS		LaST		ACST [37]		CoST		TS2Vec		TNC		**PCDformer**		VAE-GRU		Autoformer		Informer		TCN	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	**0.322**	0.376	0.377	0.422	0.324	**0.368**	0.440	0.454	0.386	0.429	0.590	0.531	0.708	0.592	**0.322**	**0.376**	0.529	0.534	0.384	0.428	0.577	0.549	0.583	0.547
	48	0.357	0.399	0.427	0.454	**0.351**	**0.380**	0.468	0.470	0.437	0.464	0.624	0.555	0.749	0.619	**0.357**	**0.399**	0.612	0.593	0.392	0.419	0.685	0.625	0.670	0.606
	168	**0.444**	**0.450**	0.638	0.577	0.468	0.453	0.535	0.511	0.643	0.582	0.762	0.639	0.884	0.699	**0.444**	**0.450**	0.758	0.647	0.490	0.481	0.931	0.752	0.811	0.680
	336	**0.485**	**0.475**	0.815	0.685	0.566	0.512	0.594	0.552	0.812	0.679	0.931	0.728	1.020	0.768	**0.485**	**0.475**	0.844	0.692	0.505	0.484	1.128	0.873	1.132	0.815
	720	**0.680**	**0.610**	0.956	0.771	0.740	0.650	0.682	0.622	0.970	0.771	1.063	0.799	1.157	0.830	0.680	0.610	1.045	0.816	**0.498**	**0.500**	1.215	0.896	1.165	0.813
ETTm1	24	**0.200**	**0.286**	0.232	0.314	0.218	0.289	0.381	0.406	0.246	0.329	0.453	0.444	0.522	0.472	**0.200**	**0.286**	0.509	0.452	0.383	0.403	0.453	0.444	0.522	0.472
	48	**0.264**	0.331	0.311	0.368	0.280	**0.329**	0.476	0.470	0.331	0.386	0.592	0.521	0.695	0.567	**0.264**	**0.331**	0.642	0.543	0.454	0.453	0.494	0.503	0.542	0.508
	96	**0.296**	**0.357**	0.360	0.402	0.323	0.360	0.511	0.497	0.378	0.419	0.635	0.554	0.731	0.595	**0.296**	**0.357**	0.600	0.540	0.481	0.463	0.678	0.614	0.666	0.578
	288	**0.374**	0.409	0.450	0.467	0.392	**0.403**	0.528	0.506	0.472	0.486	0.693	0.597	0.818	0.649	**0.374**	**0.409**	0.769	0.678	0.634	0.528	1.056	0.786	0.991	0.735
	672	**0.449**	**0.457**	0.612	0.563	0.491	0.466	0.565	0.529	0.620	0.574	0.782	0.653	0.932	0.712	**0.449**	**0.457**	0.799	0.673	0.606	0.542	1.192	0.926	1.032	0.756
Electricity	24	**0.107**	**0.207**	–	–	0.125	0.222	–	–	0.136	0.242	0.287	0.375	0.354	0.423	**0.107**	**0.207**	0.190	0.250	0.165	0.286	0.312	0.387	0.235	0.346
	48	**0.124**	**0.225**	–	–	0.146	0.245	–	–	0.153	0.258	0.309	0.391	0.376	0.438	**0.124**	**0.225**	0.228	0.280	0.178	0.295	0.392	0.431	0.253	0.359
	168	**0.165**	**0.265**	–	–	0.170	**0.265**	–	–	0.175	0.275	0.335	0.410	0.402	0.456	**0.165**	**0.265**	0.240	0.297	0.215	0.327	0.515	0.509	0.278	0.372
	336	0.191	0.287	–	–	**0.188**	**0.280**	–	–	0.196	0.296	0.351	0.422	0.417	0.466	**0.191**	**0.287**	0.262	0.318	0.218	0.329	0.759	0.625	0.287	0.382
	720	0.230	0.322	–	–	**0.223**	**0.309**	–	–	0.232	0.327	0.378	0.440	0.442	0.483	**0.230**	**0.322**	0.296	0.347	0.252	0.356	0.969	0.788	0.287	0.381
Avg.		**0.313**	**0.364**	N/A	N/A	0.334	0.369	N/A	N/A	0.412	0.434	0.586	0.537	0.680	0.585	**0.313**	**0.364**	0.555	0.511	0.390	0.420	0.757	0.647	0.630	0.557

