A Dynamic Stiefel Graph Neural Network for Efficient Spatio-Temporal Time Series Forecasting

Jiankai Zheng, **Liang Xie***
School of Mathematics and Statistics, Wuhan University of Technology, Wuhan 430070, China
312796@whut.edu.cn, whutxl@hotmail.com

Abstract

Spatio-temporal time series (STTS) have been widely used in many applications. However, accurately forecasting STTS is challenging due to complex dynamic correlations in both time and space dimensions. Existing graph neural networks struggle to balance effectiveness and efficiency in modeling dynamic spatio-temporal relations. To address this problem, we propose the Dynamic Spatio-Temporal Stiefel Graph Neural Network (DST-SGNN) to efficiently process STTS. For DST-SGNN, we first introduce the novel Stiefel Graph Spectral Convolution (SGSC) and Stiefel Graph Fourier Transform (SGFT). The SGFT matrix in SGSC is constrained to lie on the Stiefel manifold, and SGSC can be regarded as a filtered graph spectral convolution. We also propose the Linear Dynamic Graph Optimization on Stiefel Manifold (LDGOSM), which can efficiently learn the SGFT matrix from the dynamic graph and significantly reduce the computational complexity. Finally, we propose a multi-layer SGSC (MSGSC) that efficiently captures complex spatio-temporal correlations. Extensive experiments on seven spatio-temporal datasets show that DST-SGNN outperforms state-of-the-art methods while maintaining relatively low computational costs.

1 Introduction

In recent years, spatio-temporal time series (STTS) have become increasingly important in fields such as traffic flow forecasting, financial market analysis, and weather forecasting [Saad *et al.*, 2024; Li *et al.*, 2024b]. These data encompass changes and relationships in both time and space dimensions. The spatial relationships, in a broader sense, not only represent geographical locations but also denote the relationships between other types of entities, such as companies or sensors. Accurately forecasting STTS is crucial for decision support and resource optimization in these areas.

In contrast to traditional time series forecasting, spatio-temporal forecasting faces the challenge of analyzing complex and dynamic spatio-temporal correlations among STTS data. The key problem is how to effectively model both the temporal dependencies and the spatial interactions between variables. Existing representative methods for multivariate time series forecasting, such as TimeMixer [Wang *et al.*, 2024a] and CycleNet [Lin *et al.*, 2024], have demonstrated excellent performance in capturing periodicity and trend changes in time series data. These methods utilize techniques such as decomposition to model complex temporal patterns effectively. However, they primarily focus on temporal dimensions and often overlook the spatial correlations in STTS, limiting their optimal performance in spatio-temporal forecasting. Graph Neural Networks (GNNs) [Kipf and Welling, 2016], on the other hand, have gained popularity in spatio-temporal analysis due to their ability to model spatial relationships through graph structures. Recent advances in this area include the development of extended message-passing mechanisms for simultaneous spatial and temporal feature extraction [Kong *et al.*, 2024b], integrating spatio-temporal graph convolution networks for traffic prediction [Gupta *et al.*, 2023], and geometric masking techniques for interpreting GNNs in spatio-temporal contexts [Yu *et al.*, 2024]. These approaches fully utilize the advantages of GNNs in handling spatial interactions between nodes, thereby providing more accurate and comprehensive results for spatio-temporal forecasting.

However, although GNNs have great potential in spatio-temporal forecasting, their application faces two major challenges. Firstly, existing graph-based methods are computationally expensive, limiting their performance on large-scale datasets. Secondly, many graph methods struggle to adapt to the dynamic changes in spatio-temporal data, which further increases the computational costs [Bastos *et al.*, 2024]. Although some graph Fourier methods, such as FourierGNN [Yi *et al.*, 2024a], have alleviated these issues to some extent, they still fall short in analyzing dynamic spatio-temporal data.

To address these challenges, this paper proposes the Dynamic Spatio-temporal Stiefel Graph Neural Network (DST-SGNN), which introduces the Stiefel Graph Spectral Convolution (SGSC) to effectively handle the dynamic changes and high-dimensional characteristics of spatio-temporal data. DST-SGNN first reduces the dimensionality and complexity of spatio-temporal data through patch-based sequence decomposition, breaking it into smaller, manageable patches to

*Corresponding Author: Liang Xie

reduce redundancy and simplify the structure. Building on this, the model introduces Multi-layer Stiefel Graph Spectral Convolution (MSGSC) to effectively capture complex spatio-temporal correlations while maintaining high efficiency. To efficiently exploit dynamic graphs, Linear Dynamic Graph Optimization on Stiefel Manifold (LDGOSM) is proposed to efficiently learn the transformation matrix for SGSC.

The main advantages of DST-SGNN are as follows:

- DST-SGNN can dynamically model the evolving spatial correlations over time, thereby providing more accurate forecasting results.
- DST-SGNN achieves significant optimization in terms of time and space complexity compared to traditional graph neural networks, making it more efficient for processing large-scale spatio-temporal data.
- Extensive experiments on seven spatio-temporal datasets demonstrate DST-SGNN's superiority in accuracy and efficiency in comparison with state-of-the-art spatio-temporal and time series forecasting methods.

2 Related Work

2.1 Spatio-Temporal Time Series Forecasting

Multivariate time series forecasting is a key application in time series analysis, aiming to predict future values by modeling multiple correlated variables. Patch-based methods (such as PatchTST [Nie *et al.*, 2022] and MTST [Zhang *et al.*, 2024]) have made significant progress by dividing time series into local segments and using the Transformer architecture to capture both local and global temporal dependencies. Meanwhile, methods based on time series decomposition, such as TimesNet [Wu *et al.*, 2022] and TimeMixer [Wang *et al.*, 2024a], have further improved the performance of long-sequence forecasting by optimizing the decomposition process, and they better capture the intrinsic temporal structure of time series.

In the field of STTS, researchers have developed several innovative methods. STPGNN [Kong *et al.*, 2024a] identifies pivotal nodes and captures their complex spatio-temporal dependencies to enhance traffic prediction. MiTSformer [Chen and Zhao, 2024] addresses mixed time series by recovering latent continuous variables and leveraging multi-scale context for balanced modeling. ModWaveMLP [Sun *et al.*, 2024] offers an efficient and simple solution for traffic forecasting using MLPs combined with mode decomposition and wavelet denoising. Lastly, GPT-ST [Li *et al.*, 2024b] introduces a spatio-temporal pre-training framework that enhances downstream models' learning capabilities through masked autoencoders and adaptive masking strategies. These methods, each with unique strengths, provide diverse solutions for spatio-temporal time series forecasting.

