DGraFormer: Dynamic Graph Learning Guided Multi-Scale Transformer for Multivariate Time Series Forecasting

Han Yan[1], **Dongliang Chen**[1], **Guiyuan Jiang**[1], **Bin Wang**[1], **Lei Cao**[2], **Junyu Dong**[1] and **Yanwei Yu**[1*]
[1]Faculty of Information Science and Engineering, Ocean University of China
[2]The Department of Computer Science, University of Arizona
yhan@stu.ouc.edu.cn, {chendongliang,jiangguiyuan,wangbin9545}@ouc.edu.cn, caolei@arizona.edu, {dongjunyu,yuyanwei}@ouc.edu.cn

Abstract

Multivariate time series forecasting is a critical focus across many fields. Existing transformer-based models have overlooked the explicit modeling of inter-variable correlations. Similarly, the graph-based methods have also failed to address the dynamic nature of multivariate correlations and the noise in correlation modeling. To overcome these challenges, we propose a novel **D**ynamic **Gra**ph Learning Guided Multi-Scale Trans**former** (DGraFormer) for multivariate time series forecasting. Specifically, our method consists of two main components: Dynamic correlation-aware graph Learning (DCGL) and multi-scale temporal transformer (MTT). The former aims to capture dynamic correlations across different time windows, filters out noise, and selects key weights to guide the aggregation of relevant feature representations. The latter can effectively extract temporal patterns from patch data at varying scales. Finally, the proposed method can capture rich local correlation graph structures and multi-scale global temporal features. Experimental results demonstrate that DGraFormer significantly outperforms existing state-of-the-art models on ten real-world datasets, achieving the best performance across multiple evaluation metrics. The code is available at https://github.com/yh-Hanniel/DGraFormer.

1 Introduction

In the context of the digital era, time series forecasting has become a crucial focus for businesses and organizations, offering significant applications across various domains, including finance [Cao, 2022], energy [Zhu *et al.*, 2023], meteorology [Zhao *et al.*, 2024], transportation [Cao *et al.*, 2025]. These applications often involve vast amounts of multivariate time series data. *Multivariate time series* refers to multiple sequences of data evolving over time, where each sequence represents a specific time-dependent variable. These sequences often show inherent interdependencies or correlations. By analyzing and modeling historical data, organizations can predict trends, detect anomalies, and facilitate data-driven decision-making, ultimately optimizing economic activities and resource allocation.

[*]Corresponding author: Yanwei Yu.

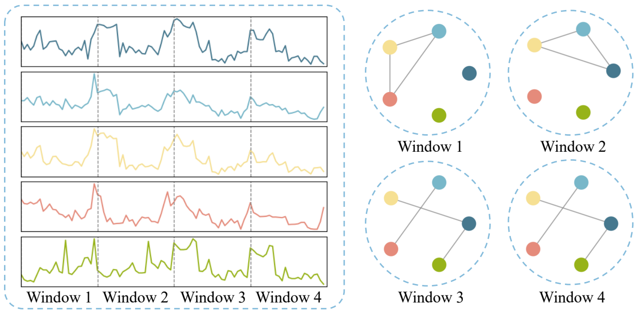

Figure 1: Dynamic and complex correlations of variables across varying time windows. The left shows the time series data divided into time windows, while the right illustrates the adjacency graphs corresponding to the top three weighted edges for each time window.

Deep learning models like Recurrent Neural Networks [Chen *et al.*, 2021] and Long Short-Term Memory networks [Zhao *et al.*, 2017] excel at processing nonlinear, nonstationary time series data with time lags and seasonality, which has promoted the development of time series forecasting. Recently, Transformer achieves outstanding success across various fields [Karita *et al.*, 2019; Brown *et al.*, 2020; Chang *et al.*, 2024], and it shows great promise in time series forecasting as well [Chen *et al.*, 2024; Zhou *et al.*, 2024b].

Although a few transformer-based methods have been proposed for multivariate time series forecasting, they have neglected the explicit modeling of correlations between variables to better explore their complex relationships. For instance, PatchTST [Nie *et al.*, 2023] adopts a channel independence approach, treating each variable separately and intentionally ignoring the complex correlations between them. Thus, a key challenge in advancing multivariate time series forecasting is how to capture these inter-variable correlations better. Graph Neural Networks offer an effective solution by representing multivariate time series as graphs, where variables are nodes and edge weights indicate inter-variable correlations [Chen *et al.*, 2023]. Multivariate time series forecasting faces a challenge for obtaining a good dynamic graph representation due to the dynamic and complex relationships among variables. Generally, existing methods have made some progress in correlation modeling, but two issues remain.

Dynamic nature of multivariate correlations: The relationships between variables are not static, and they can evolve over time, showing periodic or trend-based changes. As shown in Fig. 1, by treating time points as a time window, we observe that the correlation similarities vary across time windows. By learning from real sequence data, we can capture the dynamic relationships that evolve across different windows. Existing method (e.g., MTGNN [Wu *et al.*, 2020]) has used a single graph to propagate correlation information across the entire time scale, and it fails to capture the varying correlation weights across different time windows. This limits the accuracy of feature aggregation based on overall sequence correlation, thereby affecting prediction precision.

Noise caused by negative relationships: Although existing methods (e.g., MSGNet [Cai *et al.*, 2024]) have explored learnable graph structures, they often struggle to learn the optimal structure. This difficulty arises because real-world datasets are typically influenced by various unforeseen factors, which can cause uncorrelated variables to exhibit spurious similarities. As a result, these variables might propagate negative relationships (noise information) during the message-passing process, leading to model performance degradation. The noise can hinder the effective transmission of critical information between key variables, ultimately reducing the ability of model for capturing essential patterns.

To address the challenges outlined above, we propose a **D**ynamic **Gra**ph Learning Guided Multi-Scale Trans**former** (DGraFormer), which comprises two main components: Dynamic Correlation-aware Graph Learning (DCGL) and Multi-Scale Temporal Transformer (MTT). The method views time series data as multiple time windows with stable correlations. It adaptively learns the graph adjacency matrices for each time window, filters out noise, and selects the most crucial correlation weights to guide the aggregation of feature representations based on correlations. This approach significantly improves the ability of the model to adapt to dynamic correlation changes and overcomes the issue of redundant information propagation caused by external noise in existing methods. Our contributions are summarized in three main aspects:

- We propose a novel multivariate time series forecasting model, named DGraFormer, which combines DCGL and MTT in a unified framework to simultaneously achieve the dynamic correlation-aware graph learning and multi-scale temporal information interaction.
- A DCGL is designed to adaptively capture the dynamic correlations among multivariate variables in temporal windows, and then select essential information to guide the process of correlation feature aggregation, effectively eliminating noise in the weights and learning critical corresponding adjacency weights for dynamic graph learning.
- Extensive experiments on ten real-world datasets demonstrate that DGraFormer outperforms SOTA multivariate time series forecasting models. Compared to the latest graph-based model, MSGNet, our model achieves average reductions of 11% in MSE and 10.6% in MAE.