change the value of d_{model}, keeping other hyper-parameters the same as Table V. In Fig. 7 (c), when the input length keeps unchanged and the value of the d_{model} increases to a certain extent, PCDformer can obtain the optimal performance. However, too large a d_{model} will also make the performance worse, which may be caused by overfitting. In practice, we will adaptively adjust the value of d_{model} according to the length of the input sequence, shown in Table III & V & VII, so that the performance of the model can be optimal.

E. Ablation study

The performance of parallel processing layer: In this experiment, we testify to the performance of the parallel processing layer for extracting the temporal dependencies. To better illustrate the influence of the parallel processing layer on the model, we remain other settings unchanged and just remove the parallel processing layer from PCDformer. As we can see from Table VIII, the performance of PCDformer† is inferior to PCDformer. It proves the parallel processing layer's ability to capture the temporal dependencies of time steps, which enhances the model's performance of prediction.

The performance of the sparse self-attention: In this study, we substitute the canonical self-attention for the sparse self-attention, while the other experimental settings are aligned with that of multivariate time series forecasting. From Table VIII, we can learn that except for Horizon=720 on ETTh1, PCDformer achieves better performance compared with PCDformer‡ when forecasting various time steps. The reason we think is that although there may be correlations between the input sequences of different variables, there is also a large amount of useless information, which will interfere with the training and inference of the model, and eventually reduce the accuracy of the model's prediction. However, unlike self-attention, which considers the correlation between all

TABLE V
THE HYPER-PARAMETERS OF PCDFORMER ON ETTH1, ETTM1 AND ELECTRICITY DATASETS FOR MULTIVARIATE TIME-SERIES FORECASTING.

Model configurations		ETTh1					ETTm1					Electricity				
Hyper-parameter	Horizon	24	48	168	336	720	24	48	96	288	672	24	48	168	336	720
	Look-back window	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201
	Batch size	16	32	32	32	64	16	32	32	64	64	16	16	32	32	32
	Learning rate	8e-5	1.5e-4	1.5e-4	1.5e-4	1.5e-5	1e-4	5e-5	5e-5	5e-5	1e-4	1.5e-4	1.5e-4	1.5e-4	2.5e-4	5e-4
	d_{model}	256	256	512	512	1024	256	256	256	512	1024	256	256	512	512	512
	Dropout	0.05	0.45	0.5	0.5	0.5	0.05	0.2	0.25	0.25	0.25	0.05	0.05	0.05	0.02	0.05
	Kernel size	25	45	45	45	45	45	45	45	45	45	45	45	45	45	45
	Top k	1	2	2	2	2	1	1	1	1	1	1	1	1	1	1

TABLE VI
LONG-TERM FORECASTING PERFORMANCE COMPARISON WITH TRANSFORMER-BASED MODELS ON ELECTRICITY, TRAFFIC AND WEATHER.

