UFGTime: Mining Intertwined Dependencies in Multivariate Time Series via an Efficient Pure Graph Approach

Ruikun Li*
The University of Sydney
Sydney, Australia
ruikun.li@sydney.edu.au

Dai Shi*
The University of Sydney
Sydney, Australia
dai.shi@sydney.edu.au

Ye Xiao*
The University of Sydney
Sydney, Australia
ye.xiao@sydney.edu.au

Junbin Gao†
The University of Sydney
Sydney, Australia
junbin.gao@sydney.edu.au

ABSTRACT

Graph Neural Networks (GNNs) have become a cornerstone in multivariate time series forecasting by addressing the challenge of modeling inter-series dependencies often overlooked by traditional temporal approaches. However, real-world temporal dependencies (inter- and intra-dependencies) are inherently intertwined, making it difficult to treat them as separate processes. Recent pure graph paradigms attempt to capture these dependencies holistically by transforming time series into fully connected graphs. While effective, these methods suffer from prohibitive computational complexity $O((NT)^2)$, limiting their scalability for large-scale data and long-term forecasting. To address these challenges, we propose UFGTime, a novel framework that leverages spectral signals to construct a "spectral-variate graph," embedding multivariate temporal dependencies in a compact spectral representation and modeling inter- and intra-signal connections through frequency similarities. Empowered by our proposed graph framelet message-passing function, UFGTime efficiently aggregates global information, avoids over-smoothing, and achieves near-linear complexity $O(kNT)$. Extensive experiments on diverse datasets demonstrate that UFGTime consistently outperforms state-of-the-art baselines, offering a scalable, accurate, and resource-efficient pure graph solution for multivariate time series forecasting.

PVLDB Reference Format:
Ruikun Li, Dai Shi, Ye Xiao, and Junbin Gao. UFGTime: Mining Intertwined Dependencies in Multivariate Time Series via an Efficient Pure Graph Approach. PVLDB, 18(9): 3175 - 3188, 2025.
doi:10.14778/3746405.3746436

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/WonderHeiYi/UFGTIME.

*These authors contributed equally to this research.
†Corresponding author

Proceedings of the VLDB Endowment, Vol. 18, No. 9 ISSN 2150-8097.
doi:10.14778/3746405.3746436

1 INTRODUCTION

Multivariate time series forecasting plays a crucial role in industrial applications such as transportation, manufacturing, and energy management [26]. Recent advances in deep neural networks have significantly improved forecasting accuracy [27, 35, 48]. However, these methods primarily focus on exploring temporal patterns within individual time series, often overlooking the inter-relations between time series. These inter-relations are critical for capturing complex dependencies in multivariate time series data and have the potential to enhance model performance [6, 17].

To harness these inter-relations, a graph structure is often employed to represent these connections, serving as the foundation for applying Graph Neural Networks (GNNs). Renowned for their effectiveness in graph representation learning, GNNs have been utilized to model these inter-correlations, enabling each time series to leverage information from others [6, 17]. Forecasting models with GNN modules enable the simultaneous modeling of inter-series relationships (via GNNs) and intra-series dependencies (via temporal model), providing a unified framework for leveraging both types of information in time series data, resulting as many developed state-of-the-art such as DCRNN [21], GraphWaveNet [41], STGCN [46], and MTGNN [40]. Despite their success, these models often rely on integrating GNN modules with separate temporal components to capture both types of dependencies, treating the modeling of inter- and intra-series connections as two independent processes. This separation is inconsistent with the entangled nature of the underlying dynamics via multivariate time series data.

To resolve the problem, recent research, FourierGNN [45], moves beyond traditional GNN approaches to these models, which build end-to-end frameworks to capture inter- and intra-temporal relations with separate modules. Instead, it explicitly explores the intertwined interactions within the multivariate temporal system. To globally model the entangled temporal dependencies, it proposes a fully connected framework that disregards the distinction between inter- and intra-temporal connections, treating all connections as the same type. This approach assumes that all time points in a multivariate time series system share uniform connections, thereby transforming the multivariate time series into a novel graph structure characterized by a fully connected graph. This serves as an initial prototype that applies the pure graph method to multivariate time series analysis.

Nonetheless, this "nascent" pure graph paradigm leaves several challenges unaddressed. Treating the multivariate temporal system as a complete graph results in computational complexity of $O((NT)^2)$, which is computationally expensive. While FourierGNN partially mitigates this issue by reducing the complexity to $O(NT \log(NT))$ through learning a neural frequency scaling filter to approximate a fully connected graph convolution, this approach still requires further optimization to improve efficiency. Moreover, representing global connections with a complete graph causes all time slices to share the same fully connected structure, failing to reflect dynamic features in multivariate time series. This static representation overlooks the varying relationships between time points across different periods, limiting its ability to capture temporal dynamics effectively. Considering the limitations of transforming time series into complete graphs, the question of ***"how to find a compact and non-fully connected graph solution that effectively represents the global and dynamic dependencies of multivariate time series"*** emerges as ***Challenge 1*** in this work.

In time series analysis, spectral representations are often used as an alternative [19, 38, 43] to temporal representations to capture information in a *global* and *overarching* form (e.g., periodicity), offering the detection of the trends that are not easily discernible in temporal representations. Inspired by this, we propose leveraging the spectral domain to construct a pure spectral graph as a compact alternative for capturing the intertwined patterns in multivariate temporal systems (against ***Challenge 1***). Given the global nature of spectral signals, applying GNNs on a partially connected graph in the spectral domain to mix various frequencies can effectively represent overarching temporal information. Based on signal similarities, a customized graph with appropriate densities can be constructed, providing an efficient foundation for diverse downstream applications. However, the flexible connection levels of this graph impose higher requirements on the GNNs applied. When the graph is sparse, GNNs focus excessively on local connectivity, neglecting critical global neighboring information on the graph. Conversely, GNN propagates on dense graphs, exacerbating the over-smoothing (OSM) issue [30] and aforementioned high computational complexity. This leads to a critical question: ***"how can we design efficient GNNs to capture global neighboring information (e.g., by spectral filtering [11]) with limited connections and avoid over-smoothing even when the dense graph is needed?"*** This forms ***Challenge 2***, which will be extensively discussed in Section 3.5.

Based on the above analysis, this paper develops a novel framework, UFGTime, for time series forecasting by leveraging an efficient framelet GNN (Defined in Section 3) on the spectral domain graph. Specifically, we introduce multi-scale spectral filtering within our framework, enabling the aggregating of global information that would otherwise be neglected when the input graph is sparse. Furthermore, we demonstrate that our GNN is immune to over-smoothing under mild conditions, regardless of graph density (against ***Challenge 2***). To the best of our knowledge, this is the first work that represents the spectral frequencies of a multivariate time series as a graph and utilizes connections between different frequencies to capture the complex dynamics of multivariate temporal systems. The contributions of this work are summarized as follows:

- We propose the spectral-variate graph, which transforms spectral signals of multivariate time series into graph features connected by signal frequency similarities. This graph globally embeds temporal dependencies into a sparse graph, enabling more efficient graph operations.
- We propose a novel framelet GNN that effectively utilizes global information on sparse graphs and demonstrates that it prevents the over-smoothing issue under mild conditions. Furthermore, our analysis shows that the method remains compact and efficient, with a complexity of $O(kNT)$ when applied to the proposed spectral-variate graph.
- Through evaluations of twelve short-term forecasting datasets, two long-term datasets, and supplementary tests, we demonstrate that our model consistently outperforms state-of-the-art baselines with an efficient design.

2 PRELIMINARIES AND RELATED WORKS

In this section, we introduce the fundamental notations and define the forecasting problem. Additionally, we discuss two paradigms for applying GNNs to multivariate time series forecasting, which provide valuable insights and guidelines for designing our framework.

2.1 Problem Definition

A multivariate time series $\boldsymbol{X} \in \mathbb{R}^{N \times T \times D}$ represents a sequence of D-dimensional vector observations of N entities recorded over a time period T. We denote $\boldsymbol{X}_t = [X_{t-T+1}, \ldots, X_{t-1}, X_t] \in \mathbb{R}^{N \times T \times D}$ representing the observations on the looking back-window of size T at timestamp t, where $X_t \in \mathbb{R}^{N \times D}$ is the observation for all the N entities at t. A typical forecasting task is to learn a model $f(\cdot)$, by minimizing a predefined loss function, such that

$$\widehat{Y}_{t+1} = f(\boldsymbol{X}_t) = f\left([X_{t-T+1}, \ldots, X_t]\right) \tag{1}$$

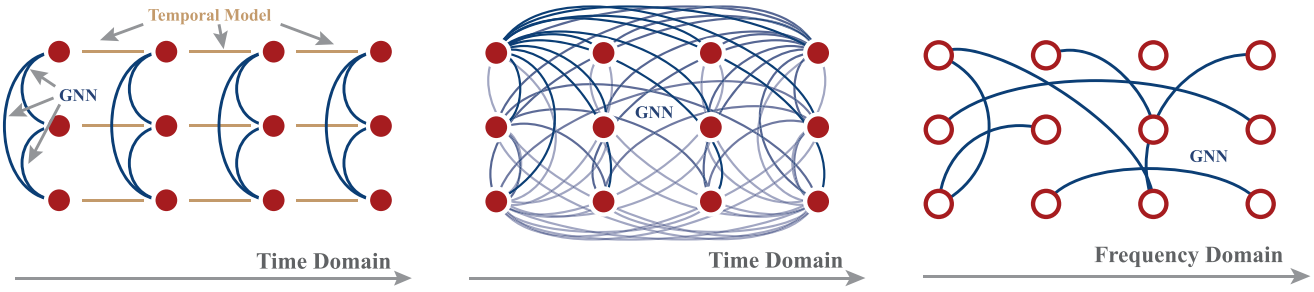
predicts the next τ steps of time series $\boldsymbol{Y}$ at $t+1$, e.g., $\boldsymbol{Y}_{t+1} = [X_{t+1}, \ldots, X_{t+\tau}] \in \mathbb{R}^{N \times \tau \times D}$, forecasting within the prediction time window τ.