2.2 Graph Neural Networks for Spatio-Temporal Analysis

GNNs are widely applied in spatio-temporal prediction. Their powerful modeling capabilities can effectively capture the complex relationships within the data. In the context of spatio-temporal prediction, the graph structure needs to be dynamically updated according to the data to adapt to spatio-temporal changes [Li *et al.*, 2024a; Ben *et al.*, 2024]. Meanwhile, as demonstrated by LLGformer [Jin *et al.*, 2025] and MPFGSN [Zheng *et al.*, 2025], full spatio-temporal graphs can significantly improve the prediction performance and help uncover potential patterns. However, the dynamic update and the construction of full spatio-temporal graphs will incur significant computational overhead. Especially when dealing with large-scale dynamic data, this limits the widespread application of GNNs.

Graph spectral convolution methods perform convolution in the spectral domain based on the graph Fourier transform, which can effectively capture the complex relationships in STTS [Cao *et al.*, 2020]. Nevertheless, traditional methods require eigenvalue decomposition of the adjacency matrix, resulting in high computational costs. Methods such as GCN [Kipf and Welling, 2016; Peng and Zhang, 2023] and ChebNet [Defferrard *et al.*, 2016; Lei *et al.*, 2024] replace eigenvalue decomposition with a spatial-like convolution that aggregates information via the adjacency matrix. However, the construction and dynamic update of the adjacency matrix remain computationally expensive. The FourierGNN [Yi *et al.*, 2024a] uses the standard Fourier transform to improve computational efficiency, but it is only applicable to fixed graph structures and is not suitable for dynamic spatio-temporal relationships.

3 Problem Definition

The problem of STTS forecasting involves forecasting future values of multiple variables based on their past observations across both time and space dimensions. STTS is similar to multivariate time series, and the main difference between them is that there exist spatial correlations among variables of STTS. Specifically, given a time series dataset with N variables observed over T consecutive time points, the goal is to predict the values of these variables at future time points $t+1$ to $t+\tau$, where τ is the prediction length. The prediction function $\mathbf{F}_\theta$ takes historical data X_t as input and outputs the predicted values Y_t for the next τ time points, capturing complex patterns and dependencies in the STTS data.

Formally, the prediction problem can be defined as finding a function $\mathbf{F}_\theta$ such that $Y_t = \mathbf{F}_\theta(X_t)$, where Y_t represents the predicted multivariate values for future time points $t+1$ to $t+\tau$.

4 Methodology

This section details the DST-SGNN model methodology. Section 4.1 presents the overall framework, while Sections 4.2–4.5 cover module implementation, with Section 4.4 focusing on the core theory of the Stiefel graph neural network. Section 4.6 describes training and inference procedures.

4.1 Overall Framework

Figure 1 illustrates the overall framework of the DST-SGNN model, which mainly includes four modules: Patching, Decomposition, LDGOSM, and MSGSC. First, the Patching module decomposes the original high-dimensional spatio-temporal data into multiple smaller sub-sequences (patches)

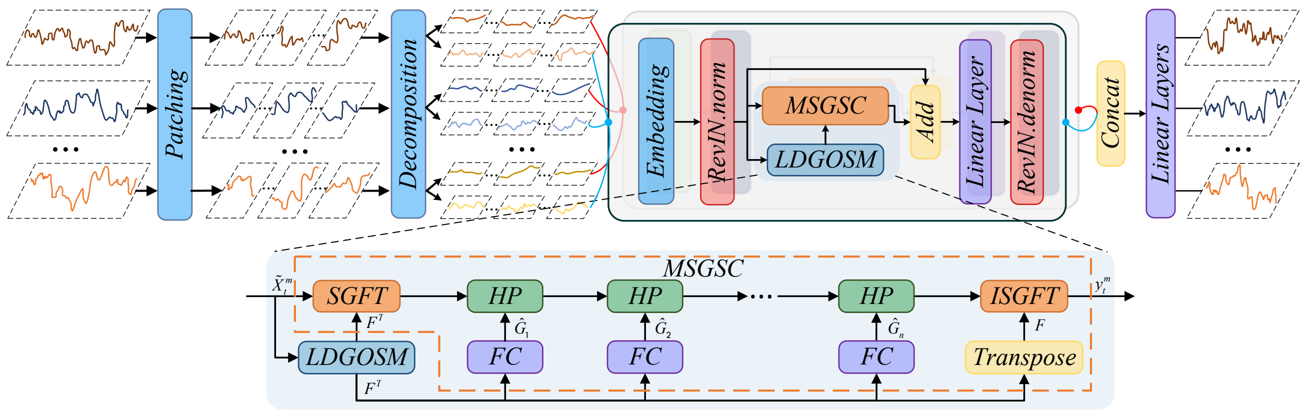

Figure 1: Framework diagram of the DST-SGNN model.

to reduce data complexity. Next, the Decomposition module further decomposes each patch to extract seasonal and trend components, which are used as nodes to construct a hyperpatch graph. After constructing the hyperpatch graph, an initial transformation matrix is formed from the graph's data. Subsequently, the LDGOSM module dynamically optimizes the transformation matrix on the Stiefel manifold. Finally, the MSGSC module performs multi-step graph spectral convolutions, where HP denotes the element-wise Hadamard product operation in the spectral convolution process. The seasonal and trend components are concatenated through the Concat module, and the resulting features are fed into the Linear Layers module to generate the final predictions. In the following, we will introduce each module in detail.