2 Related Work

2.1 Time Series Forecasting

Time series forecasting has been extensively studied. Recently, Transformer [Vaswani, 2017; Kitaev *et al.*, 2020; Tang and Matteson, 2021] shows superior performance in semantic extraction and long-range dependency modeling, outperforming methods based on TCNs [Bai *et al.*, 2018; Liu *et al.*, 2022a; Lai *et al.*, 2023] and RNNs [Rangapuram *et al.*, 2018; Qin *et al.*, 2017]. This trend drives interest in Transformer-based models. Informer [Zhou *et al.*, 2021] has led the adoption of Transformer for time series forecasting. Autoformer [Wu *et al.*, 2021] introduces seasonal trend decomposition, while FEDformer [Zhou *et al.*, 2022] extends this approach by transforming the time domain into the frequency domain for decomposition learning. Pyraformer [Liu *et al.*, 2022b] employs convolution to capture multi-scale features, and Non-stationary Transformers [Liu *et al.*, 2022e] focus on time series stationarization.

Despite challenges raised by linear models [Zeng *et al.*, 2023; Das *et al.*, 2023], Transformers remain top-performing in time series forecasting [Jiang *et al.*, 2023]. PatchTST [Nie *et al.*, 2023] leverages channel independence by representing each variable as a separate channel and segmenting it into subsequence-level patches, which are subsequently used as tokens for the Transformer model. In contrast, Crossformer [Zhang and Yan, 2023] adopts variable patching and a hierarchical encoder-decoder to capture cross-dimensional dependencies. Recent work [Zhou *et al.*, 2024a] proposes inputting the same number of auxiliary variables for each variable into the Transformer for learning. And iTransformer [Liu *et al.*, 2024] encodes variables independently as distinct tokens, using attention for inter-variable and feed-forward networks for temporal dependencies. Further techniques are under exploration [Pan *et al.*, 2021; Deng *et al.*, 2022; Li *et al.*, 2023; Liu *et al.*, 2023; Deng *et al.*, 2024a; Deng *et al.*, 2024b], including TimesNet [Wu *et al.*, 2023], which projects one-dimensional time series data into a two-dimensional space and analyzes it with sophisticated visual networks.

Most existing methods overlook the modeling of correlations between variables. Existing methods for learning variable correlations fail to account for the dynamic nature of multivariate correlations and the noise in correlation modeling. These approaches propagate information across all variables, while neglecting isolated variables (those not correlated with others).

2.2 Graph Modeling for Multivariate Time Series Forecasting

The use of GNNs [Kipf and Welling, 2016; Defferrard *et al.*, 2016; Abu-El-Haija *et al.*, 2019] introduces a new perspective to model multivariate time series data. However, most existing approaches apply them to specific domains, such as traffic [Yu *et al.*, 2017; Li *et al.*, 2017; Wu *et al.*, 2019; Wen *et al.*, 2023; Liu *et al.*, 2022c], where prior knowledge (e.g., Euclidean distance or POI) can naturally define graph structures. In contrast, general multivariate time series fore-

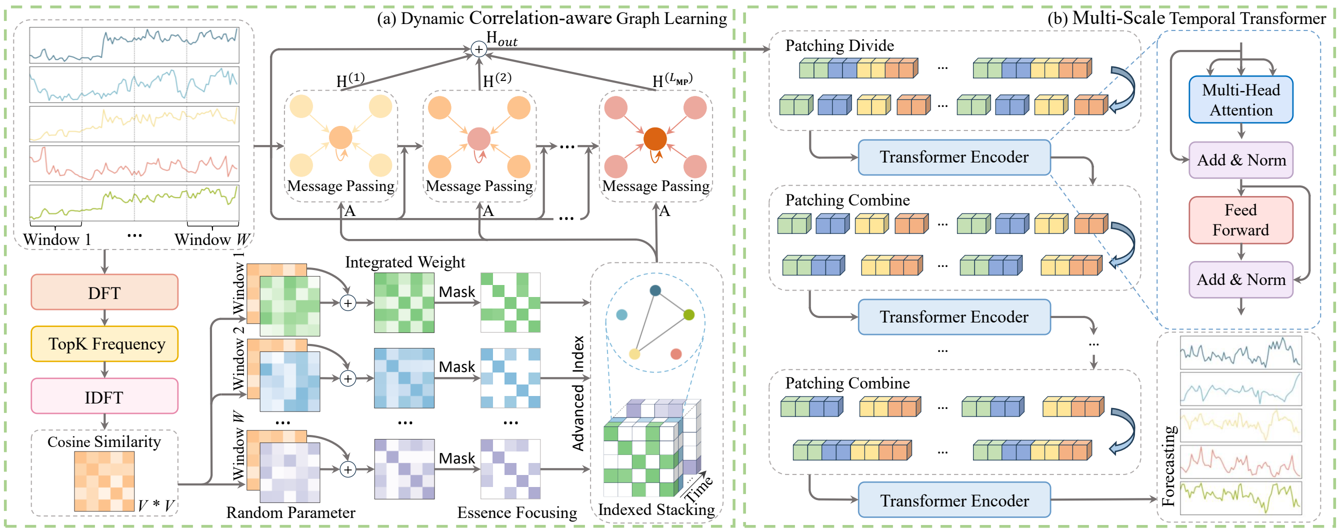

Figure 2: The overall architecture of the proposed DGraFormer.

casting models involve datasets from multiple domains, making it difficult to obtain a unified, predefined graph structure.

To address this, some methods [Wu *et al.*, 2020] have explored learnable graph structures, replacing predefined graphs with learnable parameter matrices. However, methods such as MAGNN [Chen *et al.*, 2023] and MSGNet [Cai *et al.*, 2024] still employ static graphs to represent time series, which are insufficient to capture the dynamic nature of variable correlations. Although TPGNN [Liu *et al.*, 2022d] captures temporal dynamics through matrix polynomials, it ignores noise caused by external factors. Consequently, effectively learning the core correlations between variables remains a critical challenge in correlation modeling.

3 Problem Definition

For multivariate time series forecasting, we denote the input data over a look-back window as $\mathbf{X} = \{x_1, \ldots, x_T\} \in \mathbb{R}^{N\times T}$, where T is the number of time steps and N is the number of variables. The objective is to forecast the next τ time steps, represented as $\mathbf{Y} = \{x_{T+1}, \ldots, x_{T+\tau}\} \in \mathbb{R}^{N\times\tau}$.

We model the multivariate time series data using graphs $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E} \in \mathbb{R}^{N\times N}$ is the correlation weight matrix between the nodes. To capture the dynamic correlations between variables, we divide the time series data into W windows, each containing m time steps. The correlation graph structure is then represented as $\mathcal{G} = \{\mathcal{V}, \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_W\}\}$, where $\mathcal{E}_w \in \mathbb{R}^{N\times N}$ is the correlation weight matrix for the w-th time window.