2.2 Paradigm of GNNs in Multivariate Time Series Forecasting

Multivariate time series forecasting has been widely studied in various fields, such as economics, finance, and traffic, using deep learning algorithms such as convolutional neural networks (CNNs) [2, 4], recurrent neural networks (RNNs) [9, 16], and transformers [49, 52]. These deep learning approaches often employ sophisticated structures to model the temporal patterns within individual time series. However, a major limitation of these methods is their inability to account for inter-dependencies across time series, which can significantly hinder forecasting accuracy. To address these challenges in modeling temporal and structural dependencies, GNNs have been leveraged across diverse paradigms (shown in Table 1), enabling the integration of temporal dynamics and inter-series relationships. These paradigms can be categorized as follows:

Table 1: Comparison of Different GNNs in Multivariate Time Series Forecasting

Paradigm	Core Design	Methods	Graph Type	Graph Domain	Graph Operation	Entangled Pattern	Complexity
Modulity	GNN + Recurrence	STGNN [37], AGCRN [3], etc.	Sparse	Temporal	Yes	No	-
	GNN + Convolution	GWNet [41], MTGNN [40], etc.	Sparse	Temporal	Yes	No	-
	GNN + Attention	GMAN [50], TPGNN [25], etc.	Sparse	Temproal	Yes	No	-
Pure Graph	Hypervariate Graph + Frequency Scaling	FourierGNN [45]	Complete	Temproal	No (Equivalent)	Yes	$O(NT\log(NT)D + NTD^2)$
	Spectralvariate Graph + Graph Framelet	**Ours**	**Sparse**	**Spectral**	**Yes**	**Yes**	$O(NTkD + NTD^2)$



(a). Modularity Paradigm (b). Pure Graph Paradigm (FourierGNN) (c). Pure Graph Paradigm (Ours)

Figure 1: Visual Demonstration of GNN Methods of Different Paradigms

Modularity Paradigm. The introduction of DCRNN [21], which integrates graph and recurrent modules into an end-to-end framework, represented a significant step forward in capturing inter-series correlations and temporal dynamics (intra-series dependencies). This design has emerged as a dominant paradigm for applying GNNs in multivariate time series forecasting (as demonstrated in Figure 1 (a)). Variants of this paradigm can be categorized into three groups: *GNNs with recurrence* (e.g., ST-MetaNet [28], STGNN [37], AGCRN [3], GTS [31], and HiGP [8]), *GNNs with convolution* (e.g., GraphWaveNet [41], MTGNN [40], StemGNN [5], STGODE [13], MTGODE [18], and CaST [42]), and *GNNs with temporal attention* (e.g., GMAN [50], STAR [47], and TPGNN [25]). Despite their effectiveness, these frameworks often separate cross-series modeling and temporal dynamic learning into two distinct processes. This separation may lead to a disjointed representation of the complex dependencies observed in nature, inadequately capturing the inherent spatio-temporal interconnections, thereby potentially limiting forecasting performance.

Pure Graph Paradigm. To address the limitations of the *modularity paradigm* in capturing the complex entanglement of inter- and intra-temporal information, FourierGNN [45] introduces the pure graph paradigm, treating multivariate time series as a fully connected structure known as a ***hypervariate graph***:

Definition 1 (**Hypervariate Graph** [45]). *Given a multivariate time window $\boldsymbol{X}_t \in \mathbb{R}^{N\times T\times D}$ at timestamp t, a hypervariate graph is defined as $\boldsymbol{G}_t^H = \left(\boldsymbol{X}_t^G, \boldsymbol{J}\right)$, where $\boldsymbol{X}_t^G \in \mathbb{R}^{NT\times D}$ represents node features, and $\boldsymbol{J} \in \mathbf{1}^{NT\times NT}$ denotes the fully connected adjacency matrix.*

Unlike *modularity paradigms* that separately model inter-series and intra-series dependencies, the hypervariate graph treats multivariate time series as a single, fully connected graph, embedding both temporal dynamics and inter-series correlations, as shown in Figure 1 (b). This representation reformulates the forecasting problem as:

$$\widehat{Y}_{t+1} = g_\theta(\boldsymbol{G}_t^H) = g_\theta(\boldsymbol{X}_t^G, \boldsymbol{J}), \tag{2}$$

where $g_\theta(\cdot)$ is a GNN that accepts node features and adjacency matrix as inputs. This unified paradigm addresses key limitations of modularity-based methods and offers significant potential for time series analysis. However, the fully connected nature of the hypervariate graph introduces substantial computational challenges. The time and space complexity of graph operations grow quadratically with the number of time points and series, making direct implementation infeasible for large-scale data. To address this, they employ a neural frequency scaling filter in the Fourier domain, which has been proven to be equivalent to graph aggregation on a complete graph, thereby significantly reducing complexity while maintaining global connectivity.

Despite these innovations, the pure graph paradigm proposed by Yi et al. [45] has certain *limitations*. First, scaling each frequency individually overlooks interactions across signals, potentially oversimplifying their relationships. Second, the fully connected assumption of the hypervariate graph causes the connections to remain static across different periods, failing to capture the dynamic nature of temporal variations. Third, the equivalent fully connected graph convolution tend to lead *oversmoothing*, a phenomenon where enforced similarity between nodes results in indistinguishable representations, reducing the model's ability to preserve distinctive representation of nodes. Finally, the computational complexity,

$O(NT\log(NT)D + NTD^2)$, poses significant challenges for scaling to large-scale data. These limitations underscore the need for alternative compact strategies, such as leveraging sparse graph structures to reduce complexity or employing adaptive filters to better capture frequency interactions.

3 PROPOSED METHOD

In this section, we provide an overview of UFGTime (Section 3.1) and introduce two key components of our framework: the Formation of the Spectral-variate Graph (Section 3.2) and the Global Graph Framelet Message-Passing Operator (Section 3.4).

3.1 Overview of Framework

As shown in Figure 2, our framework, UFGTime, decomposes the multivariate temporal forecasting task into two stages: (1) *spectral-variate graph transformation*, and (2) *global graph framelet operation*.

In the first stage, we employ the Fourier transformation to project temporal information into the spectral domain, treating all frequency signals as node features of a graph. We then use the k-nearest neighbors (KNN) algorithm to capture the similarity between different Fourier signals and establish edge connections. By combining the signals (as node features) and edge connections calculated by KNN, we construct the *spectral-variate graph* (as Definition 2), which is a k-customized density graph representation of the multivariate temporal system.

In the second stage, we introduce the global graph framelet message-passing function to apply graph convolution on the spectral-variant graph. This function leverages fast graph-level framelet decomposition to enable efficient graph convolution while mitigating the over-smoothing problem.

3.2 Transformation from Multivariate Time Series to Sparse Graph

As discussed in Section 1 and 2.2, representing multivariate time series as a fully connected graph captures both inter- and intra-relations within the temporal system. However, this static connectivity fails to accurately reflect the dynamic dependencies of real-world systems and significantly increases computational complexity, making it unsuitable for large-scale data analysis. Thus, constructing a sparse graph that captures both global and dynamic dependencies becomes the first challenge of this work.

To preserve global information from multivariate time series without relying on a fully connected structure to model temporal patterns, we turn to spectral representations of time series. The Discrete Fourier Transform (DFT) is a computational tool widely used to convert data from the time domain to the spectral domain. Given a finite-length sequence $\boldsymbol{x}[t \in T]$, the DFT and Inverse Discrete Fourier Transform (IDFT) denoted as $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$, are defined as follows:

$$\boldsymbol{s}[c] = \mathcal{F}(\boldsymbol{x})[\boldsymbol{c}] \coloneqq \sum_{t=0}^{T-1} \boldsymbol{x}[t] e^{-i2\pi ct/T}, \tag{3}$$

$$\boldsymbol{x}[t] = \mathcal{F}^{-1}(\boldsymbol{s})[t] \coloneqq \frac{1}{T}\sum_{c=0}^{T-1} \boldsymbol{s}[c] e^{i2\pi ct/T}, \tag{4}$$

where $\boldsymbol{s}$ is the complex spectral representation of the time-domain sequence $\boldsymbol{x}$. Notably, the basis functions $e^{-i2\pi ct/T} = \cos(2\pi ct/T) - i\sin(2\pi ct/T)$ span the entire time domain, covering all discrete points. Each frequency component $\boldsymbol{s}[c]$ captures a global pattern of the sequence $\boldsymbol{x}$.

By adopting this insight, we first transform the real multivariate time series $\boldsymbol{X}_t$ into its spectral representation $\boldsymbol{S}_t$ that keep low-frequency spectra $C = \lfloor T/2 \rfloor + 1$, as follows:

$$\boldsymbol{S}_t = \mathcal{F}(\boldsymbol{X}_t) \in \mathbb{C}^{N\times C\times D}. \tag{5}$$

The spectral representation of a multivariate time series comprises multiple Fourier signals, each corresponding to distinct frequencies. These signals also exhibit dependencies, reflecting both intra- and inter-signal relationships in the time domain. Thus, the entire spectral representation can be viewed as a graph in the spectral domain, encapsulating the complex interactions within the temporal domain. We note that the transform $\mathcal{F}(\boldsymbol{X}_t)$ defined in (5) can be adopted to those datasets with different temporal dynamics (i.e., different T). Furthermore, we show our models' performances via long-term time series forecasting task with different time steps (T) in Table 4.

To construct the graph in the spectral domain, we first apply a vectorization operation to the spectral representation to form the graph node features:

$$\boldsymbol{S}_t^G = \mathrm{vec}_{(N,C)}(\boldsymbol{S}_t) \in \mathbb{C}^{NC\times D}, \tag{6}$$

where $\boldsymbol{S}_t^G$ represents the graph node features that formed by spectral signals.

Next, we define the graph edge connections. Since each signal in $\boldsymbol{S}_t^G$ retains global information from the temporal domain, we construct a sparse connection by measuring the similarity between each signal's frequency components. The cosine similarity between two arbitrary graph features (Fourier signals) is defined as:

$$\text{similarity}\left(\boldsymbol{s}_t^G(i), \boldsymbol{s}_t^G(j)\right) = \frac{\boldsymbol{s}_t^G(i)\cdot \boldsymbol{s}_t^G(j)}{\|\boldsymbol{s}_t^G(i)\|\|\boldsymbol{s}_t^G(j)\|}, \tag{7}$$

where $i, j \in NC$ are the graph node indices, and $\boldsymbol{s}_t^G(i)$ and $\boldsymbol{s}_t^G(j)$ are the features of nodes i and j in $\boldsymbol{S}_t^G$.

Using cosine similarity, we apply the K-nearest neighbors (KNN) algorithm to select the top-k similar signals:

$$\mathcal{N}_k(\boldsymbol{s}_t^G(i)) = \underset{\boldsymbol{s}_t^G(j),\, j\neq i}{\arg\operatorname{top} k}\left(\text{similarity}\left(\boldsymbol{s}_t^G(i), \boldsymbol{s}_t^G(j)\right)\right), \tag{8}$$

where $\mathcal{N}_k(\boldsymbol{s}_t^G(i))$ denotes the nearest neighbors of node i.