4.2 Patching and Decomposition

Patching

Given a multivariate time series window, the input is $X_t \in \mathbb{R}^{T\times N}$ of N variables at timestamp t. Let p denote the patch size, and s denote the stride (the length of the non-overlapping region between two consecutive patches). Then, X_t is fed into the $Patching$ module $\mathcal{T}$: $\mathbb{R}^{T\times N} \to \mathbb{R}^{J\times p\times N}$. This module transforms X_t into $J = \lfloor (T-p)/s \rfloor + 2$ patches. The specific calculation for the $Patching$ module $\hat{X}_t = \mathcal{T}(X_t)$ is as follows:

$$\hat{X}_{t,j} = X_{t[s(j-1)+1:s(j-1)+p]}, (j = 1, 2, ..., J) \qquad (1)$$

where $\hat{X}_t \in \mathbb{R}^{J\times p\times N}$ represents the output of the $Patching$ module, and $\hat{X}_{t,j}$ is its j-th column. To ensure that J is an integer, we pad the last s values of the sequence to the end of the original sequence.

Intra-patch Sequence Decomposition for Constructing the Hyperpatch Graph

First, the time series $\hat{X}_t$ is decomposed into seasonal component $\hat{X}_t^{(1)}$ and trend component $\hat{X}_t^{(2)}$ using the series decomposition block of Autoformer [Wu *et al.*, 2021].

$$\hat{X}_t^{(1)}, \hat{X}_t^{(2)} = \text{SeriesDecomp}(\hat{X}_t) \qquad (2)$$

In the hyperpatch graph structure, each patch in the seasonal component $\hat{X}_t^{(1)}$ and trend component $\hat{X}_t^{(2)}$ is defined as a node in the graph. For the sliding window of the time series, each element within the window is considered to have a connection with every other element, thus it is represented in the form of a spatio-temporal fully connected graph. Based on $\hat{X}_t^m \in \mathbb{R}^{J\times N\times p}, m = (1), (2)$, a hyperpatch graph structure $G_t^m = (X_t^{G,m}, A_t^{G,m})$ is further constructed. This structure is initialized as a fully connected graph containing $(J\times N)\times p$ nodes, where $X_t^{G,m} \in \mathbb{R}^{(J\times N)\times p}$ represents the node features, and $A_t^{G,m} \in \mathbb{R}^{((J\times N)\times p)\times((J\times N)\times p)}$ is the adjacency matrix, used to describe the relationships between nodes.

4.3 Embedding and Normalization

In the DST-SGNN model, the Embedding module maps the data $X_t^{G,m} \in \mathbb{R}^{(J\times N)\times p}$ after Decomposition into the hidden space $\tilde{X}_t^m \in \mathbb{R}^{(J\times N)\times K}$, where K is the dimension of the hidden layer. This process reduces the data dimensionality and enhances the feature representation capability. The Normalization module normalizes each time series instance to have zero mean and unit variance, thereby reducing the distribution differences between training and testing data [Kim *et al.*, 2021]. This design improves the model's generalization ability and adaptability to different data distributions.

4.4 Stiefel Graph Spectral Convolution

Basic Concepts

Definition 1. *The Stiefel Graph Spectral Convolution of a signal $x \in \mathbb{R}^{n\times 1}$ with a kernel $g \in \mathbb{R}^{n\times 1}$ is defined as:*

$$x *_s g = F(F^T x \odot F^T g) \qquad (3)$$

where F satisfies the following optimization problem:

$$\begin{aligned} &Min \quad Tr(F^T LF) \\ &s.t. \quad F \in St(n, d) \end{aligned} \qquad (4)$$

In Definition 1, $\odot$ denotes the Hadamard product, $St(n, d) \triangleq \{F \in \mathbb{R}^{n\times d} : F^T F = I_d\}$ is the Stiefel manifold, $L = I_n - D^{-1/2}AD^{-1/2}$ is the Laplacian matrix, $A \in \mathbb{R}^{n\times n}$ is the graph adjacency matrix, and D is the diagonal matrix, and its diagonal element is the sum of all elements in each row of A, which represents the degree of the corresponding node in an undirected graph (i.e., the number of edges connected to the node).

Theorem 1. *The matrix F can be obtained by solving the eigenvalue decomposition of $D^{-1/2}AD^{-1/2}$ and selecting the eigenvectors corresponding to the d largest eigenvalues.*

The proof of Theorem 1 is provided in Appendix A.1. [1]

The $F^T x$ in Definition 1 can be considered as a pseudo-graph Fourier transform, which we refer to as the Stiefel Graph Fourier Transform (SGFT), defined as:

$$S(x) = F^T x \tag{5}$$

We can also define the Inverse Stiefel Graph Fourier Transform (ISGFT) as:

$$S^{-1}(x) = Fx \tag{6}$$

The SGSC of x and g can be obtained by first applying the SGFT to x and g separately, then performing element-wise multiplication, and finally applying the ISGFT. Since F is not orthogonal, it does not satisfy the definition of the standard graph Fourier transform. However, we can prove that SGSC is still a specific type of standard graph spectral convolution [Defferrard *et al.*, 2016].

Theorem 2. *The Stiefel Graph Spectral Convolution can be viewed as the following specific filtered graph spectral convolution:*

$$x *_s g = P g_\theta(\Lambda) P^T x \tag{7}$$

where $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_n)$, and $\lambda_1, \lambda_2, \cdots, \lambda_n$ are the eigenvalues of A sorted in descending order. $P = (p_1, p_2, \cdots, p_n)$, where $p_i \in R^{n\times 1}$ is the eigenvector corresponding to λ_i. $g_\theta(\Lambda) = diag(\theta)$, and $\theta = (\theta_1, \theta_2, \cdots, \theta_n)$ satisfies:

$$\theta_i = \begin{cases} p_i^T g & \text{if } \lambda_i \geq \lambda_d \\ 0 & \text{if } \lambda_i < \lambda_d \end{cases} \tag{8}$$

The proof of Theorem 2 is provided in Appendix A.2.

According to Theorem 2, it can be seen that the Stiefel Graph Spectral Convolution satisfies all the properties of graph spectral convolution. Moreover, setting the coefficients θ_i corresponding to smaller eigenvalues to zero reduces the time and space complexity of the graph spectral convolution. And it also serves as a filtering and noise-reduction mechanism.