Model		**PCDformer**		CLformer [12]		FEDformer [21]		Autoformer [8]		Pyraformer [20]		Informer [9]		Transformer [36]		LogTrans [19]		Reformer [18]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	**0.160**	**0.259**	0.191	0.308	0.193	0.308	0.201	0.317	0.386	0.449	0.274	0.368	0.263	0.359	0.258	0.357	0.312	0.402
	192	**0.178**	**0.276**	0.200	0.314	0.201	0.315	0.222	0.334	0.378	0.443	0.296	0.296	0.273	0.374	0.266	0.368	0.348	0.433
	336	**0.204**	**0.298**	**0.204**	0.319	0.214	0.329	0.231	0.338	0.376	0.443	0.300	0.394	0.277	0.373	0.280	0.380	0.350	0.433
	720	0.246	0.338	**0.224**	**0.335**	0.246	0.355	0.254	0.361	0.376	0.445	0.373	0.439	0.290	0.378	0.283	0.376	0.340	0.420
Traffic	96	**0.577**	**0.350**	0.581	0.363	0.587	0.366	0.613	0.388	0.867	0.468	0.719	0.391	0.638	0.354	0.684	0.384	0.732	0.423
	192	**0.563**	**0.335**	0.589	0.361	0.604	0.373	0.616	0.382	0.869	0.467	0.696	0.379	0.647	0.354	0.685	0.390	0.733	0.420
	336	**0.575**	0.345	0.603	0.368	0.621	0.383	0.622	**0.337**	0.881	0.469	0.777	0.420	0.669	0.364	0.733	0.408	0.742	0.420
	720	**0.606**	**0.364**	0.613	0.367	0.626	0.382	0.660	0.408	0.896	0.473	0.864	0.472	0.707	0.386	0.717	0.396	0.755	0.423
Weather	96	**0.173**	**0.243**	0.218	0.298	0.217	0.296	0.266	0.336	–	–	0.300	0.384	0.656*	0.576*	0.458	0.490	0.689	0.596
	192	**0.243**	**0.308**	0.288	0.356	0.276	0.336	0.307	0.367	–	–	0.598	0.544	1.139*	0.765*	0.658	0.589	0.752	0.638
	336	**0.328**	**0.373**	0.339	0.391	0.339	0.380	0.359	0.395	–	–	0.578	0.523	1.411*	0.858*	0.797	0.652	0.639	0.596
	720	0.440	0.476	0.406	**0.424**	**0.403**	0.428	0.419	0.428	–	–	1.059	0.741	1.931*	1.035*	0.869	0.675	1.130	0.792

[1] $*$ denotes re-implementation. — denotes that the methods are not implemented on this dataset. The other models' results can refer to [8], [12], [21], [40].
[2] The experimental setting is the same as the Autoformer, using the input length of 96 to predict various future horizons$\{96, 192, 336, 720\}$.

TABLE VII
THE HYPER-PARAMETERS OF PCDFORMER ON ELECTRICITY, TRAFFIC AND WEATHER DATASETS FOR MULTIVARIATE TIME-SERIES FORECASTING.

Model configurations		Electricity				Traffic				Weather			
Hyperparameter	Horizon	96	192	336	720	96	192	336	720	96	192	336	720
	Look-back window	96											
	Batch size	32	32	16	32	16	16	16	16	32	32	64	256
	Learning rate	1.5e-4	2.5e-4	1e-4	1e-4	2e-4	2e-4	2.5e-4	3e-4	1e-4	5e-5	5e-5	9e-6
	d_{model}	512	512	1024	1024	256	512	512	512	256	256	512	512
	Dropout	0.05	0.04	0.02	0.02	0.2	0.2	0.25	0.3	0.05	0.05	0.025	0.01
	Kernel size	45	45	45	45	25	25	25	25	25	25	25	25
	Top k	1	1	1	1	1	1	1	1	1	1	1	1

variables, sparse self-attention only considers the parts of the variables that are strongly correlated, eliminating the negative impact of redundant information.

The performance of the decomposition module: In this test, we use PCDformer§ to demonstrate the effectiveness of the strategy of series decomposition. In the overall results of Table VIII, we can see that the general prediction performance of PCDformer§ is worse than PCDformer except for Horizon=720 on ETTh1. We can conclude that the series decomposition beneficial to the enhancement of the model's prediction accuracy is worth adopting.

TABLE VIII
ABLATION STUDY ON ETTH1 AND ETTM1 DATASETS UNDER MULTIVARIATE SETTINGS. KEEP ALL HYPER-PARAMETERS THE SAME AS TABLE V.