Finally, the adjacency matrix $\boldsymbol{A}_t$ is defined as:

$$\boldsymbol{A}_t(i,j) = \begin{cases} 1, & \text{if } \boldsymbol{s}_t^G(j) \in \mathcal{N}_k(\boldsymbol{s}_t^G(i)); \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Using this adjacency matrix, a graph is constructed based on the spectral signals $\boldsymbol{S}_t$. This new graph structure is formally defined as follows:

Definition 2 (**Spectral-variate Graph**). *Given a general multivariate time window $\boldsymbol{X}_t \in \mathbb{R}^{N\times T\times D}$ at time step t, the spectral temporal signals $\boldsymbol{S}_t$ are defined as the Fourier-transformed time series, $\boldsymbol{S}_t = \mathcal{F}(\boldsymbol{X}_t) \in \mathbb{C}^{N\times C\times D}$. The spectral-variate graph $\boldsymbol{G}_t^S$ at*

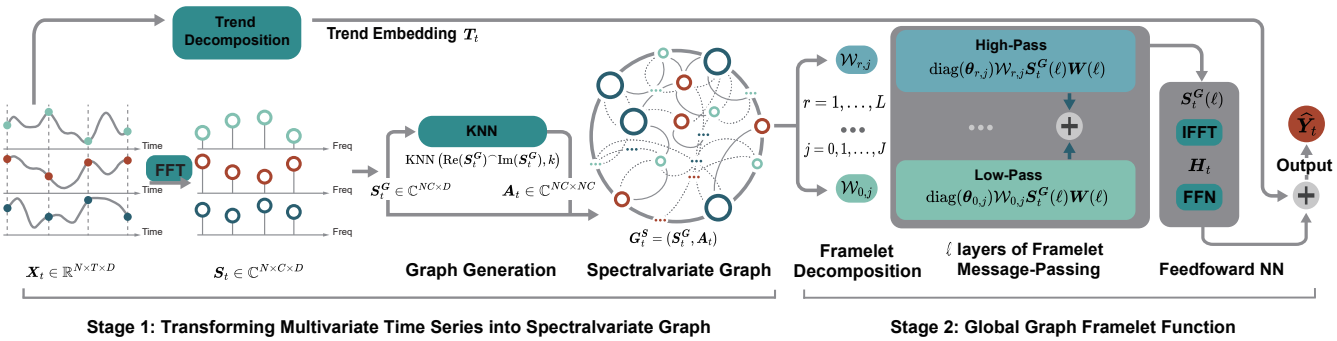


Figure 2: A workflow demonstration of UFGTime framework for predicting $\widehat{Y}_{t+1}$ with input X_t

timestamp t is then defined as $G_t^S = \left(S_t^G, A_t\right)$, where $S_t^G \in \mathbb{C}^{NC\times D}$ represents the graph features, and $A_t \in \{0,1\}^{NC\times NC}$ is the adjacency matrix associate with spectral-variate graph features S_t^G.

Our motivation for advocating spectral-variate graphs lies in the inherent global representation capability of the Fourier transformation. Instead of relying on fully connected graphs to capture global patterns in multivariate time series, we leverage the KNN algorithm to construct sparse connections between various Fourier signals, thereby modeling complex dependencies in the spectral domain. This approach implicitly retains the global temporal patterns while significantly reducing computational overhead. With higher sparsity, spectral-variate graphs offer great potential for improving graph operation efficiency.

3.3 Challenges and Requirements for GNNs on Spectral-variate Graphs

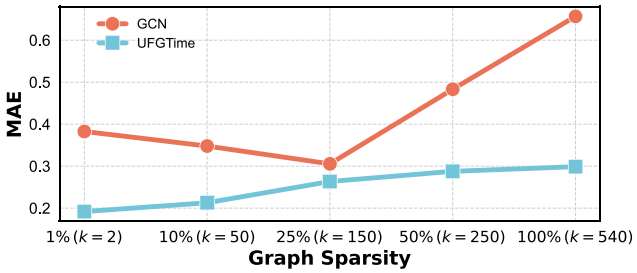


Figure 3: A case study of various graph sparsity impact on GCN and UFGTime performances on spectral-variate graph formed by Covid-Cal dataset. One can check that UFGTime shows better robustness to the sparsity change of the graph (which risks over-smoothing) compared to the classic GCN model.

Unlike the hypervariate graph introduced in Definition 1, where fully connected graphs are constructed across all timestamps, the adjacency matrices in the spectral-variate graph are estimated based on signal similarity using the KNN algorithm. Once the graph structure is defined, the next challenge is processing the features S^G with the generated graph structure A_t at a customized density k. When A_t is sparse, employing traditional GNNs to propagate spectral-variate graph features may result in short-sightedness, focusing only on local, nearest-neighbor connections and failing to capture global relationships. Conversely, when A_t is densely constructed (large k of KNN), feature propagation risks over-smoothing [30]. This issue causes node features to become indistinguishable after graph propagation, which is detrimental to downstream tasks such as forecasting.

Therefore, a suitable GNN model for spectral-variate graphs preferably addresses the following requirements:

- *Prevent excessive smoothing (similarity) of spectral-variate graph features by preserving the identifiability (sharpness) of each node's features during propagation.*
- *Propagate node features in a global manner, even though the graph is sparsely connected.*

How to select GNN for spectral-variate graphs? GNNs that satisfy the first requirement often employ a diffusion-reaction paradigm, where node features are first homogenized through spatial propagation (i.e., using A). Then the ego-graph feature (e.g., S_t^G) is added to reintroduce variation into the system [7, 14, 36]. While these models have achieved remarkable results, spectral GNNs, such as ChebNet [11], typically learn filtering functions (e.g., diagonal matrices) in the spectral domain (i.e., the eigenspace of the graph Laplacian), which enables feature propagation from a global perspective, thus satisfying the second requirement. Consequently, an ideal model would either be a spectral GNN that can induce multiple feature dynamics or a spatial GNN that accounts for global dependencies between features.

In light of these considerations, we focus on a family of spectral GNNs known as **Graph Framelets**, which meet the aforementioned requirements. In Figure 3, we conduct a case study to show how the changes in the graph sparsity (i.e., number of edges) affect the forecasting performance between GCN, which suffers from the over-smoothing problem particularly when the graph is dense, and our global graph framelet message-passing function (to be defined in the next section) which is immune to the over-smoothing problem e.g., caused by dense graph (See the main theorem 1 in Section 3.5 for details.). We finally remark that although GCN eventually owns the worst performance with the highest MAE when the graph is nearly fully connected, the initial MAE drop remains to show

that one customized graph sparsity (i.e., to describe the inter and intra-temporal relations) is of the necessity of producing higher model forecasting performances.

3.4 Framelet Message Passing on the Fourier domain

In this section, we formulate a novel global graph framelet message-passing system, specifically designed for spectral-variate graphs in multivariate time series forecasting tasks, addressing the unique demand posed by this setting. While numerous framelet variants have been developed in recent years [15, 24, 34], these approaches primarily focus on node features in the real domain. In contrast, our work refines the original graph framelet framework [44, 51] to the spectral domain of spectral-variate graphs. This new design enables us to more effectively capture the intricate relationships of the spectral-variate graph.

3.4.1 ***Graph Framelet Message-Passing.*** Graph framelets are defined by a set of filter banks, denoted as $\eta_{a,b} = \{a; b^{(1)}, \ldots, b^{(L)}\}$, and the corresponding complex-valued scaling functions. These scaling functions are typically expressed as $\Psi = \{\alpha; \beta^{(1)}, \ldots, \beta^{(L)}\}$, where L represents the number of high-pass filters. The framelet framework adheres to the following refinement relationship between the scaling functions and filter banks:

$$\widehat{\alpha}(2\xi) = \widehat{a}(\xi)\widehat{\alpha}(\xi), \tag{10}$$

$$\widehat{\beta^{(r)}}(2\xi) = \widehat{b^{(r)}}(\xi)\widehat{\alpha}(\xi), \ \forall\, \xi \in \mathbb{R},\ r = 1, \ldots, L, \tag{11}$$

where $\widehat{\alpha}$ and $\widehat{\beta^{(r)}}$ denote the Fourier transforms of α and $\beta^{(r)}$, respectively, and $\widehat{a}$, $\widehat{b^{(r)}}$ represent the Fourier series of a and $b^{(r)}$. The graph framelets are then defined as

$$\varphi_{j,p}(v) = \sum_{i=1}^{n} \widehat{\alpha}\left(\frac{\Lambda_i}{2^j}\right) u_i(p) u_i(v), \tag{12}$$

$$\psi^r_{j,p}(v) = \sum_{i=1}^{n} \widehat{\beta^{(r)}}\left(\frac{\Lambda_i}{2^j}\right) u_i(p) u_i(v), \tag{13}$$

for $r = 1, \ldots, L$ and scale level $j = 1, \ldots, J$. Here, $u_i(v)$ refers to the eigenvector u_i at node v. The functions $\varphi_{j,p}(\cdot)$ and $\psi^r_{j,p}(\cdot)$ are commonly referred to as the *low-pass framelets* and *high-pass framelets* at node p. We further denote $\mathbf{\Lambda}$ be the matrix that contains the eigenvalues of the graph Laplacian. One can define the framelet decomposition matrices $\mathcal{W}_{0,J}$ and $\mathcal{W}_{r,J}$ as:

$$\mathcal{W}_{0,J} = \boldsymbol{U}\widehat{a}\left(\frac{\mathbf{\Lambda}}{2^{m+J}}\right) \cdots \widehat{a}\left(\frac{\mathbf{\Lambda}}{2^m}\right) \boldsymbol{U}^\top, \tag{14}$$

$$\mathcal{W}_{r,0} = \boldsymbol{U}\widehat{b^{(r)}}\left(\frac{\mathbf{\Lambda}}{2^m}\right) \boldsymbol{U}^\top, \quad \text{for } r = 1, \ldots, L, \tag{15}$$

$$\mathcal{W}_{r,j} = \boldsymbol{U}\widehat{b^{(r)}}\left(\frac{\mathbf{\Lambda}}{2^{m+j}}\right) \widehat{a}\left(\frac{\mathbf{\Lambda}}{2^{m+j-1}}\right) \cdots \widehat{a}\left(\frac{\mathbf{\Lambda}}{2^m}\right) \boldsymbol{U}^\top, \tag{16}$$

for $r = 1, \ldots, L,\ j = 1, \ldots, J$.