The SGSC we define can be easily extended to the case of multi-dimensional features. For $X \in R^{n\times k}$, if we choose the convolution kernel $G \in R^{n\times k}$, then the SGSC is given by:

$$X *_s G = F(F^T X \odot F^T G) \tag{9}$$

Algorithm 1 Implementation of LDGOSM

```
Input: $X \in R^{n\times d}$, $E \in R^{n\times d}$
Output: $W \in R^{d\times d}$
1: Perform eigenvalue decomposition on $B = X^T X$ to obtain eigenvector matrix $D$ and eigenvalue diagonal matrix $\Lambda$;
2: Let $M = D\Lambda^{-1/2}D^T$;
3: Compute $C = E^T X$;
4: Compute $H = B + C^T C$;
5: Perform eigenvalue decomposition on $M^T HM$ to obtain eigenvector matrix $U$;
6: return $W = MU$;
```

Linear Dynamic Graph Optimization on Stiefel Manifold

In the SGSC, the introduction of the Stiefel manifold simplifies the convolution computation, but the eigenvalue decomposition of L still results in a time complexity of $O(N^3)$, where N represents the total number of nodes across all variables and all patches in our paper. Moreover, for spatio-temporal data, the spatial relationships between variables change over time, requiring graph adjacent matrix A to be recalculated frequently. Each computation of the graph requires solving for eigenvalues, leading to a more complex computation. To simplify the dynamic computation of A, inspired by [Yang *et al.*, 2024], we introduce the following calculation of the dynamic graph adjacency matrix:

$$A = I_N + EE^T, \quad E = ReLU(X) \tag{10}$$

Furthermore, to reduce the complexity of the SGSC, we improve the computation of F in Theorem 1. We propose an efficient Linear Dynamic Graph Optimization on Stiefel Manifold (LDGOSM). Inspired by [Huang *et al.*, 2018], we introduce a linear transformation: $F = XW$, where $W \in \mathbb{R}^{d\times d}$. By substituting equation (10) into the equivalent form of (4) (as proven in Theorem 1 in Appendix A.1), the optimization problem can be transformed into the following objective function:

$$\begin{aligned} \max \quad & \mathrm{Tr}(W^T(X^T X + X^T EE^T X)W) \\ \text{s.t.} \quad & W^T X^T XW = I \end{aligned} \tag{11}$$

The optimal solution W for the objective function (11) can be obtained through Algorithm 1 (see Appendix A.3 for detailed derivation).

In Algorithm 1, the complexity matrix B and the eigenvalue decomposition are $O(nd^2)$ and $O(d^3)$, respectively. Computing matrices M, C, and H are $O(d^3)$, $O(nd^2)$, and $O(d^2)$. Computing $M^T HM$ and the subsequent eigenvalue decomposition are both $O(d^3)$. Finally, computing matrix W is $O(d^3)$. The total complexity is $O(nd^2 + kd^3)$, where k is a small constant. If $d \ll n$, the complexity is dominated by $O(nd^2)$, making it approximately linear with nodes number n.

[1]The appendix can be accessed at: https://github.com/komorebi424/DST-SGNN.

Datasets	PEMS03	PEMS07	Electricity	CSI300	exchange_rate	Solar	METR-LA
Features	358	883	321	100	8	593	207
Timesteps	26207	28223	26304	2791	7588	8760	34272

Table 1: Dataset statistics.

Multi-layer SGSC

To enhance the effectiveness of dynamic spatio-temporal graph learning, we employ the following Multi-layer SGSC (MSGSC):

$$MSGSC(X, G) = \sum_{i=1}^{m} X *_s G_1 *_s G_2 *_s \cdots *_s G_i \tag{12}$$

where G_i represents the convolutional kernel of each layer in the MSGSC.

Theorem 3. *The MSGSC has the following equivalent computational form:*

$$MSGSC(X, G) = S^{-1}(\sum_{i=1}^{m} S(X) \odot \prod_{j=1}^{i} S(G_j)) \tag{13}$$

The proof of Theorem 3 is provided in Appendix A.4.

Through Theorem 3, when computing the MSGSC, multiple SGFT and ISGFT operations are avoided, thereby improving computational efficiency while retaining the effects of the original multi-step convolution.

In our implementation, $S(G_j) = F^T G_j$ can be learned via a linear fully connected layer (FC), i.e., $\hat{G}_j = FC(F^T)$. Therefore, the computation of MSGSC can be expressed as:

$$y_t^m = MSGSC(\tilde{X}_t^m, F^T) = S^{-1}(\sum_{i=1}^{m} S(\tilde{X}_t^m) \odot \prod_{j=1}^{i} \hat{G}_j) \tag{14}$$

where, F is optimized using the Algorithm 1.

4.5 Concat Module and Linear Layers Module

In DST-SGNN, the seasonal and trend components $y_t^{(1)}$ and $y_t^{(2)}$ output from the MSGSC module are processed through inverse RevIN, resulting in $\hat{Y}_t^{(1)}$ and $\hat{Y}_t^{(2)}$. These features are then concatenated via the Concat module to form $\hat{Y}_t$, which is subsequently fed into the Linear Layers module to generate the final prediction Y_t.

4.6 The Inference and Training of DST-SGNN

In DST-SGNN, spatio-temporal data is decomposed into patches and split into seasonal and trend components. The LDGOSM module optimizes the transformation matrix F on the Stiefel manifold. The MSGSC module captures complex patterns via graph spectral convolutions. Predictions are generated by concatenating features and passing them through linear layers. The training process involves only standard eigenvalue decomposition, which can be efficiently handled using PyTorch's linalg module, and uses MAE loss for optimization.

5 Experiments

5.1 Datasets and Settings

Datasets

To evaluate the performance of DST-SGNN[2], this study conducted experiments on seven spatio-temporal benchmark datasets: PEMS03, PEMS07, Electricity, CSI300, exchange_rate, Solar, and METR-LA. Table 1 provides a summary of the dataset statistics, with detailed information on the seven public benchmark datasets as follows:

- PEMS03 and PEMS07: These two datasets consist of California highway traffic flow data, collected in real-time every 30 seconds by the California Performance Measurement System (PEMS) [Song *et al.*, 2020]. The original traffic flow data is aggregated at 5-minute intervals. The datasets record the geographical information of the sensor stations. PEMS03 and PEMS07 are datasets from two specific regions.
- Electricity: Electricity contains hourly time series of the electricity consumption of 321 customers from 2012 to 2014 [Asuncion *et al.*, 2007].
- CSI300: This is a stock dataset, compiling the closing prices of one hundred stocks that have been continuously listed in the CSI 300 from 2012 to 2024 [Zheng *et al.*, 2025].
- exchange_rate: This dataset records the daily exchange rates of eight different countries from 1990 to 2016 [Lai *et al.*, 2018].
- Solar[3]: The dataset selects power plant data points in Florida as the dataset, which includes 593 points. The data collection period spans from January 6, 2001, to December 31, 2006, with samples taken every hour, totaling 8,760 in length.
- METR-LA: The dataset includes traffic information collected by loop detectors on highways in Los Angeles County from March 1, 2012, to June 30, 2012. It contains data from 207 sensors, with samples taken every 5 minutes. [Yi *et al.*, 2024a].