4 Methodology

To effectively capture multivariate correlations, we propose a novel multivariate time series forecasting model called DGraFormer. The overall architecture of DGraFormer is illustrated in Fig. 2, comprising two main components: Dynamic correlation-aware graph Learning (DCGL) and Multi-Scale Temporal Transformer (MTT).

First, the DCGL generates the corresponding graph structure matrices with dynamic essential multivariate correlations for the input sequence. Then it takes the instance-normalized data and its corresponding edge weight matrices to learn the correlation representations. The MTT extracts temporal representations from the patch data at varying scales. Finally, the learned representations, which encapsulate both correlation and temporal information, are passed through the Flatten and Linear layers in the predictor to produce the final forecast output.

4.1 Dynamic Correlation-aware Graph Learning

The Dynamic Correlation-aware Graph Learning is designed to capture graph structures that reflect dynamic correlations and focus on essential correlation weights. We construct the weight matrix $\mathbf{A} \in \mathbb{R}^{N\times N\times T}$, which guides the propagation of the correlation information for the input data $\mathbf{X}$.

Dynamic Multivariate Correlation Weight Learning

Existing methods have modeled the entire time series with a single graph, which failed to capture dynamic changes in multivariate correlations. To overcome this limitation, and enable more precise correlation modeling, we divide the time series into W time windows, each containing m data points. For each time window, we independently learn its correlation graph weights.

First, we compute the overall correlation weights based on the information from the entire dataset used for training. It represents the general relationships between variables in the training set $\mathbf{X}_{\text{all}} \in \mathbb{R}^{N\times M}$, where M denotes the total number of time steps in the training set. Raw data is often influenced by trends and external factors, which complicates the extraction of the underlying correlations among variables masked by the overall trend. To mitigate these influences, we transform the data from the time domain to the frequency domain, removing trend components and extracting seasonal information. Specifically, we decompose $\mathbf{X}_{\text{all}}$ into Fourier bases using the Discrete Fourier Transform (DFT$(\cdot)$), select the top K_f amplitudes, and apply the Inverse Discrete Fourier Transform (IDFT$(\cdot)$) to obtain the seasonal representation of the data, denoted as $\mathbf{X}_{\text{sea}} \in \mathbb{R}^{N \times M}$:

$$\mathbf{X}_{\text{sea}} = \text{IDFT}\left(\arg\text{top}_{K_f}\left(|\text{DFT}(\mathbf{X}_{all})|\right), \mathcal{A}, \Phi\right), \quad (1)$$

where $\mathcal{A}$ is the set of amplitudes for all selected frequency components, and Φ is the set of phase angles for all selected frequency components. Using $\mathbf{X}_{\text{sea}}$, we calculate the cosine similarity between all pairs of variables, which results in the overall matrix of correlation weights $\mathbf{C} \in \mathbb{R}^{N \times N}$:

$$\mathbf{C} = \frac{\mathbf{X}_{\text{sea}}\mathbf{X}_{\text{sea}}^\top}{\|\mathbf{X}_{\text{sea}}\|\|\mathbf{X}_{\text{sea}}\|^\top}. \quad (2)$$

Secondly, we generate a unique parametric correlation matrix for each time window and optimize it during the learning process. This enables precise modeling of variable correlations by capturing the fine-grained correlation graph structures within different time windows. Specifically, for the matrix of weights $\mathbf{R}_w \in \mathbb{R}^{N \times N}$ of the w-th time window, it is computed as follows:

$$\begin{aligned}(\mathbf{F}_w)_1 &= \text{Linear}\big((\mathbf{E}_w)_1\big),\\ (\mathbf{F}_w)_2 &= \text{Linear}\big((\mathbf{E}_w)_2\big),\\ \mathbf{R}_w &= \text{ReLU}\big(tanh((\mathbf{F}_w)_1(\mathbf{F}_w)_2^\top\big),\end{aligned} \quad (3)$$

where $(\mathbf{E}_w)_1$ and $(\mathbf{E}_w)_2 \in \mathbb{R}^{N \times h}$ denote randomly initialized node embeddings (with each row corresponding to a variable). The subscripts 1 and 2 indicate different initialization instances, both of which are learnable during training. h is the embedding dimensionality.

Finally, we aggregate $\mathbf{C}$ and $\mathbf{R}_w$ to derive the correlation graph weights matrix $\mathcal{E}_w \in \mathbb{R}^{N \times N}$ for the w-th time window:

$$\mathcal{E}_w = \alpha\mathbf{C} + (1-\alpha)\mathbf{R}_w, \alpha \in [0.1, 0.9], \quad (4)$$

where α governs the relative contribution of $\mathbf{C}$ to $\mathcal{E}_w$. Initially, α is set to 0.9 and is gradually decreased to 0.1 as training progresses. This adjustment ensures that the parameterized weights matrix $\mathbf{R}_w$ progressively takes precedence over $\mathbf{C}$ in $\mathcal{E}_w$, gradually reflecting the unique correlations among variables in the w-th time window.

Essential Correlation Information Focusing

To eliminate redundant information that may interfere with the learning of correlation features, such as temporary false correlations induced by external factors, and to focus on the essential correlation weights, we focus essential correlation information to $\mathcal{E}_w$ for each time window. Specifically, we generate a corresponding mask matrix $\mathbf{M}_w \in \mathbb{R}^{N \times N}$ for the w-th time window by selecting the top K_e values from the vectorized form of $\mathcal{E}_w$:

$$\begin{aligned}\text{indices} &= \arg\text{top}_{K_e}\big(\text{vec}(\mathcal{E}_w)\big),\\ \mathbf{M}_w &= \text{reshape}\left(\mathbf{1}_{\text{indices}}, N, N\right).\end{aligned} \quad (5)$$

Given the learnable nature of $\mathcal{E}_w$, $\mathbf{M}_w$ evolves dynamically during training, adaptively guiding the model in selecting the essential correlation weights. We apply the mask matrix to $\mathcal{E}_w$ as follows $\mathcal{E}_w$:

$$\tilde{\mathcal{E}}_w = \mathbf{M}_w \odot \mathcal{E}_w, \quad (6)$$

where $\tilde{\mathcal{E}}_w \in \mathbb{R}^{N \times N}$ denotes the sparsified correlation weight matrix for the w-th time window, and $\odot$ indicates element-wise multiplication. We apply the above operation to the correlation edge weight matrix of each time window, obtaining a set of dynamically sparsified correlation edge weight matrices $\tilde{\mathcal{E}} = \{\tilde{\mathcal{E}}_1, \tilde{\mathcal{E}}_2, \ldots, \tilde{\mathcal{E}}_W\} \in \mathbb{R}^{N \times N \times W}$.