Here, m represents the coarsest scale level, which is the smallest value satisfying $2^{-m}\lambda_{(N)} \leq \pi$. Let the set $\mathcal{I} = \{(r, j) : r = 1, \ldots, L,\ j = 0, 1, \ldots, J\} \cup \{(0, J)\}$, one can verify the so-called *tightness* of the framelet decomposition and reconstruction such that $\sum_{(r,j)\in\mathcal{I}} \mathcal{W}^\top_{r,j}\mathcal{W}_{r,j} = \boldsymbol{I}$.

We highlight that, in practice, to avoid the heavy eigen-decomposition of the graph Laplacian, one may adopt the K-order polynomial to boost implementation speed. For notation simplicity, we denote the polynomials as $\mathcal{T}_j(\xi)$ instead of $\mathcal{T}^K_j(\xi)$. Accordingly, the above framelet decomposition can be approximated as:

$$\mathcal{W}_{0,J} \approx \mathcal{T}_0\left(\frac{1}{2^{L+m}}\widetilde{\boldsymbol{L}}\right) \cdots \mathcal{T}_0\left(\frac{1}{2^m}\boldsymbol{L}\right), \tag{17}$$

$$\mathcal{W}_{r,0} \approx \mathcal{T}_r\left(\frac{1}{2^m}\boldsymbol{L}\right), \quad \text{for } r = 1, ..., L, \tag{18}$$

$$\mathcal{W}_{r,j} \approx \mathcal{T}_r\left(\frac{1}{2^{m+\ell}}\boldsymbol{L}\right)\mathcal{T}_0\left(\frac{1}{2^{m+\ell-1}}\widetilde{\boldsymbol{L}}\right) \cdots \mathcal{T}_0\left(\frac{1}{2^m}\boldsymbol{L}\right), \tag{19}$$

for $r = 1, ..., L,\ j = 1, ..., J$.

in which we let $\boldsymbol{L}$ be the graph Laplacian matrix and note that $k = 2$ is good enough for providing high-quality approximations in practice [44]. In summary, one can explicitly denote the feature propagation of the graph framelet (without activation) as

$$\boldsymbol{H}(\ell + 1) = \sum_{(r,j)\in\mathcal{I}} \mathcal{W}^\top_{r,j}\mathrm{diag}(\boldsymbol{\theta}_{r,j})\mathcal{W}_{r,j}\boldsymbol{H}(\ell)\boldsymbol{W}(\ell), \tag{20}$$

where we generically denote $\boldsymbol{H}(\ell)$ be the feature matrix at layer ℓ (e.g., $\boldsymbol{H}(\ell) = \boldsymbol{S}^G_t(\ell)$), and $\mathrm{diag}(\boldsymbol{\theta})$ contains learnable coefficients in each frequency domain, $\boldsymbol{W}(\ell)$ is the weight matrix that is shared across the different frequency domains. In addition, we further simplify the framelet model by omitting the reconstruction process [24] and applying identical channel-mixing (i.e., $\mathbf{W}$) among all frequency domains. This leads to

$$\boldsymbol{H}(\ell + 1) = \sum_{(r,j)\in\mathcal{I}} \mathrm{diag}(\boldsymbol{\theta}_{r,j})\mathcal{W}_{r,j}\boldsymbol{H}(\ell)\boldsymbol{W}(\ell), \tag{21}$$

and thus forming a multiscale message-passing model that propagates the node features through efficient spectral filtering.

3.5 Model Properties

As aforementioned (e.g., Challenge 2 in Introduction), it is preferable to have a GNN that **can avoid over-smoothing even when the graph is dense.** Below, we show that the propagation defined in (21) can avoid over-smoothing under the mild conditions. First, we state the main theorem as follows.

Theorem 1 (Avoid Over-Smoothing). *Assuming the graph is connected and unweighted, the propagation defined in* (21) *can avoid over-smoothing regardless of the sparsity (i.e., k in KNN) of the graph.*

The proof of Theorem 1 is done by showing that the propagation in (21) can induce both low and high-frequency dominant dynamics (LFD and HFD) under the mild conditions, regardless of the sparsity of the graph. Without loss of generality, we let g_θ be any GNN model. We will also denote $\widehat{\boldsymbol{L}} = \boldsymbol{I} - \widehat{\boldsymbol{A}} = \boldsymbol{I} - \mathbf{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\mathbf{D}^{-\frac{1}{2}}$ as the normalized graph Laplacian matrix, and let $\rho_{\widehat{\boldsymbol{L}}}$ be the spectral radius (i.e., the difference between the smallest and largest eigenvalue) of $\widehat{\boldsymbol{L}}$. We then introduce the following definitions of LFD and HFD.

Definition 3. *$\dot{\boldsymbol{H}}(t) = g_\theta(\boldsymbol{H}(t), t)$ is LFD if $E\big(\boldsymbol{H}(t)/\|\boldsymbol{H}(t)\|\big) \to 0$ as $t \to \infty$, and is HFD if $E\big(\boldsymbol{H}(t)/\|\boldsymbol{H}(t)\|\big) \to \rho_{\widehat{\boldsymbol{L}}}/2$ as $t \to \infty$.*

Our definitions of LFD and HFD are adopted from the recent work in [12]. We note that although the phenomenon defined in Definition 3 is defined via the continuous domain (i.e., t), it is

straightforward to analogize the definition to the discrete domain, i.e., layers of GNNs. Another important remark here is that if one GNN model is LFD, then the model's smoothing effect on the node features will dominate the entire dynamic; thus, asymptotically, the features will become identical, causing the over-smoothing phenomenon. On the other hand, if a GNN model can induce an HFD dynamic, the features will tend to be dissimilar to each other, causing the GNN to avoid the problem. Accordingly, aligning with the conclusion introduced in [12], one can further characterize the notion of LFD and HFD in the following Lemma:

Lemma 1 (LFD/HFD Behaviors [12]). *A GNN model is LFD (resp. HFD) if and only if for each $t_j \to \infty$, there exists a sub-sequence indexed by $t_{j_k} \to \infty$ and $\boldsymbol{H}_\infty$ such that $\boldsymbol{H}(t_{j_k})/\|\boldsymbol{H}(t_{j_k})\| \to \boldsymbol{H}_\infty$ and $\widehat{\boldsymbol{L}}\boldsymbol{H}_\infty = 0$ (resp. $\widehat{\boldsymbol{L}}\boldsymbol{H}_\infty = \rho_{\widehat{L}}\boldsymbol{H}_\infty$).*

The detailed proof of the Lemma can be found in [12], and it is straightforward to generalize the Lemma to the feature frequency representation (i.e., S); thus, we omit it here. We now demonstrate that our proposed framelet model, with the propagation defined in (21), can induce both LFD and HFD-type dynamics as follows. We note that for simplicity reasons, the following conclusion is for the framelet model with Haar type filtering function [44] of scale one (i.e., J = 1) in (21), and we have

$$\mathcal{W}_{0,1} = \boldsymbol{U}\boldsymbol{\Lambda}_{0,1}\boldsymbol{U}^\top = \boldsymbol{U}\cos(\boldsymbol{\Lambda}/8)\boldsymbol{U}^\top,$$
$$\mathcal{W}_{1,1} = \boldsymbol{U}\boldsymbol{\Lambda}_{1,1}\boldsymbol{U}^\top = \boldsymbol{U}\sin(\boldsymbol{\Lambda}/8)\boldsymbol{U}^\top.$$

Accordingly, the feature propagation defined in (21) becomes

$$\boldsymbol{H}(\ell+1) = \boldsymbol{U}\mathrm{diag}(\boldsymbol{\theta}_{0,1})\cos(\boldsymbol{\Lambda}/8) + \mathrm{diag}(\boldsymbol{\theta}_{1,1})\sin(\boldsymbol{\Lambda}/8)\boldsymbol{U}^\top\boldsymbol{H}(\ell)\boldsymbol{W}(\ell), \quad (22)$$

and we have the following conclusion.

Lemma 2. *Assuming the graph is connected and unweighted, the framelet message-passing model defined in* (21) *with Haar-type filter of scale 1 can induce both LFD and HFD dynamics. Specifically, let $\boldsymbol{\theta}_{0,1} = \mathbf{1}$ and $\boldsymbol{\theta}_{1,1} = \theta\mathbf{1}$ where $\mathbf{1}$ is a vector of all* 1*s. Suppose $\theta \geq 0$. Then, when $\theta \in [0, 1)$, the dynamic in* (21) *LFD; otherwise, the model is HFD.*

Proof. The proof of the Lemma extends the conclusion in [15] when the framelet model is without reconstruction. First, one can assume the propagation in (22) as the gradient flow (e.g., with stepsize τ) of the evolution of the quadratic or energy term, and the channel-mixing matrix $\mathbf{W}$ is symmetric [12, 15], plunging in the settings in Lemma 2, result in the following.

$$\begin{aligned}\mathrm{vec}\big(\boldsymbol{H}(m\tau)\big) &= \tau^m\Big(\mathbf{W}\otimes(\mathcal{W}_{0,1}+\theta\mathcal{W}_{1,1})\Big)^m\mathrm{vec}\big(\boldsymbol{H}(0)\big),\\ &= \tau^m\sum_{k,i}\Big(\lambda_k^W\big(\cos(\lambda_i/8)+\theta\sin(\lambda_i/8)\big)\Big)^m c_{k,i}(0)\phi_k^W\otimes\mathbf{u}_i, \quad (23)\end{aligned}$$

where we denote $\ell = m\tau$ and $\{(\lambda_k^W, \phi_k^W)\}_{k=1}^c$ as the eigenvalue and eigenvector pairs of $\mathbf{W}$ and $c_{k,i}(0) := \langle\mathrm{vec}(\boldsymbol{H}(0)), \phi_k^W\otimes\mathbf{u}_i\rangle$ as the projection of the initial node features to the domain constructed by $\phi_k^W\otimes\mathbf{u}_i$. Now one can check that

$$|\lambda_k^W(\cos(\lambda_i/8)+\theta\sin(\lambda_i/8))| \leq \Delta^W(\cos(\lambda_i/8)+\theta\sin(\lambda_i/8)),$$

where $\Delta^W := \max_k|\lambda_k^W|$. One can further check that for example when $\theta > 1$ the function $\cos(\lambda_i/8)+\theta\sin(\lambda_i/8)$ is monotonically increasing in $\lambda_i \in [0, \rho_{\widehat{L}}]$ and its maximum is achieved at $\lambda_i = \rho_{\widehat{L}}$ (with sufficiently large λ_k^W). On the other hand, with small θ, the function is monotonically decreasing and its maximum is achieved at $\lambda_i = 0$. More specifically, one can let $\delta := \max_{i:\lambda_i\neq\rho_{\widehat{L}}}|\big(\lambda^W((\lambda_i^{\Lambda_{0,1}})+\theta(\lambda_i^{\Lambda_{1,1}}))\big)|$. Also denote $\mathbf{P}_\rho = \sum_k(\boldsymbol{\phi}_k\otimes\mathbf{u}_\rho)(\boldsymbol{\phi}_k\otimes\mathbf{u}_\rho)^\top$ where $\mathbf{u}_\rho$ is the eigenvector of $\widehat{\boldsymbol{L}}$ associated with eigenvalue $\rho_{\widehat{L}}$. Then we can decompose (23) as