We divided all the datasets chronologically into a training set (70%), a validation set (10%), and a test set (20%).

Experimental Settings

The DST-SGNN model is implemented in Python using PyTorch 2.2.0 and is trained on a GPU (NVIDIA GeForce RTX 4090). MSE and MAE are used as evaluation metrics. We set the input window size T to 96, 192, or 336, and the learning rate to 0.0001. All of the models follow the same experimental setup with forecasting window sizes

[2]Our code at https://github.com/komorebi424/DST-SGNN.

[3]https://www.nrel.gov/grid/solar-power-data.html.

Model		DST-SGNN		TimeMixer		TimeXer		MiTSformer		iTransformer		FourierGNN		FreTS		Fredformer		FITS	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
PEMS03	96	0.1936	**0.2635**	0.3607	0.4203	**0.1820**	0.2962	1.3222	0.9412	1.7197	1.0753	0.2978	0.4049	0.2459	0.3503	1.1039	0.8235	0.4239	0.5127
	192	**0.2122**	**0.2760**	0.4477	0.4660	0.2431	0.3452	1.7115	1.0879	1.9911	1.2053	0.3284	0.4251	0.2808	0.3751	1.1645	0.8395	0.3541	0.4484
	336	**0.2318**	**0.2910**	0.3431	0.3970	0.2889	0.3758	1.3013	0.8972	1.3191	0.9035	0.3134	0.4104	0.2618	0.3541	0.8229	0.6543	0.3106	0.4087
	720	0.3040	**0.3506**	0.4247	0.4594	0.3002	0.3800	1.5214	0.9890	1.6422	1.0467	0.3485	0.4354	**0.2997**	0.3842	0.8356	0.6663	0.4988	0.5591
PEMS07	60	0.4696	0.4417	0.2905	0.3802	**0.2101**	**0.3277**	0.8547	0.7161	1.6041	1.0468	0.2644	0.3668	0.2403	0.3357	0.8227	0.6662	0.3207	0.4345
	96	0.4645	0.4444	0.3826	0.4261	0.4625	0.5196	1.2500	0.8942	1.8446	1.1342	0.3178	0.4109	**0.2860**	**0.3699**	1.5280	0.9913	0.4041	0.5043
	192	**0.3436**	**0.3762**	0.4582	0.4739	0.5380	0.5585	1.4574	0.9694	1.7269	1.0716	0.3822	0.4379	0.3479	0.4093	1.5146	0.9855	0.3689	0.4674
	336	0.3346	**0.3675**	0.3624	0.4091	0.4565	0.4973	1.1288	0.8114	1.4432	0.9504	0.3462	0.4148	**0.3129**	0.3844	0.8725	0.6821	0.3327	0.4399
Electricity	96	0.1552	0.2448	0.1586	0.2494	**0.1441**	0.2438	0.1558	0.2483	0.1460	**0.2381**	0.2931	0.3838	0.1792	0.2666	0.2344	0.3201	0.5628	0.6111
	192	0.1671	0.2555	0.1706	0.2606	**0.1618**	0.2602	0.1741	0.2634	0.1636	**0.2553**	0.3030	0.3927	0.1837	0.2731	0.2382	0.3259	0.5974	0.6299
	336	0.1806	**0.2707**	0.1882	0.2788	0.1829	0.2820	0.2021	0.2893	**0.1803**	0.2720	0.3173	0.4064	0.1996	0.2905	0.2556	0.3418	0.6450	0.6547
	720	0.2133	0.2999	0.2296	0.3159	0.2162	0.3129	0.2332	0.3138	**0.2082**	**0.2975**	0.3456	0.4256	0.2353	0.3244	0.4009	0.4527	0.6437	0.6490
CSI300	36	**0.1435**	**0.2182**	0.2121	0.2625	0.1845	0.2580	0.2574	0.2970	0.2285	0.2862	0.4066	0.3398	0.2148	0.2757	0.3369	0.3655	0.2511	0.3733
	48	**0.1876**	**0.2487**	0.3067	0.3098	0.2269	0.2792	0.3401	0.3411	0.2803	0.3159	0.6206	0.4078	0.3078	0.3197	0.3722	0.3863	0.5647	0.6130
	60	**0.2241**	**0.2717**	0.3406	0.3350	0.2648	0.3024	0.5111	0.4111	0.3762	0.3729	0.9337	0.4849	0.6574	0.4090	0.4013	0.4030	0.2586	0.3798
	96	**0.2987**	**0.3259**	0.5120	0.4107	0.5740	0.4651	0.6084	0.4585	0.4247	0.4017	1.2942	0.5696	1.1953	0.5384	0.4838	0.4507	0.6159	0.6428
exchange_rate	48	**0.0439**	0.1464	0.0572	0.1631	0.0919	0.2123	0.0474	0.1486	0.0917	0.2213	0.0726	0.1957	0.0532	0.1634	0.0794	**0.0794**	0.1456	0.2887
	96	**0.0791**	**0.2020**	0.1184	0.2394	0.1155	0.2513	0.1148	0.2291	0.1042	0.2338	0.1516	0.2905	0.1218	0.2515	0.1241	0.2519	0.2278	0.3653
	192	**0.1508**	0.2860	0.2405	0.3454	0.2019	**0.2019**	0.1857	0.2957	0.2175	0.3448	0.2953	0.4134	0.1915	0.3243	0.2215	0.3399	0.4276	0.5084
	336	**0.2611**	**0.3812**	0.4107	0.4612	0.3794	0.4486	0.3921	0.4390	0.3879	0.4562	0.4328	0.4985	0.4050	0.4825	0.3688	0.4437	0.6705	0.6398
Solar	96	0.1802	**0.2007**	**0.1606**	0.2216	0.1677	0.2472	0.1807	0.2532	0.1631	0.2191	0.3143	0.4134	0.1955	0.2482	0.3434	0.4379	0.7187	0.7212
	192	0.1822	**0.2003**	**0.1593**	0.2289	0.1937	0.2657	0.1875	0.2632	0.1663	0.2226	0.3103	0.4207	0.2020	0.2478	0.3343	0.4409	0.7260	0.7284
	336	0.1849	**0.1989**	**0.1624**	0.2295	0.1745	0.2596	0.1874	0.2536	0.1706	0.2272	0.3114	0.4217	0.2038	0.2538	0.3907	0.4743	0.6669	0.6959
	720	0.2021	**0.2102**	**0.1709**	0.2361	0.1812	0.2722	0.1989	0.2647	0.1830	0.2323	0.4217	0.4295	0.2323	0.2674	0.3496	0.4652	0.7663	0.7561
METR-LA	96	1.2756	**0.6243**	1.1316	0.6483	1.4146	0.7436	1.3261	0.7250	1.5068	0.7609	1.1737	0.7085	**1.0759**	0.6834	1.2441	0.6516	1.2678	0.7873
	192	1.3912	**0.6619**	1.3192	0.7097	1.5473	0.7941	1.4698	0.7821	1.6180	0.7923	1.2967	0.7678	**1.2428**	0.7310	1.5024	0.7636	1.3339	0.8080
	336	1.4780	**0.6841**	1.3539	0.7247	1.5242	0.7706	1.5705	0.7883	1.6485	0.7838	1.3287	0.7842	**1.2913**	0.7513	1.5445	0.7613	1.4403	0.8430
	720	1.7026	**0.7302**	1.5596	0.8309	1.7827	0.8716	1.7972	0.8438	1.9614	0.8813	1.4944	0.8244	**1.4582**	0.7957	1.8800	0.8500	1.6439	0.9186
avg rank		**2.7857**	**1.5000**	3.8929	3.8214	3.7857	4.2143	5.9286	5.6786	5.5714	5.3929	5.8571	6.1071	3.8929	4.2143	6.7500	6.3214	6.5357	7.7500