For each input data $\mathbf{X} \in \mathbb{R}^{N \times T}$, we apply advanced indexing to extract the weight matrix $\mathbf{A} \in \mathbb{R}^{N \times N \times T}$ for correlation feature learning:

$$\mathbf{A}[:,:,t] = \tilde{\mathcal{E}}[:,:,\mathbf{I}[t]], \quad \text{for } t = 1, 2, \ldots, T, \quad (7)$$

where $\mathbf{I} \in \mathbb{R}^{T \times 1}$ denotes the index information of the time window for each time point. We then feed $\mathbf{A}$ into the GCN-based correlation fusion module to guide the information propagation process.

Correlation-aware Graph Message Passing

To propagate information based on the output weight matrix $\mathbf{A}$, we use input data processed with Instance Norm, which is a technique used to mitigate distribution shifts between training and testing data, to learn feature representations that aggregate essential correlations. Inspired by MTGNN [Wu *et al.*, 2020], we adopt a graph convolution approach that incorporates residual connections and the mix-hop concept for learning correlation representations. The information propagation step is defined as follows:

$$\mathbf{H}^{(l)} = \beta\mathbf{H}^{(0)} + (1-\beta)\mathbf{A}\mathbf{H}^{(l-1)}, \quad (8)$$

where $\mathbf{H}^{(0)}$ denotes the input data $\mathbf{X}$ processed with Instance Normalization and feature dimensional upsampling, and $\mathbf{H}^{(l)}$ represents the result of the l-th layer of message passing. β controls the proportion of the root node's original state retained, ensuring that the propagated node states explore the depth of the neighborhood while preserving locality. We use the following approach for information selection:

$$\mathbf{H}_{\text{out}} = \text{Concat}(\mathbf{H}^{(0)}, \mathbf{H}^{(1)}, \ldots, \mathbf{H}^{(L_{\text{MP}})}) \cdot \mathbf{W}, \quad (9)$$

where the learnable parameter matrix $\mathbf{W}$ acts as a feature selector, and L_{MP} represents the total number of layers for message passing. It adaptively selects the message passing features at different depths to generate the output $\mathbf{H}_{\text{out}} \in \mathbb{R}^{N \times T}$ of the correlation fusion module.

4.2 Multi-Scale Temporal Transformer

The Multi-Scale Temporal Transformer is designed to capture the temporal patterns within sequence data. Existing methods, such as PatchTST [Nie *et al.*, 2023], have shown that dividing time series into patches and using them as tokens in a Transformer Encoder for feature learning can significantly improve the accuracy of multivariate time series prediction. Aggregating time steps into subsequence-level patches enhances locality and captures comprehensive semantic information that cannot be obtained at the point level. However, using a single patch size to divide the time series overlooks the features present at different time resolutions. To address this, we adopt a multi-scale patch division and combination strategy to model temporal correlations.

Patch Division

For the output $\mathbf{H}_{\text{out}} \in \mathbb{R}^{N\times T}$ of the correlation fusion module, we divide the i-th univariate time series $\mathbf{h}^i_{\text{out}} \in \mathbb{R}^{1\times T}$ into $S = T/P$ non-overlapping patches, where P represents the number of time steps in each patch, denoted as $\mathbf{h}^i_p \in \mathbb{R}^{P\times S}$. We then apply a trainable linear projection $\mathbf{W}_p \in \mathbb{R}^{D\times P}$ to map the patches to the Transformer latent space of dimension D. In addition, we use a learnable additive position encoding $\mathbf{W}_{pos} \in \mathbb{R}^{D\times S}$ to capture the temporal order of the patches:

$$\bar{\mathbf{x}}^i = \mathbf{W}_p\mathbf{h}^i_p + \mathbf{W}_{\text{pos}}, \tag{10}$$

where $\bar{\mathbf{x}}^i \in \mathbb{R}^{D\times S}$ denotes the univariate input to the Transformer encoder.

Transformer Encoder

The structure of our transformer encoder follows the standard vanilla Transformer architecture. Initially, the multi-head self-attention mechanism is used to model the temporal correlations between different patches. By applying a linear transformation to the input $\bar{\mathbf{x}}^i$, we obtain the query, key, and value matrices, denoted as $\mathbf{Q}^i, \mathbf{K}^i, \mathbf{V}^i$. The attention output $\mathbf{z}^i_a \in \mathbb{R}^{D\times S}$ is then computed as follows:

$$\mathbf{z}^i_a = \text{softmax}\left(\frac{\mathbf{Q}^i{\mathbf{K}^i}^T}{\sqrt{d_k}}\right)\mathbf{V}^i. \tag{11}$$

Subsequently, the attention output is processed through the Add & Norm layer, which integrates residual connections and batch normalization to improve training stability and convergence. This operation can be mathematically expressed as:

$$\mathbf{z}^i_{\text{add}} = \text{Norm}(\mathbf{z}^i_{\text{a}} + \bar{\mathbf{x}}^i), \tag{12}$$

where Norm$(\cdot)$ denotes the BatchNorm. Following the Add & Norm layer, the intermediate representation is further refined using a position-wise feed-forward network (FFN). The FFN is composed of two linear transformations separated by a non-linear activation function. The output of this stage is given by:

$$\mathbf{z}^i_{\text{ffn}} = \text{Norm}\big(\text{FFN}(\mathbf{z}^i_{\text{add}}) + \mathbf{z}^i_{\text{add}}\big). \tag{13}$$

As a result, the transformer encoder produces the multivariate output $\mathbf{Z} \in \mathbb{R}^{N\times D\times S}$, which encapsulates the temporal feature representation of the input sequence.

Patch Combination and Multi-Scale Encoding

To capture the diverse features embedded in multi-scale patches, we combine neighboring patches pairwise and input them into another identical Transformer encoder for further learning. This process is represented as:

$$\begin{gathered}\mathbf{X}^{(l)}_p = \text{reshape}\big(\mathbf{Z}^{(l-1)}, [N, 2D^{(l-1)}, \frac{S^{(l-1)}}{2}]\big),\\ \mathbf{Z}^{(l)} = \text{TE}^{(l)}(\mathbf{X}^{(l)}_p),\end{gathered} \tag{14}$$

where $\mathbf{Z}^{(l)} \in \mathbb{R}^{N\times D^{(l)}\times S^{(l)}}$ represents the output of the l-th layer of the transformer encoder TE$^{(l)}(\cdot)$, which includes multi-head attention, BatchNorm layers, and feed-forward network. The input to the first encoder TE$^0(\cdot)$ is denoted as $\mathbf{X}^{(0)}_p$, and its output as $\mathbf{Z}^{(0)}$. Through multiple iterations, we learn feature representations that capture information across different time resolutions. Finally, After being processed by L_{TE} layers of Transformer encoder, the output of the last Transformer encoder is flattened, and a Linear layer is applied to generate the final prediction result $\mathbf{Y} = \{x_{T+1}, \ldots, x_{T+\tau}\} \in \mathbb{R}^{N\times\tau}$.