$$\begin{aligned}\mathrm{vec}\big(\boldsymbol{H}(m\tau)\big) &= \tau^m\sum_k\delta_{\mathrm{HFD}}^m c_{k,\rho_L}(0)\mathbf{u}_k\otimes\mathbf{u}_\rho\\ &+\tau^m\sum_k\sum_{i:\lambda_i\neq\rho_L}\Big(\big(\lambda^W(\lambda_i^{\Lambda_{0,1}})+\theta(\lambda_i^{\Lambda_{1,1}})\big)\Big)^m c_{k,i}(0)\boldsymbol{\phi}_k\otimes\mathbf{u}_\rho\\ &\leq \tau^m\delta_{\mathrm{HFD}}^m(\mathbf{P}_\rho\mathrm{vec}\big(\boldsymbol{H}(0)\big)+\sum_k\sum_{i:\lambda_i\neq\rho_{\widehat{L}}}\left(\frac{\delta}{\delta_{\mathrm{HFD}}}\right)^m c_{k,i}(0)\boldsymbol{\phi}_k\otimes\mathbf{u}_\rho,\end{aligned} \quad (24)$$

where $\delta < \delta_{\mathrm{HFD}}$. One can then normalize the results of the above derivation, one can see that $\frac{\mathrm{vec}\left(\boldsymbol{H}(m\tau)\right)}{\|\mathrm{vec}\left(\boldsymbol{H}(m\tau)\right)\|} \to \frac{\mathbf{P}_\rho(\mathrm{vec}(\boldsymbol{H}(0)))}{\|\mathbf{P}_\rho\mathrm{vec}(\boldsymbol{H}(0))\|}$ as $m \to \infty$, where the latter term is a unit vector $\mathbf{h}_\infty$ satisfying $(\boldsymbol{I}_D\otimes\widehat{\boldsymbol{L}})\mathbf{h}_\infty = \rho_L\mathbf{h}_\infty$, where D is the feature dimension. This directly suggests the dynamic is HFD according to the Definition 3 and Lemma 1. This completes the proof. □

Remark 1. *Based on the definition of LFD and HFD, one can check that there are other ways of achieving LFD and HFD for the dynamic in* (22)*, and our settings in Lemma 2 are for illustration purposes to show our proposed model has such capability. In addition, Lemma 2 conclusion can be directly extended to other filtering functions rather than* Haar *and scales (i.e., $J > 1$), we omit the proof here.*

Based on Lemma 2, one can check that when the model dynamics is HFD, e.g., the model with a relatively large θ, node features tend to be dissimilar to each other, thus avoiding the over-smoothing problem. More importantly, Lemma 2 is still valid when the graph is dense, and this directly proves the claim in Theorem 1, further reflecting that a task-adaptive graph-forming technique can consistently benefit our model once a carefully selected GNN is paired with it to prevent the potential computational issues. We refer to the more detailed theoretical discussion in the works in [32, 34].

Remark 2. *Our final remark is that in the case where the graph of N nodes is complete, the eigenvalues of the normalized graph Laplacian are 0 with multiplicity 1 and 1 with multiplicity $N-1$. One can verify that without edge reweighting and rewiring, even multiscale GNNs (e.g., framelets) have limited filtering power. This also verifies the necessity of leveraging an adaptive, preferably sparse graph via graph neural network (GNN) propagation.*

3.6 Forecasting Pipeline with UFGTime

The main framework of UFGTime is illustrated in Figure 2. Given input multivariate time series data $\boldsymbol{X}_t \in \mathbb{R}^{N\times T\times D}$, we first apply moving-average decomposition to the input to extract trend information $\boldsymbol{X}_t^{trend} \in \mathbb{R}^{N\times T\times D}$, and then apply the Discrete Fourier Transform (DFT) on the time dimension of the input to obtain the

Algorithm 1: UFGTime

Input: $\mathcal{X} = \{X_1, X_2, \ldots, X_t, \ldots\}, X_t \in \mathbb{R}^{N\times T\times D}$, number of neighbors k, number of high-pass filters L, scale level J, coarsest scale m, and maximum iterations I

Output: Predicted results $\widehat{\mathcal{Y}} = \{\widehat{Y}_1, \widehat{Y}_2, \ldots, \widehat{Y}_t, \ldots\}$

while $i = 0$ **to** I **do**
  for X_t *in* $\mathcal{X}$ **do**
    Apply $\mathrm{MA}(X_t) \rightarrow X_t^{trend} \in \mathbb{R}^{N\times T\times D}$;
    Apply $\mathcal{F}(X_t, C) \rightarrow S_t \in \mathbb{C}^{N\times C\times D}$;
    Reshape $S_t \rightarrow S_t^G \in \mathbb{C}^{NC\times D}$;
    Apply $\mathrm{KNN}(X_t^G, k) \rightarrow A_t \in \{0,1\}^{NC\times NC}$;
    Form $G_t^S = \left(S_t^G, A_t\right)$;
    Generate graph Laplacian $\widehat{L} \leftarrow A_t$
    for $r = 0$ **to** $L-1$ **do**
      $\mathcal{W}_{r,j} = \begin{cases} \mathcal{T}_0^K(2^{-m}\mathcal{L}) & j = 1, \\ \mathcal{T}_r^K(2^{-(m+j-1)}\mathcal{L})\ldots\mathcal{T}_0^K(2^{-m}\mathcal{L}) & j = 2, ..., J \end{cases}$
    $H(0) \leftarrow S_t^G$;
    $H(\ell) = \mathrm{SiLU}\sum_{r,j}\mathrm{diag}(\boldsymbol{\theta}_{r,j})\mathcal{W}_{r,j}S_t^G(\ell-1)W(\ell)$;
    Reshape $H(\ell) \rightarrow S_t(\ell) \in \mathbb{C}^{N\times C\times D}$;
    Apply $\mathcal{F}^{-1}(S_t(\ell), T) \rightarrow X_t^H \in \mathbb{R}^{N\times T\times D}$;
    $\widehat{Y}_{t+1} = \mathrm{FFN}(X_t^H) + X_t^{trend}W_{trend}$.
return Predicted results $\widehat{\mathcal{Y}}$;

spectral signal $S_t \in \mathbb{C}^{N\times C\times D}$. The frequency signal is reshaped into a spectral-variate graph feature $S_t^G \in \mathbb{C}^{NC\times D}$. Next, we leverage KNN to generate a graph structure $A_t \in \{\mathbf{0}, \mathbf{1}\}^{NT\times NT}$. At this point, we obtain the spectral-variate graph $G_t^S = \left(S_t^G, A_t\right)$. Subsequently, to capture intricate dependencies on the spectral-variate graph, we feed the data into ℓ layers of a global framelet message-passing function with a SiLU activation function, defined as $H(\ell) = \mathrm{SiLU}\left(\sum_{r,j}\mathrm{diag}(\boldsymbol{\theta}_{r,j})\mathcal{W}_{r,j}S_t^G(\ell-1)W(\ell-1)\right)$. Afterward, we reshape $H(\ell)$ into frequency signal $S_t(\ell) \in \mathbb{C}^{N\times C\times D}$ and use the Inverse Fast Fourier Transform (IFFT) $\mathcal{F}^{-1}(S_t(\ell))$ to obtain the graph operation output $X_t^H \in \mathbb{R}^{N\times T\times D}$. Finally, based on the output hidden state X_t^H, which encodes both intra- and inter-temporal dependencies, we apply a two-layer feed-forward network (FFN) to project it forward τ steps. This result is combined with the trend embedding to yield the final output:

$$\widehat{Y}_{t+1} = \mathrm{FFN}(X_t^H) + X_t^{trend}W_{trend}. \tag{25}$$

We provide the full pipeline of our method, which is shown in Algorithm 1.

3.7 Model Complexity Analysis

For simplicity, we assume the weights of our Fourier-domain framelet message-passing function are in $\mathbb{C}^{D\times D}$, and the Fourier signals have length T identical to the original time series. Our framework consists of two main stages: spectral-variate graph construction and the graph framelet operation. The complexity of graph construction is dominated by the Fourier transformation $\mathcal{F}(\cdot)$, which is $O(NT\log(T))$. For the framelet operation on the sparse spectral-variate graph, the complexity per layer is initially $O(R(J+1)(NT)^2D + NTD^2)$, but due to graph sparsity enforced by the KNN ($|\mathcal{E}| \leq kNT$), it reduces to $O(L(J+1)NTkD + NTD^2)$. Considering that $\log(T) \ll D^2$ and constants L, J, and k are relatively small, the overall complexity simplifies further to $O(NTkD + NTD^2)$. This demonstrates the high efficiency of UFGTime, reducing the original quadratic complexity $O((NT)^2)$ to linear complexity $O(NT)$.