Table 2: Summary of results for all methods on nine datasets.The best results are in bold and the second best are underlined.

$H \in \{96, 192, 336, 720\}$ for the PEMS03, Electricity, Sola and METR-LA datasets, $H \in \{48, 96, 192, 336\}$ for the exchange_rate datasets, $H \in \{36, 48, 60, 96\}$ for the CSI300 dataset, and $H \in \{60, 96, 192, 336\}$ for the PEMS07 dataset.

5.2 Methods for Comparison

Our experiments comprehensively compared the forecasting performance of our model with several representative state-of-the-art (SOTA) models on seven datasets, including the advanced long-term forecasting models TimeMixer [Wang *et al.*, 2024a] and TimeXer [Wang *et al.*, 2024b], the Transformer-based model iTransformer [Liu *et al.*, 2023], the spatio-temporal forecasting model MiTSformer [Chen and Zhao, 2024], and the Fourier transform based models FourierGNN [Yi *et al.*, 2024a], FreTS [Yi *et al.*, 2024b], Fredformer [Piao *et al.*, 2024], and FITS [Xu *et al.*, 2023]. We used the code released in the original papers, and all models followed the same experimental settings as described in the original papers.

5.3 Main Results

Table 2 presents the experimental results of DST-SGNN compared with other methods on the seven spatio-temporal datasets, leading to the following explanations:

- The DST-SGNN model shows excellent performance on various datasets, with an average rank of 2.7857 for MSE and 1.5000 for MAE. In particular, on the CSI300 and exchange_rate datasets, the DST-SGNN significantly outperforms other comparative methods across all forecasting window lengths. This result demonstrates that DST-SGNN has a significant advantage in handling financial data and enables more accurate forecasting of stock prices and exchange rates, which is crucial in financial decision-making.
- FreTS based on Fourier transform performs well on the traffic datasets PEMS03, PEMS07, and METR-LA, but underperforms on other datasets. In contrast, iTransformer, which is based on the Transformer architecture, shows better performance on non-traffic datasets and falls short on traffic datasets. The reason is that traffic data exhibit clear periodicity and trends, which can be easily captured by Fourier transform-based methods, leading to better results on traffic datasets. Moreover, transformer-based models excel at modeling long-term dependencies and complex nonlinear patterns on non-periodic data. DST-SGNN is superior on all datasets by incorporating the strengths of MSGSC, which effectively extracts complex relationships in both temporal and spatial dimensions. Whether dealing with periodic traffic data or non-periodic financial and power data, DST-SGNN maintains high forecasting accuracy and efficiency.
- TimeXer also performs well on all datasets, but as the forecasting length increases, the advantages of DST-SGNN become increasingly evident. This is attributed to the optimization of SGSC on the Stiefel manifold, which enables the model to more effectively handle the dynamic changes in spatio-temporal data. Additionally, DST-SGNN uses patch-based sequence decomposition to break down the original data into multiple smaller subsequences. This process further reduces the high di-