5 Experiments

The experiments are designed to address the following research questions:

- **RQ1**: What is the performance of our DGraFormer as compared to various state-of-the-art baselines?
- **RQ2**: How do the key components contribute to the performance?
- **RQ3**: How do the key hyperparameters influence the performance of the proposed DGraFormer?

5.1 Datasets

In our experimental evaluation, we employ 10 real-world datasets, including ETT (4 subsets), Weather, Electricity, Solar-Energy, Traffic used by iTransformer [Liu *et al.*, 2024], Flight evaluated in MSGNet [Cai *et al.*, 2024] and AirQualityUCI[1], to assess the performance of all methods.

5.2 Baselines

We compare our DGraFormer against with four categories of baselines: (1) Graph-based method: MSGNet [Cai *et al.*, 2024]; (2) Transformer-based methods: iTransformer [Liu *et al.*, 2024], PatchTST [Nie *et al.*, 2023], Crossformer [Zhang and Yan, 2023], Pyraformer [Liu *et al.*, 2022b]; (3) Linear-based methods: DLinear [Zeng *et al.*, 2023], TiDE [Das *et al.*, 2023]; (4) TCN-based method: TimesNet [Wu *et al.*, 2023].

5.3 Experiment Settings and Evaluation Metrics

DGraFormer uses the Adam optimizer and a combination of L1 and L2 loss for model training. The number L_{MP} of graph message passing layers is set to 2, and the number L_{TE} of the Transformer encoder is set to 3. Additionally, a patch length p = 8 is used for division. The number of time points m in each time window depends on the dataset's time frequency, corresponding to the number of data points in one day (e.g., 24 for Electricity, 144 for Solar). The weight focusing ratio K_e ranges from 0.5 to 0.005 based on the number of variables in the dataset, decreasing as the number of variables increases. And we set K_f=3 in Eq. (1). All experiments are conducted using PyTorch and executed on an NVIDIA GeForce RTX 4090 24GB GPU. To ensure fair comparisons, all models are configured with the same look-back window T = 96 and prediction lengths $\tau \in \{96, 192, 336, 720\}$ for all datasets. The hyperparameters for the baseline models are set according to the configurations provided in the original papers. We evaluate model performance using two common metrics in multivariate time series forecasting: Mean Absolute Error (MAE) and Mean Squared Error (MSE).

[1]https://github.com/Gauhar1107/AirQualityUCI

Models		**Ours**		iTransformer (ICLR2024)		MSGNet (AAAI2024)		PatchTST (ICLR2023)		TimesNet (ICLR2023)		TIDE (arXiv2023)		Dlinear (AAAI2023)		Crossformer (ICLR2023)		Pyraformer (ICLR2022)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.367**	**0.393**	0.386	0.405	0.390	0.411	0.414	0.419	0.384	0.402	0.479	0.464	0.386	0.400	0.423	0.448	0.664	0.612
	192	0.438	**0.426**	0.441	0.436	0.442	0.442	0.460	0.445	**0.436**	0.429	0.525	0.492	0.437	0.432	0.471	0.474	0.790	0.681
	336	**0.479**	**0.442**	0.487	0.458	0.480	0.468	0.501	0.466	0.491	0.469	0.565	0.515	0.481	0.459	0.570	0.546	0.891	0.738
	720	**0.484**	**0.467**	0.503	0.491	0.494	0.488	0.500	0.488	0.521	0.500	0.594	0.558	0.519	0.516	0.653	0.622	0.963	0.782
ETTh2	96	0.300	**0.344**	**0.297**	0.349	0.328	0.371	0.302	0.348	0.340	0.374	0.400	0.440	0.333	0.387	0.745	0.584	0.645	0.597
	192	0.382	**0.397**	**0.380**	0.400	0.402	0.414	0.388	0.400	0.402	0.414	0.528	0.509	0.477	0.476	0.877	0.656	0.788	0.683
	336	**0.388**	**0.415**	0.428	0.432	0.435	0.443	0.426	0.433	0.452	0.452	0.643	0.517	0.594	0.541	1.043	0.7731	0.907	0.747
	720	0.418	**0.436**	0.427	0.445	**0.417**	0.441	0.431	0.446	0.462	0.468	0.874	0.679	0.831	0.657	1.104	0.763	0.963	0.783
ETTm1	96	0.326	**0.366**	0.334	0.368	**0.319**	**0.366**	0.329	0.367	0.338	0.375	0.364	0.387	0.345	0.372	0.404	0.426	0.543	0.510
	192	**0.358**	**0.376**	0.377	0.391	0.376	0.397	0.367	0.385	0.374	0.387	0.398	0.404	0.380	0.389	0.450	0.451	0.557	0.537
	336	**0.387**	**0.397**	0.426	0.420	0.417	0.422	0.399	0.410	0.410	0.411	0.428	0.425	0.413	0.413	0.532	0.515	0.754	0.655
	720	**0.446**	**0.436**	0.491	0.459	0.481	0.458	0.454	0.439	0.478	0.450	0.487	0.461	0.474	0.453	0.666	0.589	0.908	0.724
ETTm2	96	**0.173**	**0.253**	0.180	0.264	0.177	0.262	0.175	0.259	0.187	0.267	0.207	0.305	0.193	0.292	0.287	0.366	0.435	0.507
	192	0.243	**0.302**	0.250	0.309	0.247	0.307	**0.241**	**0.302**	0.249	0.309	0.290	0.364	0.284	0.362	0.414	0.492	0.730	0.673
	336	0.315	0.348	0.311	0.348	0.312	0.346	**0.305**	**0.343**	0.321	0.351	0.377	0.422	0.369	0.427	0.597	0.542	1.201	0.845
	720	0.404	0.401	0.412	0.407	0.414	0.403	**0.402**	**0.400**	0.408	0.403	0.558	0.524	0.554	0.522	1.730	1.042	3.625	1.451
Weather	96	0.168	**0.207**	0.174	0.214	0.163	0.212	0.177	0.218	0.172	0.220	0.202	0.261	0.196	0.255	**0.158**	0.230	0.896	0.556
	192	0.213	**0.249**	0.221	0.254	0.212	0.254	0.225	0.259	0.219	0.261	0.242	0.298	0.237	0.296	**0.206**	0.277	0.622	0.624
	336	**0.270**	**0.291**	0.278	0.296	0.272	0.299	0.278	0.297	0.280	0.306	0.287	0.335	0.283	0.335	0.272	0.335	0.739	0.753
	720	**0.345**	**0.338**	0.358	0.349	0.350	0.348	0.354	0.348	0.365	0.359	0.351	0.386	**0.345**	0.381	0.398	0.418	1.004	0.934
Electricity	96	**0.136**	**0.229**	0.148	0.240	0.165	0.274	0.195	0.285	0.168	0.272	0.237	0.329	0.197	0.282	0.219	0.314	0.386	0.449
	192	**0.155**	**0.244**	0.162	0.253	0.184	0.292	0.199	0.289	0.184	0.289	0.236	0.330	0.196	0.285	0.231	0.322	0.386	0.443
	336	**0.171**	**0.261**	0.178	0.269	0.195	0.302	0.215	0.305	0.198	0.300	0.249	0.344	0.209	0.301	0.246	0.337	0.378	0.443
	720	**0.210**	**0.298**	0.225	0.317	0.231	0.332	0.256	0.337	0.220	0.320	0.284	0.373	0.245	0.333	0.280	0.363	0.376	0.445
Solar	96	**0.184**	**0.219**	0.203	0.237	0.259	0.285	0.234	0.286	0.250	0.292	0.312	0.399	0.290	0.378	0.310	0.331	0.218	0.274
	192	**0.211**	**0.234**	0.233	0.261	0.268	0.293	0.267	0.310	0.296	0.318	0.339	0.416	0.320	0.398	0.734	0.725	0.227	0.284
	336	**0.234**	**0.250**	0.248	0.273	0.316	0.326	0.290	0.315	0.319	0.330	0.368	0.430	0.353	0.415	0.750	0.735	0.246	0.296
	720	**0.235**	**0.254**	0.249	0.275	0.313	0.326	0.289	0.317	0.388	0.337	0.370	0.425	0.356	0.413	0.769	0.765	0.252	0.296
Traffic	96	0.442	**0.245**	**0.395**	0.268	0.598	0.339	0.544	0.359	0.593	0.321	0.805	0.493	0.650	0.396	0.522	0.290	2.085	0.468
	192	0.469	**0.257**	**0.417**	0.276	0.616	0.358	0.540	0.354	0.617	0.336	0.756	0.474	0.598	0.370	0.530	0.293	0.867	0.467
	336	0.481	**0.266**	**0.433**	0.283	0.651	0.373	0.511	0.358	0.629	0.336	0.762	0.477	0.605	0.373	0.558	0.305	0.869	0.469
	720	0.510	**0.281**	**0.467**	0.302	0.699	0.404	0.586	0.375	0.640	0.350	0.719	0.449	0.645	0.394	0.589	0.328	0.881	0.473
Flight	96	**0.143**	**0.250**	0.144	0.252	0.183	0.301	0.175	0.305	0.237	0.350	0.225	0.341	0.221	0.337	0.151	0.259	0.229	0.343
	192	**0.143**	**0.247**	0.147	0.253	0.189	0.306	0.176	0.304	0.224	0.337	0.225	0.340	0.220	0.336	0.159	0.264	0.248	0.357
	336	**0.154**	**0.257**	0.159	0.266	0.206	0.320	0.187	0.314	0.289	0.394	0.235	0.347	0.229	0.342	0.178	0.285	0.329	0.413
	720	**0.191**	**0.291**	0.193	0.297	0.253	0.358	0.228	0.343	0.310	0.408	0.272	0.373	0.263	0.366	0.203	0.307	0.656	0.606
AirQualityUCI	96	**1.147**	**0.580**	1.192	0.601	1.232	0.616	1.240	0.620	1.203	0.607	1.309	0.650	1.303	0.682	1.177	0.621	1.350	0.750
	192	**1.238**	**0.606**	1.308	0.633	1.350	0.658	1.337	0.644	1.330	0.645	1.472	0.688	1.385	0.702	1.321	0.665	1.607	0.824
	336	**1.344**	**0.644**	1.401	0.659	1.490	0.675	1.408	0.660	1.466	0.673	1.572	0.709	1.468	0.722	1.445	0.707	1.600	0.795
	720	**1.482**	**0.696**	1.532	0.708	1.592	0.717	1.609	0.726	1.580	0.704	1.806	0.777	1.599	0.763	1.721	0.813	1.807	0.889