Table 2: Summary of Dataset Statistics and Characteristics

Datasets	#Samples	#Nodes	Granularity	Start time	Split	Characteristics
Solar-FL	4,380	593	2 hour	2006-01-01	7/2/1	stationarity
Wiki-500	803	500	1 day	2015-01-07	7/2/1	shift, stationarity
Traffic	10,560	963	1 hour	2015-01-01	7/2/1	stationarity
Ecg	4,999	140	unknown	unknown	7/2/1	stationarity, shift
Electricity2H	4,380	370	2 hour	2014-01-01	7/2/1	stationarity
Covid-Cal	345	60	1 day	2020-01-22	7/2/1	trend
Fred-MD	728	107	1 month	1959-01-01	7/2/1	trend, n/stationarity
Exchange	7,588	8	1 day	1990-01-01	7/2/1	shift, n/seasonality
Nasdaq	1,244	5	1 day	unknown	7/2/1	n/seasonality
Nyse	1,243	5	1 day	unknown	7/2/1	n/seasonality
NN5	791	111	1 day	1996-03-18	7/2/1	stationarity
Ili	966	7	1 week	2002-01-02	7/2/1	n/stationarity
ETTm	69,680	7	15 minutes	2016-07-01	6/2/2	transition
ETTh	17,420	7	1 hour	2016-07-01	6/2/2	transition

4 EMPIRICAL EVALUATION

4.1 Experimental Setup

4.1.1 Datasets. We evaluate our proposed method on 14 multivariate time series datasets spanning diverse domains, including healthcare, energy, transportation, online activity, and finance (details summarized in Table 2). The ECG[1] dataset contains recordings of patients' heartbeats but lacks temporal granularity and start time metadata. Solar-FL[2] comprises synthetic solar photovoltaic (PV) power generation data from plants located in Florida. Covid-Cal[3] includes annual confirmed COVID-19 case counts from hospitals across 60 counties in California, sourced from the CSSE COVID-19 repository. The Electricity2H[4] dataset, derived from the UCI electricity load database, was downsampled to 2-hour intervals. Wiki-500[5] consists of 500 randomly selected Wikipedia page view series. Traffic[6] provides hourly traffic volume measurements from freeway sensors in San Francisco. Fred-MD[7] contains macroeconomic indicators collected by the Federal Reserve Bank. The Exchange dataset records daily exchange rates for eight foreign currencies spanning 26 years. NASDAQ and NYSE[8] include historical stock

[1] https://timeseriesclassification.com/description.php?Dataset=ECG5000
[2] https://www.nrel.gov/grid/solar-power-data.html
[3] https://github.com/CSSEGISandData/COVID-19
[4] https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014
[5] https://drive.google.com/uc?export=download&id=1VytXoL_vkrLqXxCR5IOXgE45hN2UL5oB
[6] https://drive.google.com/uc?export=download&id=1dyeYj8IJwZ3bKvk1H67eaDTANdapKe7w
[7] https://zenodo.org/records/4654833
[8] https://github.com/fulifeng/Temporal_Relational_Stock_Ranking

Table 3: Short-Term Forecasting Results on Twelve Datasets with Average Rankings. Best and Second Best Results Per Dataset are Highlighted in Red and Blue, Respectively. Rankings are Based on Average MAE, RMSE, and MAPE Across Datasets. Ecg, Nasdaq, and Nyse Results for Partial Transformer-based Methods are Denoted as '–' Due to Missing Temporal Information.

Baselines	Solar-FL			Wiki-500			Traffic			Ecg			Electricity2H			Covid-Cal		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Autoformer	0.1078	0.1489	3.4948	0.1936	0.3917	2.8225	0.0669	0.1018	0.9734	-	-	-	0.0961	0.1273	0.4902	0.6654	1.1961	0.3363
Informer	0.0827	0.1296	3.4536	0.1255	0.3183	2.4051	0.0522	0.0837	0.6640	-	-	-	0.1241	0.1611	0.6116	2.6893	4.7431	0.8996
Pyraformer	0.1451	0.1862	3.5104	0.0957	0.2651	2.0245	0.0466	0.0768	0.6961	-	-	-	0.1525	0.1986	0.8710	3.4571	5.4846	0.9987
Crossformer	0.0858	0.1281	3.4378	0.1566	0.2927	2.7246	0.0642	0.0940	1.0889	0.0592	0.0850	0.1335	0.1403	0.1750	0.8241	2.1863	4.6706	0.5858
Dlinear	0.0895	0.1351	3.4382	0.0594	0.3159	1.4073	0.0655	0.1036	0.9161	0.0544	0.0814	0.1182	0.0859	0.1193	0.4912	0.2045	0.4458	0.2115
Dcrnn	0.4772	0.5995	3.8203	0.4397	0.5655	3.6791	0.4404	0.5507	3.2122	0.6491	0.7858	1.1309	0.5532	0.6879	1.8591	3.9790	5.9690	1.1232
Stgcn	0.0873	0.1351	3.4544	0.0761	0.1901	1.7022	0.0356	0.0619	0.5197	0.0642	0.0923	0.1472	0.1155	0.1587	0.6625	3.2116	5.4279	0.8565
Gwnet	0.0838	0.1339	3.4561	0.0513	0.1698	1.2301	0.0354	0.0638	0.5194	0.0564	0.0833	0.1231	0.0782	0.1201	0.4536	2.4842	5.0064	0.6153
Mtgnn	0.0843	0.1343	3.4613	0.0518	0.1711	1.2702	0.0353	0.0619	0.5199	0.0557	0.0824	0.1222	0.0834	0.1235	0.5106	2.4513	4.2893	0.6835
Stemgnn	0.1558	0.2002	3.4951	0.2004	0.2977	3.0139	0.0694	0.1028	1.0486	0.1147	0.1496	0.2577	0.2929	0.3598	1.0889	3.9085	5.8803	1.1068
Agcrn	0.2169	0.3441	3.4381	0.5697	0.6508	3.9500	0.0973	0.1336	1.4485	0.0991	0.1320	0.2286	0.1735	0.2193	0.9563	3.5163	5.6340	0.9627
Fouriergnn	0.0809	0.1245	3.4414	0.1040	0.2246	2.2089	0.0403	0.0696	0.5908	0.0565	0.0879	0.1366	0.0927	0.1359	0.5589	0.2729	0.5113	0.2345
Ufgtime	0.0809	0.1259	3.4372	0.0471	0.1696	0.8746	0.0351	0.0618	0.5191	0.0536	0.0806	0.1173	0.0752	0.1164	0.0455	0.1918	0.4488	0.2051

Baselines	Fred-MD			Exchange			Nasdaq			Nyse			NN5			Ili		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Autoformer	0.2085	1.2868	0.2350	0.0305	0.0500	0.0797	–	–	–	–	–	–	0.1030	0.1504	0.3972	0.1850	0.2767	0.6109
Informer	1.1117	5.6775	0.7958	0.0803	0.0990	0.1850	–	–	–	–	–	–	0.0995	0.1440	0.3857	0.2579	0.3582	0.7265
Pyraformer	0.1959	1.2032	0.2306	0.0300	0.0494	0.0679	–	–	–	–	–	–	0.1013	0.1518	0.3919	0.1792	0.2651	0.6018
Crossformer	1.6651	6.5407	1.3223	0.0357	0.0517	0.0839	0.0903	0.1249	0.0917	0.0882	0.1172	0.6057	0.2450	0.3056	0.8243	0.1867	0.2893	0.3873
Dlinear	0.1457	0.9382	0.1869	0.0254	0.0437	0.0780	0.0709	0.1037	0.0763	0.0408	0.0612	0.3037	0.1092	0.1590	0.4238	0.1775	0.2668	0.6042
Dcrnn	1.9966	7.0783	1.3772	0.6065	0.7544	1.3214	0.9298	1.1339	0.9613	0.4369	0.5383	2.4912	0.5109	0.6391	1.4664	0.7532	0.9625	1.7924
Stgcn	1.3315	6.8512	0.8610	0.0425	0.0638	0.1618	0.2492	0.3247	0.2100	0.0590	0.0720	0.4345	0.0967	0.1343	0.3377	0.3534	0.4430	1.0736
Gwnet	0.9881	3.6123	1.2256	0.0289	0.0439	0.0919	0.1450	0.1826	0.1451	0.0439	0.0546	0.3052	0.0954	0.1352	0.3376	0.2304	0.3269	0.7119
Mtgnn	0.3808	1.9662	0.3931	0.0284	0.0435	0.0926	0.1118	0.1490	0.1135	0.0599	0.0742	0.4520	0.0906	0.1283	0.3208	0.2010	0.2924	0.6861
Stemgnn	1.1810	6.7190	0.6430	0.0341	0.0533	0.1224	0.1493	0.1914	0.1376	0.2506	0.2750	1.5064	0.1074	0.1441	0.3907	0.2920	0.3960	0.6489
Agcrn	1.0822	6.1587	0.5804	0.0267	0.0413	0.0829	0.0687	0.1050	0.0725	0.0848	0.1252	0.5822	0.0898	0.1271	0.3257	0.1826	0.2643	0.5980
Fouriergnn	1.3784	4.7566	1.1396	0.0488	0.0709	0.1687	0.0872	0.1229	0.0913	0.1766	0.2164	1.0829	0.0956	0.1308	0.3305	0.1849	0.2863	0.5917
Ufgtime	0.1052	0.5343	0.1279	0.0263	0.0409	0.0786	0.0669	0.1013	0.0710	0.0300	0.0437	0.1932	0.0951	0.1309	0.3358	0.1770	0.2603	0.5988

transaction data from the NASDAQ and NYSE exchanges, respectively. The NN5[9] dataset records daily cash withdrawal volumes from 111 ATMs across England. Finally, the ILI[10] dataset includes weekly influenza-like illness (ILI) cases reported by the U.S. CDC. For long-term forecasting, we use two ETT datasets (ETTm/h)[11]. These datasets include electricity data from two transformer stations (ETTm/h (1/2), we select ETTm/h1) in China, each with 7 time series (e.g., electricity load, oil temperature).

We adopt the original train-validation-test splits from prior studies [23, 45, 52], as indicated in the "Split" column. To ensure a comprehensive evaluation, we follow the time series attribution method described in [29] to select datasets covering five representative characteristics. Detailed dataset statistics are summarized in Table 2.

4.1.2 Baselines. To comprehensively evaluate our proposed method alongside state-of-the-art approaches in multivariate time series forecasting, we select a variety of representative baselines classified into three main categories.

- *Transformer-based methods*: Crossformer [49], Autoformer [39], Informer[52], and Pyraformer [22].
- *Graph-based methods*: DCRNN [21], STGCN [46], GWNet [41], MTGNN [40], StemGNN [5], AGCRN [3], and Fouriergnn [45].
- *Linear-based methods*: DLinear [48], and TiDE [10].