Methods		PEMS03				Electricity				CSI300				METR-LA			
		96	192	336	720	96	192	336	720	36	48	60	96	96	192	336	720
DST-SGNN	MSE	**0.1936**	**0.2122**	**0.2318**	0.3040	**0.1552**	**0.1671**	**0.1806**	**0.2133**	**0.1435**	**0.1876**	**0.2241**	**0.2987**	**1.2756**	**1.3912**	**1.4780**	**1.7026**
	MAE	**0.2635**	**0.2760**	**0.2910**	**0.3506**	**0.2448**	**0.2555**	**0.2707**	**0.2999**	**0.2182**	**0.2487**	**0.2717**	**0.3259**	**0.6243**	**0.6619**	**0.6841**	**0.7302**
Rep-StdGSC	MSE	0.2177	0.2475	0.2513	0.3304	0.2018	0.2026	0.2168	0.2494	0.1508	0.2466	0.2479	0.3850	1.3613	1.4401	1.5140	1.7244
	MAE	0.2973	0.3301	0.3200	0.3803	0.2760	0.2818	0.3011	0.3286	0.2254	0.2841	0.2846	0.3744	0.6420	0.6888	0.6996	0.7428
Rep-SpatConv	MSE	0.2180	0.2255	0.2469	0.3270	0.2021	0.2067	0.2172	0.2535	0.1687	0.2338	0.2507	0.3249	1.2997	1.5104	1.5474	1.7176
	MAE	0.3043	0.3000	0.3218	0.3798	0.2775	0.2890	0.3024	0.3342	0.2356	0.2748	0.2863	0.3421	0.6377	0.6843	0.6974	0.7347
w/o-Stiefel	MSE	0.2199	0.2297	0.2443	**0.3037**	0.2015	0.2029	0.2145	0.2484	0.1599	0.2142	0.2411	0.3753	1.2766	1.4854	1.5107	1.6990
	MAE	0.2966	0.2983	0.3128	0.3686	0.2731	0.2833	0.2970	0.3270	0.2318	0.2628	0.2859	0.3529	0.6563	0.6924	0.6976	0.7405

Table 3: Comparative performance of DST-SGNN and variants. The best results are in bold and the second best are underlined.

mensionality and complexity of the data. As a result, our model can maintain high efficiency and accuracy even with long forecasting windows. These characteristics make the advantages of DST-SGNN more prominent in long forecasting sequences and provide a more efficient and accurate solution for STTS forecasting.

5.4 Ablation Studies

To further investigate the specific role of the core module SGSC in DST-SGNN, we conducted ablation studies with the following three variants:

- Rep-StdGSC: Variant replacing SGSC with standard graph spectral convolution, using an orthogonal matrix $F \in \mathbb{R}^{n \times n}$ for Graph Fourier Transform to perform spectral convolutions.
- Rep-SpatConv: Variant replacing SGSC with spatial graph convolution, where the convolution operation is defined as $\sum_i A_i X_i W$, aggregating neighboring node features directly in the spatial domain.
- w/o-Stiefel: Variant removing the Stiefel manifold constraint and replacing the original LDGOSM with a multilayer perceptron (MLP).

The experimental results shown in Table 3 indicate that the three variants—Rep-StdGSC, Rep-SpatConv, and w/o-Stiefel—perform worse than DST-SGNN, although they still demonstrate the effectiveness of GNNs in spatio-temporal analysis. DST-SGNN's superior performance can be attributed to its usage of the Stiefel manifold, which provides filtering capabilities by discarding low-information eigenvalues. This mechanism not only reduces noise but also enhances the model's ability to capture complex spatio-temporal patterns efficiently. Compared to Rep-StdGSC and Rep-SpatConv, DST-SGNN's other advantage lies in its computational efficiency, which will be further analyzed in the subsequent sections regarding time complexity.

5.5 Time Consumption Analysis

Table 4 presents the experimental time consumption and parameter count for DST-SGNN, Rep-StdGSC, Rep-SpatConv, MiTSformer, Fredformer, and iTransformer on the PEMS07 dataset. The results show that DST-SGNN achieves lower time and space complexity than most methods due to its optimized SGSC. Although DST-SGNN has a longer training time than iTransformer due to the eigenvalue decomposition in the Stiefel manifold optimization, its inference time and parameter count are lower. This is attributed to the reduced complexity of eigenvalue decomposition to $O(d^3)$ with $d \ll n$, which enables faster inference despite the additional training overhead. In addition, DST-SGNN has the fewest parameters. This highlights DST-SGNN's advantage in spatial complexity, making it more memory-efficient while also reducing inference overhead. For more experimental results, please refer to Appendix C.

Methods		PEMS07			
		60	96	192	336
DST-SGNN	Train	236.84	240.14	271.23	304.4
	Inference	61.7	**65.98**	**78.78**	**101.34**
	Parameters	**4176K**	**4213K**	**4311K**	**4459K**
Rep-StdGSC	Train	696.04	701.09	720.5	742.07
	Inference	174.4	177.33	209.57	248.13
	Parameters	225163K	225200K	225298K	225446K
Rep-SpatConv	Train	698.55	682.25	798.8	699.72
	Inference	179.04	173.05	183.96	185.44
	Parameters	76546K	76583K	76681K	76829K
MiTSformer	Train	2933.64	2668.45	2907.72	2952.88
	Inference	854.95	862.67	858.43	1379.61
	Parameters	6821K	6848K	6922K	7033K
Fredformer	Train	966.38	950.04	1088.01	1281.55
	Inference	74.01	84	108.58	173.62
	Parameters	74936K	119803K	239537K	419379K
iTransformer	Train	**98.5**	**98.73**	**97.55**	**106.78**
	Inference	**56.31**	67.3	111.35	182.18
	Parameters	6393K	6411K	6461K	6534K

Table 4: Experimental time consumption (seconds per epoch) and parameter analysis on PEMS07 dataset. The best results are in bold and the second best are underlined.

6 Conclusions

The DST-SGNN model proposed in this paper effectively handles the dynamic changes and high-dimensional characteristics of spatio-temporal data by combining the advantages of graph neural networks and SGSC. DST-SGNN reduces data dimensionality and complexity through patch-based sequence decomposition and optimizes its core module, the SGSC, on the Stiefel manifold, reducing computational complexity while maintaining sensitivity to dynamic spatio-temporal relationships. Experiments show that DST-SGNN outperforms existing methods in forecasting accuracy and computational efficiency on seven public benchmark datasets, particularly in handling financial time series data and long forecasting windows. Our experiments have covered a variety of spatio-temporal data. In the future, we will consider more spatio-temporal application scenarios and further utilize and improve our model to apply it to these scenarios.

Acknowledgments

This research was supported by the PCL-CMCC Foundation for Science and Innovation (Grant No. 2024ZY2B0050).

References

[Asuncion *et al.*, 2007] Arthur Asuncion, David Newman, et al. Uci machine learning repository, 2007.