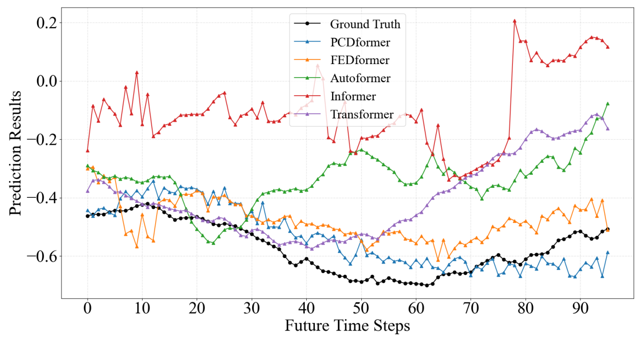
Dataset		ETTh1					ETTm1				
Look-back window		201					201				
Horizon		24	48	168	336	720	24	48	96	288	672
PCDformer	MSE	**0.322**	**0.357**	**0.444**	**0.485**	0.680	**0.200**	**0.264**	**0.296**	**0.374**	**0.449**
	MAE	**0.376**	**0.399**	**0.450**	**0.475**	0.610	**0.286**	**0.331**	**0.357**	**0.409**	**0.457**
PCDformer†	MSE	0.349	0.386	0.493	0.510	0.693	0.205	0.297	0.319	0.405	0.490
	MAE	0.403	0.425	0.484	0.496	0.613	0.291	0.357	0.374	0.434	0.482
PCDformer‡	MSE	0.330	0.363	0.448	0.493	0.671	0.216	0.270	0.306	0.385	0.523
	MAE	0.382	0.405	0.455	0.481	0.605	0.294	0.339	0.365	0.417	0.512
PCDformer§	MSE	0.334	0.414	0.473	0.541	**0.641**	0.218	0.278	0.315	0.398	0.480
	MAE	0.386	0.457	0.470	0.509	**0.592**	0.295	0.347	0.373	0.425	0.477

[1] PCDformer† removes the parallel processing layer from PCDformer.
[2] PCDformer‡ applies the canonical self-attention mechanism.
[3] PCDformer§ removes the series decomposition and trend module from PCDformer.

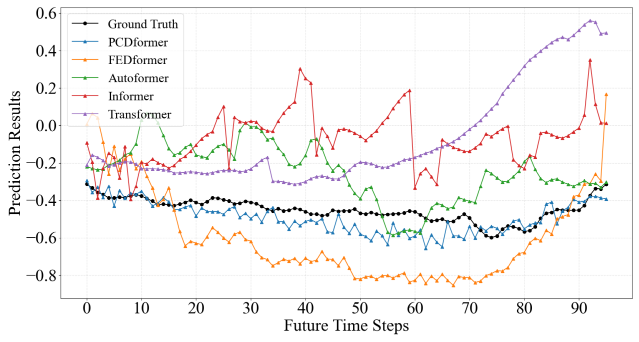
F. Complexity analysis and direction of improvement

Based on the analysis conducted in Section IV-A, we can know that the complexity of PCDformer is $\mathcal{O}(n^2 * d)$, where n represents the total number of input variables, and d represents the d_{model}. Unlike other models that have complexity dependent on the length of the input sequence [8], [9], [18], the complexity of our model primarily depends on the total number of input variables. Therefore, our model can demonstrate a significant improvement in complexity when dealing with a smaller number of input variables, especially for long series. However, when the total number of input variables is substantial, our model becomes relatively more complex to handle. The main reason is that the sparse self-attention mechanism considers attention magnitudes between each variable and all others when computing Q and K. While this effectively captures attention relationships, it also increases computational workload. Hence, exploring feasible algorithms to enhance sparse self-attention is a potential improvement direction.