Table 1: Forecast results with 96 look back window and prediction length $\{96, 192, 336, 720\}$. Best results are in bold, followed by underline.

5.4 Experiment Results

Performance Comparison (RQ1)

The results of multivariate time series forecasting are presented in Table 1, DGraFormer achieves optimal performance on most of the evaluation metrics. Compared with 2024 SOTA models iTransformer and MSGNet, DGraFormer reduces MSE and MAE by an average of 6.75% and 7.25%. All models face performance decline with longer prediction lengths due to information decay and error accumulation. Yet DGraFormer consistently outperforms SOTA, e.g., exceeding MSGNet by 10.5%/10.8% (MSE/MAE) at horizon=192, 11.7%/11.0% at 336, and 10.5%/9.8% at 720. This demonstrates its robust capability in long-term forecasting.

Specifically, on high-dimensional datasets (Electricity, Solar, Traffic) with 100+ variables, DGraFormer reduces MAE by 6.6% over iTransformer by (1) capturing richer inter-variable correlations via graph attention and (2) leveraging more contextual information for improved accuracy. Although Transformer-based models (e.g., iTransformer) employ multi-head attention to capture inter-variable correlations, they do not explicitly model multivariate correlations. In contrast, graph-based methods more flexibly model dynamic, complex variable correlations. iTransformer uses shared attention across variables, limiting its ability to capture evolving, heterogeneous correlations in multivariate series.

Despite both using graph-based correlation modeling, DGraFormer outperforms MSGNet by 11% MSE and 10.6% MAE on average. On MSGNet's FLight dataset, DGraFormer reduces MSE by 24% and MAE by 18.7%. While MSGNet learns graph structures across multiple scales, it does not dynamically model within each scale or handle noise effectively. This demonstrates that dynamic modeling and focusing on key correlations can significantly improve forecast accuracy.

Compared with linear-based models TiDE and DLinear, DGraFormer reduces MSE/MAE by 21.7%/17.9%. This further confirms that a well-designed Transformer model can still achieve strong performance in time series forecasting.

Ablation Study (RQ2)

We ablate DGraFormer's components with four variants:

- **w/o Dynamic Time Windows (DTW)**: Using a single graph to learn variable correlations across all time steps.
- **w/o Dynamic Graph Learning (DGL)**: Using only the initial adjacency matrix $\mathbf{C}$ to model variable correlations.
- **w/o Essential Correlation Weight Focusing (ECF)**: Using all positive weights for graph message propagation.
- **w/o Multi-scale Transformer Encoding (MTE)**: Using a fixed uniform patch size across all Transformer encoder layers for temporal learning, without patch combination.