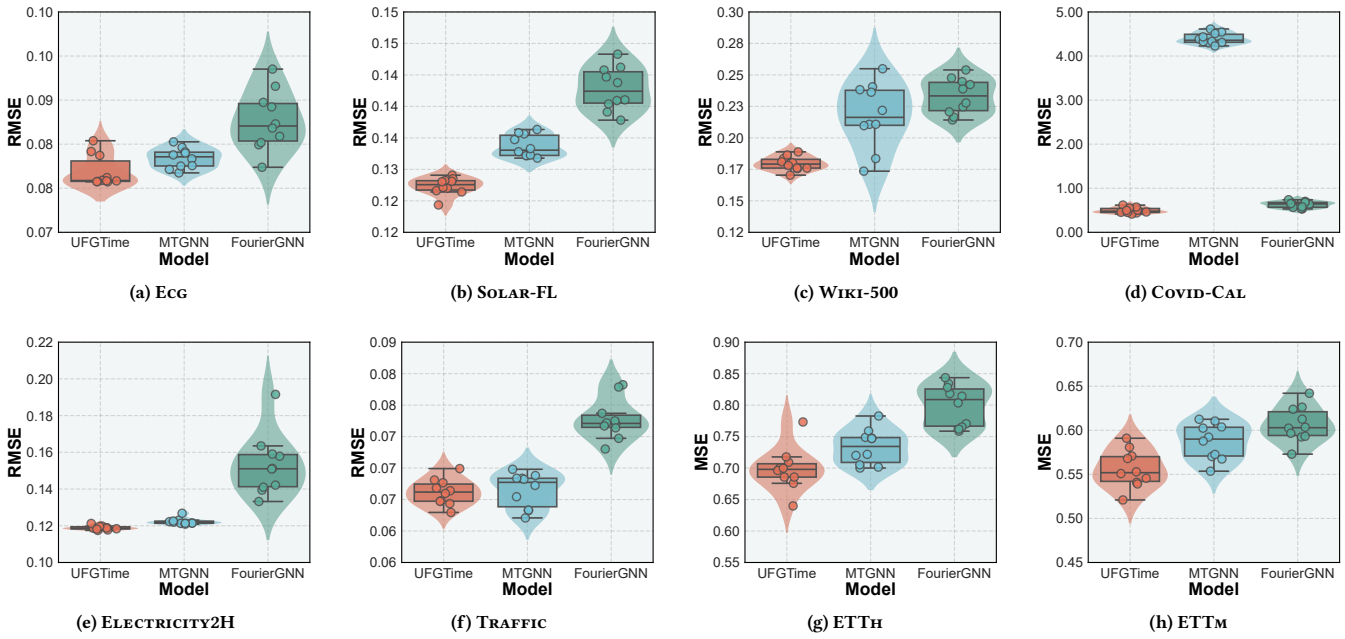
4.1.3 Implementation. We reproduce the baseline models using modified scripts from FourierGNN [45] and the fair benchmarking toolkit, BasicTS+[12] [33]. Our model and all baselines are fine-tuned using the Optuna[13] toolkit [1] with RMSprop optimizers to minimize the Mean Squared Error (MSE) loss.

4.1.4 Evaluation Metrics. Following the evaluation methodology in prior works [23, 45, 52], we adopt multiple metrics to comprehensively assess the performance of our method. For multi-step forecasting accuracy, we use mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE). To evaluate module contributions in the ablation study, we perform

[9] https://rdrr.io/cran/TSPred/man/NN5.A.html
[10] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html
[11] https://github.com/zhouhaoyi/ETDataset
[12] https://github.com/GestaltCogTeam/BasicTS
[13] https://optuna.org/

Table 4: Long-Term Multivariate Time Series Forecasting Results on ETT Datasets. Best and Second Best Results Per Dataset Highlighted in Red and Blue Respectively.

Datasets	Steps	UFGTime		FourierGNN		Crossformer		TiDE		DLinear		Pyraformer		Autoformer		Informer		FEDformer		StemGNN	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm	96	0.324	0.357	0.581	0.462	0.375	0.415	0.364	0.387	0.339	0.380	0.543	0.510	0.505	0.475	0.672	0.571	0.456	0.490	0.528	0.536
	192	0.370	0.385	0.904	0.643	0.453	0.474	0.398	0.404	0.391	0.410	0.557	0.537	0.573	0.509	0.795	0.669	0.517	0.524	0.693	0.610
	336	0.404	0.421	0.919	0.646	0.548	0.526	0.428	0.425	0.433	0.438	0.754	0.655	0.621	0.537	1.212	0.871	0.570	0.557	0.667	0.625
	720	0.515	0.473	0.927	0.648	0.857	0.713	0.487	0.461	0.489	0.481	0.908	0.724	0.749	0.5694	1.307	0.893	0.613	0.588	0.689	0.624
	Avg	0.412	0.409	0.833	0.600	0.563	0.532	0.419	0.419	0.413	0.427	0.691	0.607	0.612	0.523	0.997	0.751	0.539	0.540	0.644	0.599
ETTh	96	0.416	0.418	0.476	0.495	0.441	0.457	0.479	0.464	0.451	0.475	0.664	0.612	0.449	0.459	0.865	0.713	0.551	0.544	0.609	0.590
	192	0.449	0.445	0.547	0.571	0.521	0.503	0.525	0.492	0.496	0.506	0.790	0.681	0.500	0.482	1.008	0.792	0.612	0.580	0.742	0.650
	336	0.474	0.471	1.173	0.574	0.659	0.603	0.569	0.551	0.536	0.535	0.891	0.738	0.521	0.496	1.107	0.809	0.643	0.600	0.685	0.635
	720	0.634	0.575	0.733	0.716	0.893	0.736	0.770	0.672	0.650	0.610	0.963	0.782	0.514	0.512	1.181	0.865	0.788	0.681	0.777	0.683
	Avg	0.531	0.477	0.567	0.589	0.628	0.574	0.541	0.507	0.533	0.532	0.827	0.703	0.496	0.487	1.040	0.794	0.649	0.601	0.703	0.639



(a) Ecg (b) Solar-FL (c) Wiki-500 (d) Covid-Cal

(e) Electricity2H (f) Traffic (g) ETTh (h) ETTm

Figure 4: Robustness Comparison between Our Method and Baselines (MTGNN, FourierGNN) across 8 Datasets. Results are Illustrated Using Box Plots and Density Plots based on 10 Repeated Experiments with Random Initialization. All Results are Reported in RMSE, Except for the ETTh/m Dataset, Which uses MSE.

a Two-Way ANOVA test and report the p-statistic results. Additionally, we measure computational efficiency using Gflop/s to compare the computational cost across baselines.

4.1.5 Hardware and Setting. Our experiments were conducted on a server equipped with an AMD EPYC 7J13 64-core CPU, 256 GB of RAM, and four NVIDIA GeForce RTX 4090 GPUs. This setup was used to evaluate our and baseline models across all datasets.

4.2 Overall Performance Analysis

4.2.1 Can UFGTime effectively capture temporal patterns from short input sequences? The performance of short-term multivariate time series forecasting is presented in Table 3, where both the history window and forecasting length are set to 12. Notably, some transformer-based methods, such as Autoformer, Informer, and Pyraformer, fail to produce results on the ECG, NASDAQ, and NYSE datasets due to the absence of timestamp information. Compared to all state-of-the-art baselines, UFGTime demonstrates competitive performance. Specifically, on datasets such as Covid-Cal, Fred-MD, Exchange, NASDAQ, NYSE, and ILI, which exhibit strong non-stationary patterns, UFGTime effectively captures complex dynamic patterns that often challenge transformer-based methods. This highlights the strength of our framework's spectral-variate

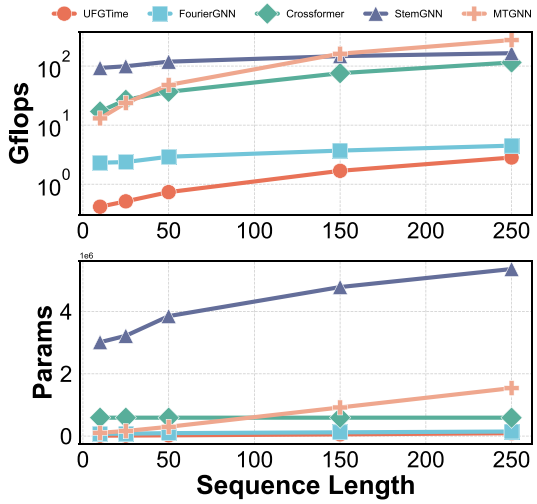
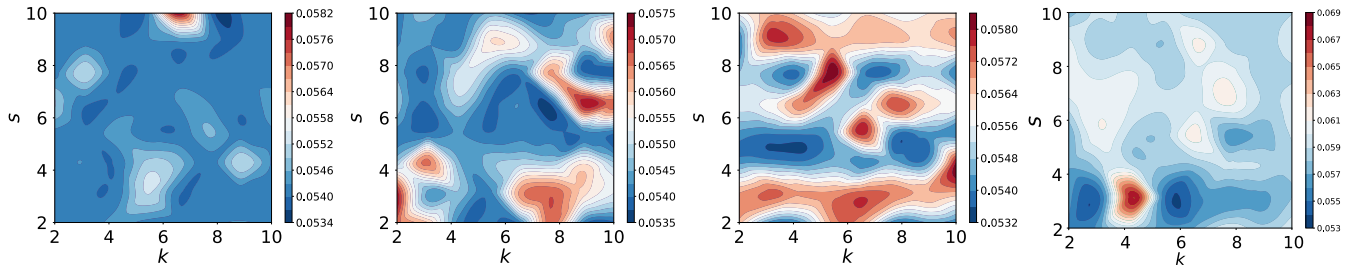


Figure 5: Scalability Test (Computational Cost and Parameters) on Various Lengths

Table 5: Comparison of Parameters and Computational Costs for Various Model Hidden Size on the Wiki-500 Dataset with a Batch Size of 32. Computational Costs are shown in Gflop/s.

Baselines	Hidden 32		Hidden 64		Hidden 128		Hidden 256	
	Param	Gflop/s	Param	Gflop/s	Param	Gflop/s	Param	Gflop/s
StemGNN	1,800,504	57.5376	1,800,504	57.5376	1,800,504	57.5376	1,800,504	57.5376
Crossformer	588,216	30.2640	1,398,616	70.0281	3,707,544	178.0567	11,077,912	508.1150
MTGNN	106,268	13.5128	195,548	26.5872	374,108	52.7361	731,228	105.0339
GWNET	68,588	6.9514	270,284	27.5671	1,073,036	54.8956	4,275,980	876.4211
FourierGNN	68,076	2.1748	70,540	2.2528	75,368	2.4084	85,324	2.7197
Pyraformer	905,800	0.2190	1,819,424	0.4494	3,700,432	0.9446	7,677,488	2.0720
Autoformer	570,996	0.2192	1,174,260	0.4507	2,479,092	0.9517	5,481,972	2.1046
Informer	183,348	0.1664	404,340	0.3642	963,060	0.8542	2,547,444	2.2118
DCRNN	33,672	0.0307	657,496	0.4915	129,366	1.2458	257,482	2.4915
STGCN	97,452	0.0246	196,940	0.1311	402,060	1.5729	836,876	2.6214
UFGTime	3,690	0.1245	7,261	0.2341	14,045	0.4532	27,613	0.8915
AGCRN	3,960	0.1232	7,080	0.2331	13,480	0.4321	26,840	0.8531
DLinear	312	0.0048	312	0.0048	312	0.0048	312	0.0048



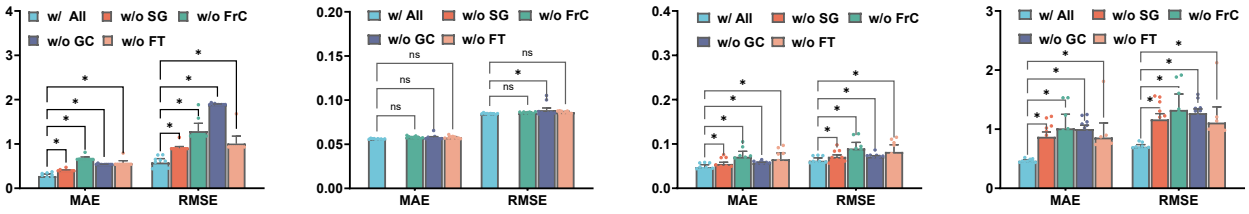
(a) Hidden Size: 32 **(b) Hidden Size:** 64 **(c) Hidden Size:** 128 **(d) Hidden Size:** 256

Figure 6: Contour Plots of Hyperparameters Surface with Different Settings of Number of Graph Neighbors k (2-10), Framelet Dilation Scale s (2-10), and Hidden Size (32, 64, 128, 256)

graph construction and robustness of graph framelet design, underscoring its effectiveness in multivariate time series forecasting.