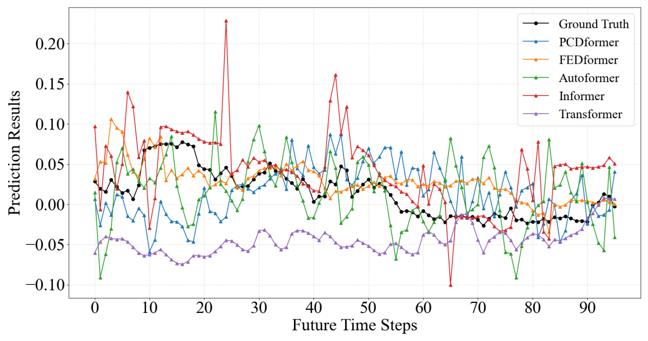
Moreover, from Fig. 6, we can find that the curve is not highly smooth although PCDformer can learn the overall



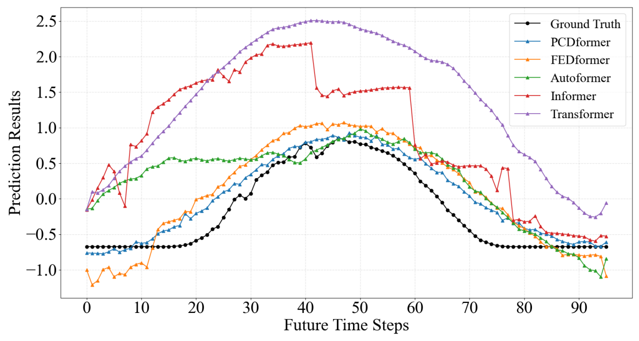
(a) Sequence 501, Variate 1



(b) Sequence 1001, Variate 6

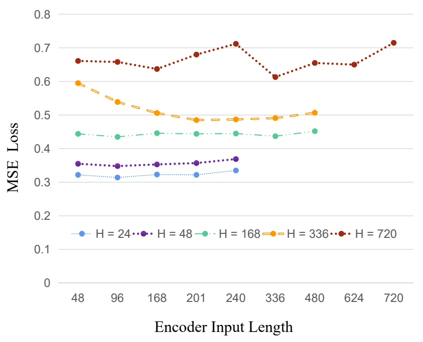


(c) Sequence 2001, Variate 12

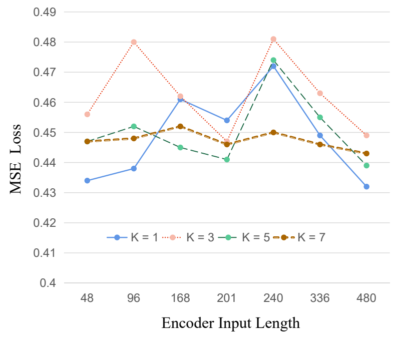


(d) Sequence 2501, Variate 18

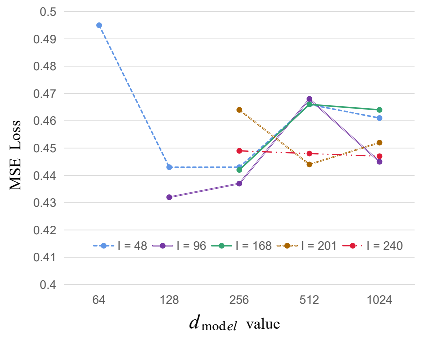
Fig. 6. The prediction results (Horizon = 96) of PCDformer, FEDformer, Autoformer, Informer, and Transformer on randomly selected sequences and variates from the Weather dataset.



(a) Input length. The legend denotes the length of prediction.



(b) Top k. The legend denotes the value of k.



(c) Representation dimension. The legend denotes the length of input.

Fig. 7. The results of the parameter sensitivity.

trend well. We think the reason is the existence of noise in the original series. Therefore, another improvement direction based on this paper is to eliminate unpredictable noise signals in the original sequence. A feasible solution to this problem is to convert the seasonal terms in the time domain into signals in the frequency domain through the Fourier transform so that the noise signal can be filtered out through the gate function.

Finally, we can consider replacing M-inception with multi-scale dilated convolutions [41] to reduce the introduction of additional parameters.

VI. Conclusion

To address challenges related to information coupling, the entanglement of temporal patterns, and the impact of irrelevant information between variables in LSTF tasks, we propose PCDformer, which incorporates a parallel processing layer, sparse self-attention mechanism, and series decomposition. By parallelizing and decomposing the input sequence, PCD-former effectively uncouples variables and disentangles different temporal patterns. A sparse self-attention mechanism is deployed to eliminate the irrelevant information between variables and make the model pay more attention to the useful information. Extensive experiments on diverse real-world datasets are conducted to show the superior performance of PCDformer compared to state-of-the-art methods in solving LSTF problems.