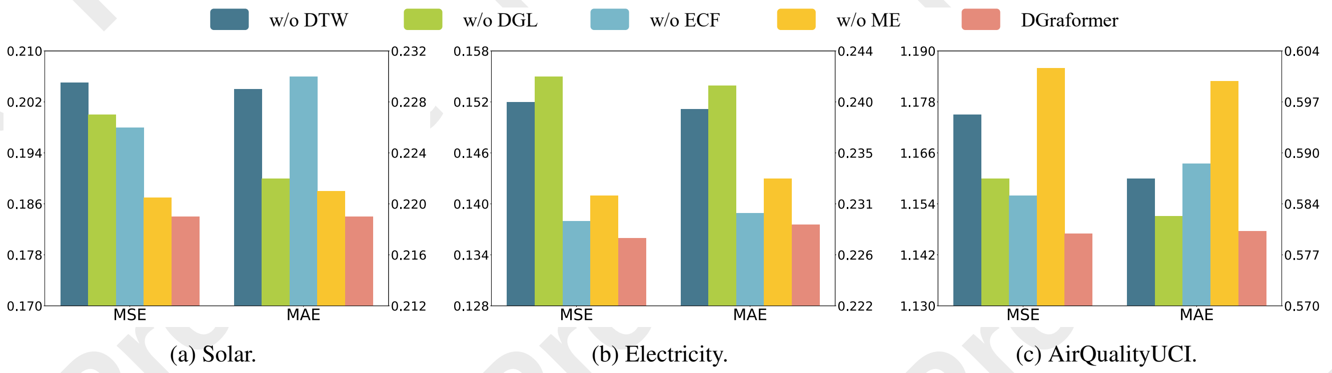
(a) Solar. (b) Electricity. (c) AirQualityUCI.

Figure 3: Performance comparison of DGraFormer and its variants. MSE uses the left y-axis, and MAE uses the right y-axis.

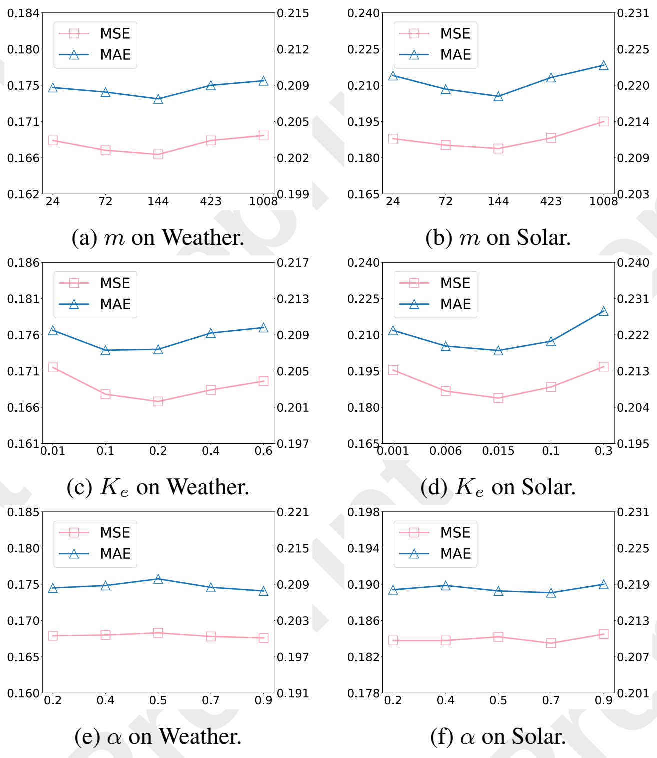
(a) m on Weather. (b) m on Solar. (c) K_e on Weather. (d) K_e on Solar. (e) α on Weather. (f) α on Solar.

Figure 4: Parameter impact of the number m of time points in each time window, the weight focusing ratio K_e, and the cosine similarity matrix proportion α on Weather and Solar datasets. MSE uses the left y-axis, and MAE uses the right y-axis.

Fig. 3 illustrates the predictive performance of different model variants on Solar, Electricity, and AirQualityUCI datasets. Removing the dynamic time window caused a substantial decrease in accuracy, highlighting the importance of modeling dynamic correlations in inter-variable relationships. The removal of dynamic graph learning impairs the model's ability to adaptively capture correlations among variables. Excluding the essential correlation weight focusing resulted in the model's inability to effectively filter out noise, which disrupted the communication of core correlations. Finally, removing multi-scale encoding restricted the model's capacity to capture time features at different scales. Overall, graph learning effectively captures the correlations between variables, while the Transformer excels in learning temporal features across different time scales. The combination of these components results in improved prediction performance.

Parameter Sensitivity Analysis (RQ3)

We analyze the parameter sensitivity of DGraFormer with respect to three key hyperparameters, the number m of time points, the weight focusing ratio K_e, and the cosine similarity matrix proportion α on Weather and Solar datasets.

Fig. 4a and 4b show the prediction performance for varying values of m, the number of time points in each time window. The model performs best when $m = 144$, corresponding to a full day, as correlations within a single day may tend to be stable. Larger time windows, however, introduce varying correlations, reducing performance.

Fig. 4c and 4d assess the impact of the weight focusing ratio K_e. As K_e decreases, performance improves up to a point before declining. A moderate decrease helps eliminate redundant noise, preserving essential correlations for accurate feature aggregation. However, too low a K_e discards important weights, weakening model performance.

Fig. 4e and 4f assess the cosine similarity matrix proportion α. Performance remains stable across α values, as graph structure learning adaptively adjusts weights to mitigate fluctuations. The cosine similarity matrix helps prevent random initialization fitting issues and guides the model to adjust based on a correlation matrix that reflects seasonal patterns. Graph learning compensates for changes introduced by varying retention ratios.

6 Conclusion

In this paper, we propose the **D**ynamic **Gra**ph Learning Guided Multi-Scale Trans**former** (DGraFormer) to address challenges in multivariate time series forecasting. The model integrates two key components: Dynamic Correlation-aware Graph Learning (DCGL), which captures dynamic correlations and filters noise, and Multi-Scale Temporal Feature Learning (MTF), which extracts temporal patterns at multiple scales. Experimental results demonstrate that DGraFormer outperforms sota methods on ten real-world datasets, achieving superior performance.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under Grant Nos. 62176243, 62372421, and 62402463, the Fundamental Research Funds for the Central Universities under Grant No. 202442005, and the National Key R&D Program of China under Grant No. 2022ZD0117201.

References

[Abu-El-Haija *et al.*, 2019] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, pages 21–29, 2019.

[Bai *et al.*, 2018] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv*, 2018.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurlPS*, 33:1877–1901, 2020.

[Cai *et al.*, 2024] Wanlin Cai, Yuxuan Liang, Xianggen Liu, Jianshuai Feng, and Yuankai Wu. Msgnet: Learning multi-scale inter-series correlations for multivariate time series forecasting. In *AAAI*, volume 38, pages 11141–11149, 2024.

[Cao *et al.*, 2025] Lingxiao Cao, Bin Wang, Guiyuan Jiang, Yanwei Yu, and Junyu Dong. Spatiotemporal-aware trend-seasonality decomposition network for traffic flow forecasting. In *AAAI*, volume 39, pages 11463–11471, 2025.

[Cao, 2022] Longbing Cao. Ai in finance: challenges, techniques, and opportunities. *ACM Computing Surveys (CSUR)*, 55(3):1–38, 2022.

[Chang *et al.*, 2024] Zhiyong Chang, Mingjun Yin, and Yan Wang. Coatformer: vision transformer with composite attention. In *IJCAI*, pages 614–622, 2024.