[Bastos *et al.*, 2024] Anson Bastos, Kuldeep Singh, Abhishek Nadgeri, Manish Singh, and Toyotaro Suzumura. Beyond spatio-temporal representations: Evolving fourier transform for temporal graphs. *arXiv preprint arXiv:2402.16078*, 2024.

[Ben *et al.*, 2024] Jie Ben, Qiguo Sun, Keyu Liu, Xibei Yang, and Fengjun Zhang. Multi-head multi-order graph attention networks. *Applied Intelligence*, pages 1–16, 2024.

[Cao *et al.*, 2020] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems*, 33:17766–17778, 2020.

[Chen and Zhao, 2024] Jiawei Chen and Chunhui Zhao. Addressing spatial-temporal heterogeneity: General mixed time series analysis via latent continuity recovery and alignment. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

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

[Gupta *et al.*, 2023] Arti Gupta, Manish Kumar Maurya, Nikhil Goyal, and Vijay Kumar Chaurasiya. Istgcn: Integrated spatio-temporal modeling for traffic prediction using traffic graph convolution network. *Applied Intelligence*, 53(23):29153–29168, 2023.

[Huang *et al.*, 2018] Lei Huang, Xianglong Liu, Bo Lang, Adams Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Jin *et al.*, 2025] Di Jin, Cuiying Huo, Jiayi Shi, Dongxiao He, Jianguo Wei, and Philip S Yu. Llgformer: Learnable long-range graph transformer for traffic flow prediction. In *Proceedings of the ACM on Web Conference 2025*, pages 2860–2871, 2025.

[Kim *et al.*, 2021] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Kong *et al.*, 2024a] Weiyang Kong, Ziyu Guo, and Yubao Liu. Spatio-temporal pivotal graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8627–8635, 2024.

[Kong *et al.*, 2024b] Ziqian Kong, Xiaohang Jin, Feng Wang, and Zhengguo Xu. Spatio-temporal propagation: An extended message passing graph neural network for remaining useful life prediction. *IEEE Sensors Journal*, 2024.

[Lai *et al.*, 2018] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[Lei *et al.*, 2024] Tianyang Lei, Kewei Yang, Jichao Li, Gang Chen, and Jiuyao Jiang. Multichannel spatial–temporal graph convolution network based on spectrum decomposition for traffic prediction. *Expert Systems with Applications*, 238:122281, 2024.

[Li *et al.*, 2024a] Hao Li, Jie Liu, Shiyuan Han, Jin Zhou, Tong Zhang, and CL Philip Chen. Stfgcn: Spatial–temporal fusion graph convolutional network for traffic prediction. *Expert Systems with Applications*, 255:124648, 2024.

[Li *et al.*, 2024b] Zhonghang Li, Lianghao Xia, Yong Xu, and Chao Huang. Gpt-st: generative pre-training of spatio-temporal graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[Lin *et al.*, 2024] Shengsheng Lin, Weiwei Lin, Xinyi Hu, Wentai Wu, Ruichao Mo, and Haocheng Zhong. Cyclenet: enhancing time series forecasting through modeling periodic patterns. *arXiv preprint arXiv:2409.18479*, 2024.

[Liu *et al.*, 2023] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

[Nie *et al.*, 2022] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[Peng and Zhang, 2023] Dunlu Peng and Yongsheng Zhang. Ma-gcn: A memory augmented graph convolutional network for traffic prediction. *Engineering Applications of Artificial Intelligence*, 121:106046, 2023.

[Piao *et al.*, 2024] Xihao Piao, Zheng Chen, Taichi Murayama, Yasuko Matsubara, and Yasushi Sakurai. Fredformer: Frequency debiased transformer for time series forecasting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2400–2410, 2024.

[Saad *et al.*, 2024] Feras Saad, Jacob Burnim, Colin Carroll, Brian Patton, Urs Köster, Rif A. Saurous, and Matthew Hoffman. Scalable spatiotemporal prediction with bayesian neural fields. *Nature Communications*, 15(1):7942, 2024.

[Song *et al.*, 2020] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 914–921, 2020.

[Sun *et al.*, 2024] Ke Sun, Pei Liu, Pengfei Li, and Zhifang Liao. Modwavemlp: Mlp-based mode decomposition and wavelet denoising model to defeat complex structures in traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9035–9043, 2024.

[Wang *et al.*, 2024a] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*, 2024.

[Wang *et al.*, 2024b] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. *arXiv preprint arXiv:2402.19072*, 2024.

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

[Wu *et al.*, 2022] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

[Xu *et al.*, 2023] Zhijian Xu, Ailing Zeng, and Qiang Xu. Fits: Modeling time series with $10k$ parameters. *arXiv preprint arXiv:2307.03756*, 2023.

[Yang *et al.*, 2024] Linghua Yang, Wantong Chen, Xiaoxi He, Shuyue Wei, Yi Xu, Zimu Zhou, and Yongxin Tong. Fedgtp: Exploiting inter-client spatial dependency in federated graph-based traffic prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6105–6116, 2024.

[Yi *et al.*, 2024a] Kun Yi, Qi Zhang, Wei Fan, Hui He, Liang Hu, Pengyang Wang, Ning An, Longbing Cao, and Zhendong Niu. Fouriergnn: Rethinking multivariate time series forecasting from a pure graph perspective. *Advances in Neural Information Processing Systems*, 36, 2024.

[Yi *et al.*, 2024b] Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Ning An, Defu Lian, Longbing Cao, and Zhendong Niu. Frequency-domain mlps are more effective learners in time series forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

[Yu *et al.*, 2024] Rui Yu, Yanshan Li, Huajie Liang, and Zhiyuan Chen. Geoexplainer: Interpreting graph convolutional networks with geometric masking. *Neurocomputing*, 605:128393, 2024.

[Zhang *et al.*, 2024] Yitian Zhang, Liheng Ma, Soumyasundar Pal, Yingxue Zhang, and Mark Coates. Multi-resolution time-series transformer for long-term forecasting. In *International Conference on Artificial Intelligence and Statistics*, pages 4222–4230. PMLR, 2024.

[Zheng *et al.*, 2025] Jiankai Zheng, Liang Xie, and Haijiao Xu. Multi-resolution patch-based fourier graph spectral network for spatiotemporal time series forecasting. *Neurocomputing*, 638:130132, 2025.