REFERENCES

[1] F. Liu, Q. Tao, D. Yang, and D. Sidorov, "Bidirectional Gated Recurrent Unit-Based Lower Upper Bound Estimation Method for Wind Power Interval Prediction," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 3, pp. 461–469, 2022.

[2] G. Li, A. Zhang, Q. Zhang, D. Wu, and C. Zhan, "Pearson Correlation Coefficient-Based Performance Enhancement of Broad Learning System for Stock Price Prediction," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp. 2413–2417, 2022.

[3] P. S. Mung and S. Phyu, "Time Series Weather Data Forecasting Using Deep Learning," in *ICCA*, 2023, pp. 254–259.

[4] H. Jiang, S. Zhang, W. Yang, X. Peng, and W. Zhong, "Integration of Encoding and Temporal Forecasting: Toward End-to-End NO $_x$ Prediction for Industrial Chemical Process," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2023.

[5] W. Liao, Z. Yang, X. Chen, and Y. Li, "WindGMMN: Scenario Forecasting for Wind Power Using Generative Moment Matching Networks," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 5, pp. 843–850, 2022.

[6] D. Liu, Y. L. Wu, X. Li, and L. Qi, "Medi-Care AI: Predicting medications from billing codes via robust recurrent neural networks," *Neural Networks*, vol. 124, pp. 109–116, 2020.

[7] S. Ma, T. Zhang, Y.-B. Zhao, Y. Kang, and P. Bai, "TCLN: A Transformer-based Conv-LSTM network for multivariate time series forecasting," *Applied Intelligence*, pp. 1–17, 2023.

[8] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting," in *NeurIPS*, 2021, pp. 22 419–22 430.

[9] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," in *AAAI*, 2021, pp. 11 106–11 115.

[10] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[11] O. Fabius, J. R. van Amersfoort, and D. P. Kingma, "Variational Recurrent Auto-Encoders," in *ICLR*, 2015.

[12] X. Wang, H. Liu, J. Du, Z. Yang, and X. Dong, "CLformer: Locally grouped auto-correlation and convolutional transformer for long-term multivariate time series forecasting," *Engineering Applications of Artificial Intelligence*, vol. 121, p. 106042, 2023.

[13] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. C. H. Hoi, "CoST: Contrastive Learning of Disentangled Seasonal-Trend Representations for Time Series Forecasting," in *ICLR*, 2022.

[14] Z. Wang, X. Xu, W. Zhang, G. Trajcevski, T. Zhong, and F. Zhou, "Learning Latent Seasonal-Trend Representations for Time Series Forecasting," in *NeurIPS*, 2022.

[15] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "TS2Vec: Towards Universal Representation of Time Series," in *AAAI*, 2022, pp. 8980–8987.

[16] S. Tonekaboni, D. Eytan, and A. Goldenberg, "Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding," in *ICLR*, 2021.

[17] X. Zheng, X. Chen, M. Schürch, A. Mollaysa, A. Allam, and M. Krauthammer, "SimTS: Rethinking Contrastive Representation Learning for Time Series Forecasting," *arXiv preprint arXiv:2303.18205*, 2023.

[18] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The Efficient Transformer," in *ICLR*, 2020.

[19] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting," in *NeurIPS*, vol. 32, 2019.

[20] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting," in *ICLR*, 2022.

[21] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting," in *ICML*, 2022, pp. 27 268–27 286.

[22] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A Seasonal-Trend Decomposition Procedure Based on Loess," *J. Off. Stat*, vol. 6, no. 1, pp. 3–73, 1990.

[23] X. Wang, H. Zhang, Y. Zhang, M. Wang, J. Song, T. Lai, and M. Khushi, "Learning Nonstationary Time-Series With Dynamic Pattern Extractions," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 5, pp. 778–787, 2022.