[Chen *et al.*, 2021] Zipeng Chen, Qianli Ma, and Zhenxi Lin. Time-aware multi-scale rnns for time series modeling. In *IJCAI*, pages 2285–2291, 2021.

[Chen *et al.*, 2023] Ling Chen, Donghui Chen, Zongjiang Shang, Binqing Wu, Cen Zheng, Bo Wen, and Wei Zhang. Multi-scale adaptive graph neural network for multivariate time series forecasting. *IEEE TKDE*, 35(10):10748–10761, 2023.

[Chen *et al.*, 2024] Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *ICLR*, 2024.

[Das *et al.*, 2023] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv*, 2023.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *NeurlPS*, 29, 2016.

[Deng *et al.*, 2022] Jinliang Deng, Xiusi Chen, Renhe Jiang, Xuan Song, and Ivor W Tsang. A multi-view multi-task learning framework for multi-variate time series forecasting. *IEEE TKDE*, 35(8):7665–7680, 2022.

[Deng *et al.*, 2024a] Jinliang Deng, Xiusi Chen, Renhe Jiang, Du Yin, Yi Yang, Xuan Song, and Ivor W Tsang. Disentangling structured components: Towards adaptive, interpretable and scalable time series forecasting. *IEEE TKDE*, 2024.

[Deng *et al.*, 2024b] Jinliang Deng, Feiyang Ye, Du Yin, Xuan Song, Ivor Tsang, and Hui Xiong. Parsimony or capability? decomposition delivers both in long-term time series forecasting. *NeurlPS*, 37:66687–66712, 2024.

[Jiang *et al.*, 2023] Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. In *AAAI*, volume 37, pages 4365–4373, 2023.

[Karita *et al.*, 2019] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. A comparative study on transformer vs rnn in speech applications. In *ASRU*, pages 449–456. IEEE, 2019.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv*, 2016.

[Kitaev *et al.*, 2020] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[Lai *et al.*, 2023] Zhichen Lai, Dalin Zhang, Huan Li, Christian S Jensen, Hua Lu, and Yan Zhao. Lightcts: A lightweight framework for correlated time series forecasting. *SIGMOD*, 1(2):1–26, 2023.

[Li *et al.*, 2017] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv*, 2017.

[Li *et al.*, 2023] Yiduo Li, Shiyi Qi, Zhe Li, Zhongwen Rao, Lujia Pan, and Zenglin Xu. Smartformer: Semi-autoregressive transformer with efficient integrated window attention for long time series forecasting. In *IJCAI*, pages 2169–2177, 2023.

[Liu *et al.*, 2022a] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *NeurlPS*, 35:5816–5828, 2022.

[Liu *et al.*, 2022b] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *ICLR*, 2022.

[Liu *et al.*, 2022c] Xu Liu, Yuxuan Liang, Chao Huang, Yu Zheng, Bryan Hooi, and Roger Zimmermann. When do contrastive learning signals help spatio-temporal graph forecasting? In *SIGSPATIAL*, pages 1–12, 2022.

[Liu *et al.*, 2022d] Yijing Liu, Qinxian Liu, Jian-Wei Zhang, Haozhe Feng, Zhongwei Wang, Zihan Zhou, and Wei Chen. Multivariate time-series forecasting with temporal polynomial graph neural networks. *NeurIPS*, 35:19414–19426, 2022.

[Liu *et al.*, 2022e] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *NeurIPS*, 35:9881–9893, 2022.

[Liu *et al.*, 2023] Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. *NeurIPS*, 36:12271–12290, 2023.

[Liu *et al.*, 2024] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *ICLR*, 2024.

[Nie *et al.*, 2023] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *ICLR*, 2023.

[Pan *et al.*, 2021] Qingyi Pan, Wenbo Hu, and Ning Chen. Two birds with one stone: Series saliency for accurate and interpretable multivariate time series forecasting. In *IJCAI*, pages 2884–2891, 2021.

[Qin *et al.*, 2017] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *IJCAI*, page 2627–2633, 2017.

[Rangapuram *et al.*, 2018] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *NeurIPS*, 31, 2018.

[Tang and Matteson, 2021] Binh Tang and David S Matteson. Probabilistic transformer for time series analysis. *NeurIPS*, 34:23592–23608, 2021.

[Vaswani, 2017] A Vaswani. Attention is all you need. *NeurIPS*, 2017.

[Wen *et al.*, 2023] Haomin Wen, Youfang Lin, Yutong Xia, Huaiyu Wan, Qingsong Wen, Roger Zimmermann, and Yuxuan Liang. Diffstg: Probabilistic spatio-temporal graph forecasting with denoising diffusion models. In *SIGSPATIAL*, pages 1–12, 2023.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, pages 1907–1913, 7 2019.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *SIGKDD*, pages 753–763, 2020.

[Wu *et al.*, 2021] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *NeurIPS*, 34:22419–22430, 2021.

[Wu *et al.*, 2023] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *ICLR*, 2023.

[Yu *et al.*, 2017] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv*, 2017.

[Zeng *et al.*, 2023] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI*, volume 37, pages 11121–11128, 2023.

[Zhang and Yan, 2023] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *ICLR*, 2023.

[Zhao *et al.*, 2017] Zheng Zhao, Weihai Chen, Xingming Wu, Peter CY Chen, and Jingmeng Liu. Lstm network: a deep learning approach for short-term traffic forecast. *IET intelligent transport systems*, 11(2):68–75, 2017.

[Zhao *et al.*, 2024] Xingyu Zhao, Zhongyu Wang, Zhihao Zhang, Fangbo Lu, Yanwei Yu, and Junyu Dong. Adaptive spatio-temporal graph recurrent network for sea surface temperature forecasting. *IEEE TGRS*, 2024.

[Zhou *et al.*, 2021] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, volume 35, pages 11106–11115, 2021.

[Zhou *et al.*, 2022] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, pages 27268–27286, 2022.

[Zhou *et al.*, 2024a] Xin Zhou, Weiqing Wang, Wray Buntine, Shilin Qu, Abishek Sriramulu, Weicong Tan, and Christoph Bergmeir. Scalable transformer for high dimensional multivariate time series forecasting. In *CIKM*, pages 3515–3526, 2024.

[Zhou *et al.*, 2024b] Ziyu Zhou, Gengyu Lyu, Yiming Huang, Zihao Wang, Ziyu Jia, and Zhen Yang. Sdformer: transformer with spectral filter and dynamic attention for multivariate time series long-term forecasting. In *IJCAI*, pages 3–9, 2024.

[Zhu *et al.*, 2023] Zhaoyang Zhu, Weiqi Chen, Rui Xia, Tian Zhou, Peisong Niu, Bingqing Peng, Wenwei Wang, Hengbo Liu, Ziqing Ma, Xinyue Gu, et al. Energy forecasting with robust, flexible, and explainable machine learning algorithms. *AI Magazine*, 44(4):377–393, 2023.