[24] S. Tonekaboni, C. Li, S. Ö. Arik, A. Goldenberg, and T. Pfister, "Decoupling Local and Global Representations of Time Series," in *AISTATS*, vol. 151, 2022, pp. 8700–8714.

[25] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *WAIM*, 2014, pp. 298–310.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[27] Y. Yang and J. Lu, "Foreformer: an enhanced transformer-based framework for multivariate time series forecasting," *Applied Intelligence*, vol. 53, no. 10, pp. 12 521–12 540, 2023.

[28] F. Li, Z. Wan, T. Koch, G. Zan, M. Li, Z. Zheng, and B. Liang, "Improving the accuracy of multi-step prediction of building energy consumption based on EEMD-PSO-Informer and long-time series," *Computers and Electrical Engineering*, vol. 110, p. 108845, 2023.

[29] H. Gao, H. Su, Y. Cai, R. Wu, Z. Hao, Y. Xu, W. Wu, J. Wang, Z. Li, and Z. Kan, "Trajectory prediction of cyclist based on dynamic Bayesian network and long short-term memory model at unsignalized intersections," *Science China Information Sciences*, vol. 64, no. 7, p. 172207, 2021.

[30] X. Zou, S. Zhang, C. Zhang, J. J. Q. Yu, and E. Chung, "Long-Term Origin-Destination Demand Prediction With Graph Deep Learning," *IEEE Transactions on Big Data*, vol. 8, no. 6, pp. 1481–1495, 2022.

[31] L. Laurenti, E. Tinti, F. Galasso, L. Franco, and C. Marone, "Deep learning for laboratory earthquake prediction and autoregressive forecasting of fault zone stress," *Earth and Planetary Science Letters*, vol. 598, p. 117825, 2022.

[32] C. J. Lynch and R. Gore, "Application of one-, three-, and seven-day forecasts during early onset on the COVID-19 epidemic dataset using moving average, autoregressive, autoregressive moving average, autoregressive integrated moving average, and naïve forecasting methods," *Data in Brief*, vol. 35, p. 106759, 2021.

[33] D. Liu, L. Y. Wu, B. Li, F. Boussaid, M. Bennamoun, X. Xie, and C. Liang, "Jacobian norm with Selective Input Gradient Regularization for interpretable adversarial defense," *Pattern Recognition*, vol. 145, p. 109902, 2024.

[34] K. E. ArunKumar, D. V. Kalaga, C. M. S. Kumar, M. Kawaji, and T. M. Brenza, "Forecasting of COVID-19 using deep layer recurrent neural networks (RNNs) with gated recurrent units (GRUs) and long short-term memory (LSTM) cells," *Chaos, Solitons & Fractals*, vol. 146, p. 110861, 2021.

[35] D. Huang, Y. Fu, N. Qin, and S. Gao, "Fault diagnosis of high-speed train bogie based on LSTM neural network," *Sci. Chin. Inf. Sci*, vol. 64, pp. 1–3, 2021.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *NeurIPS*, vol. 30, 2017.

[37] J. Hu, Z. Hu, T. Li, and S. Du, "A contrastive learning based universal representation for time series forecasting," *Information Sciences*, vol. 635, pp. 86–98, 2023.

[38] S. Liu, H. Ji, and M. C. Wang, "Nonpooling Convolutional Neural Network Forecasting for Seasonal Time Series With Trends," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 2879–2888, 2020.

[39] G. Lai, W. Chang, Y. Yang, and H. Liu, "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks," in *SIGIR*, 2018, pp. 95–104.

[40] M. LIU, A. Zeng, M. Chen, Z. Xu, Q. LAI, L. Ma, and Q. Xu, "SCINet: Time Series Modeling and Forecasting with Sample Convolution and Interaction," in *NeurIPS*, vol. 35, 2022, pp. 5816–5828.

[41] Y. L. Wu, D. Liu, X. Guo, R. Hong, L. Liu, and R. Zhang, "Multi-scale Spatial Representation Learning via Recursive Hermite Polynomial Networks." in *IJCAI*, 2022, pp. 1465–1473.