4.2.2 Is UFGTime still effective in extracting long-term temporal relationships? As discussed by Yi et al. [45], pure graph paradigms tend to focus on dynamic patterns rather than long-range dependencies, such as periodic patterns and trends, potentially limiting their performance in long-range forecasting tasks. To mitigate this limitation, our refined pure graph method introduces a spectral-variate graph to construct pure graphs from various global Fourier signals, integrates a global filter within the framelet message-passing function, and incorporates trend information. These enhancements allow our model to effectively capture global patterns, making it possible for long-term forecasting. Building on this insight, we evaluate our method and FourierGNN on two widely used ETT long-range forecasting datasets for prediction horizons of 96, 192, 336, and 720 steps. Comparison results with state-of-the-art long-range forecasting baselines are summarized in Table 4. While we notice some performance loss for very long-range forecasting (720 steps), it delivers competitive results overall, matching the performance of certain baselines specifically designed for such tasks. These findings underscore the effectiveness of our carefully designed mechanisms for preserving global patterns, demonstrating their significant contributions to long-range predictions.

4.2.3 Is UFGTime robust compared to other GNN baselines? We analyze the robustness of our method against two representative graph-based baselines: MTGNN and FourierGNN. Figure 4 illustrates the robustness evaluation results across all datasets, where RMSE is used for the six short-term datasets and MSE for the two long-term ones. The results demonstrate that UFGTime consistently achieves the lowest average prediction errors with relatively narrow distribution widths across most datasets, indicating its superior robustness and stability in diverse forecasting scenarios. However, for the ETTh dataset, UFGTime exhibits occasional outliers, suggesting some variability under specific conditions. In contrast, MTGNN and FourierGNN yield higher errors and broader variability distributions, particularly noticeable on complex datasets such as Wiki-500 and ETTh, reflecting their comparatively lower robustness.



(a) Covid-Cal (b) Ecg (c) Nyse (d) ETTh

Figure 7: Ablation Study of Key Designs of UFGTime on Four Selected Dataset. Differences were Analyzed using a Two-Way ANOVA Test. "*" Indicate Statistical Significant at the 90% Level, While "ns" Denotes No Statistically Significant.

4.3 Resource Utilization Analysis

In Table 5, we compare the resource utilization of all baselines. To ensure fairness and eliminate hardware variability, we use THOP[14] to measure key efficiency metrics, including the number of parameters and giga floating-point operations per second (Gflop/s). We assess resource consumption across varying hidden size configurations to evaluate efficiency with respect to model size. As expected, DLinear demonstrates outstanding efficiency due to its simple single-linear architecture. Compared to other baselines, both AGCRN and our method show strong size efficiency, outperforming most models. Notably, when compared to the pure graph-based FourierGNN, our method achieves comparable or better efficiency while requiring only about 1/6 of the computational resources, even under multiple model size settings.

We further evaluate the scalability of various methods with different sequence lengths, as shown in Figure 5. The results indicate that pure graph methods, including our approach, demonstrate strong scalability in terms of computational cost and parameter growth, with parameters increasing linearly as input lengths grow. This underscores the compact design of our method, characterized by its modular structure and lower computational complexity, which ensures robustness and high scalability across varying input sizes.

4.4 Hyperparameter Sensitivity Analysis

In this section, we conduct a sensitivity study of our proposed method. To evaluate the impact of the model's hyperparameters, we perform a grid search on the **ECG** dataset, exploring three architectural hyperparameters: framelet dilation scale s, number of graph neighbors k, and hidden size. To better visualize the hyperparameter landscape, we interpolate the grid search results and present them as contour plots in Figure 6.

The main observations are as follows: ***Is UFGTime Sensitive to Hyperparameters?*** Based on the results shown in Figure 6, the MAE of our model remains relatively stable, around 0.055, across a range of hyperparameter values, indicating low sensitivity. ***What Are the Patterns of Hyperparameters?*** From Figure 6, we observe that hidden size has a more pronounced effect on the model. Specifically, increasing the number of hidden units sharpens the contour surface. Furthermore, as hidden size increases, the optimal values of k and s tend to decrease, suggesting that the model dynamically balances the dilation scale, number of neighbors, and hidden size to maintain computational complexity.

[14]https://github.com/ultralytics/thop

4.5 Ablation Analysis

We conduct ablation experiments along four dimensions: sparsity of the graph (SG), framelet graph convolution (FrC), convolutions on the spectral-variate graph (GC), and frequency transformation (FT). Each setting is evaluated over 10 repeated runs, and we apply two-way ANOVA to assess statistical significance. As shown in Figure 7, replacing the sparse graph with a fully connected one degrades performance, highlighting the benefit of enforcing sparsity in temporal graph construction. Substituting the framelet convolution with a standard GCN [20] results in increased error, confirming the effectiveness of multi-scale frequency-aware operations. Removing graph convolutions altogether and replacing them with linear layers results in a significant drop in accuracy, indicating the necessity of graph-based modeling for capturing temporal dependencies. Finally, comparing the spectral-variate graph with a hypervariate graph shows that incorporating frequency-domain transformations (e.g., DFT) enables better generalization, making the spectral-variate design more suitable for pure graph-based time series forecasting. Moreover, we observe that different designs yield comparable performance on the ECG dataset (see Figure 7 (b)), as its temporal patterns are relatively easy to capture.

5 CONCLUSION

In this work, we proposed UFGTime, a novel framework for multivariate time series forecasting that effectively addresses the intertwined inter- and intra-series dependencies, a challenge often overlooked by traditional temporal models. Leveraging spectral signals, UFGTime constructs a sparse spectral-variate graph that embeds temporal dependencies into a compact spectral representation. The framework further incorporates a global framelet message-passing function, which efficiently aggregates global signal information, mitigates over-smoothing, and achieves near-linear computational complexity of $O(kNT)$. Extensive experiments demonstrate that UFGTime consistently outperforms state-of-the-art baselines, underscoring the practicality and robustness and establishing it as a promising pure graph solution for the multivariate time series forecasting task.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[2] Roy Assaf, Ioana Giurgiu, Frank Bagehorn, and Anika Schumann. 2019. MTEX-CNN: Multivariate time series explanations for predictions with convolutional neural networks. In *IEEE International Conference on Data Mining*. IEEE, Beijing, China, 952–957.

[3] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems* 33 (2020), 17804–17815.

[4] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. 2017. Conditional time series forecasting with convolutional neural networks. arXiv:1703.04691

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

[6] Hongjie Chen and Hoda Eldardiry. 2024. Graph time-series modeling in deep learning: a survey. *ACM Transactions on Knowledge Discovery from Data* 18, 5 (2024), 1–35.

[7] Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. 2023. Gread: Graph neural reaction-diffusion networks. In *International Conference on Machine Learning*. PMLR, Hawaii,USA, 5722–5747.

[8] Andrea Cini, Danilo Mandic, and Cesare Alippi. 2024. Graph-based Time Series Clustering for End-to-End Hierarchical Forecasting. In *International Conference on Machine Learning*, Vol. 235. PMLR, Vienna, Austria, 8985–8999.

[9] Jerome T Connor, R Douglas Martin, and Les E Atlas. 1994. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks* 5, 2 (1994), 240–254.

[10] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K Mathur, Rajat Sen, and Rose Yu. 2023. Long-term Forecasting with TiDE: Time-series Dense Encoder. *Transactions on Machine Learning Research* -, - (2023).

[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems* 29 (2016).

[12] Francesco Di Giovanni, James Rowbottom, Benjamin P Chamberlain, Thomas Markovich, and Michael M Bronstein. 2022. Graph Neural Networks as Gradient Flows. *arXiv preprint arXiv:2206.10991* (2022).

[13] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-temporal graph ode networks for traffic flow forecasting. In *ACM Conference on Knowledge Discovery & Data Mining*. ACM, Singapore, 364–373.

[14] Andi Han, Dai Shi, Lequan Lin, and Junbin Gao. 2024. From Continuous Dynamics to Graph Neural Networks: Neural Diffusion and Beyond. *Transactions on Machine Learning Research* (2024).

[15] Andi Han, Dai Shi, Zhiqi Shao, and Junbin Gao. 2022. Generalized energy and gradient flow via graph framelets. *ArXiv:2210.04124* (2022).

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[17] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I Webb, Irwin King, and Shirui Pan. 2024. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[18] Ming Jin, Yu Zheng, Yuan-Fang Li, Siheng Chen, Bin Yang, and Shirui Pan. 2022. Multivariate time series forecasting with dynamic graph neural odes. *IEEE Transactions on Knowledge and Data Engineering* 35, 9 (2022), 9168–9180.

[19] Minjung Kim, Yusuke Hioka, and Michael Witbrock. 2024. Neural Fourier Modelling: A Highly Compact Approach to Time-Series Analysis. arXiv:2410.04703

[20] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[21] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

[22] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*.

[23] Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. 2024. Largest: A benchmark dataset for large-scale traffic forecasting. *Advances in Neural Information Processing Systems* 36 (2024).

[24] Xinliang Liu, Bingxin Zhou, Chutian Zhang, and Yu Guang Wang. 2023. Framelet message passing. *ArXiv:2302.14806* (2023).

[25] Yijing Liu, Qinxian Liu, Jian-Wei Zhang, Haozhe Feng, Zhongwei Wang, Zihan Zhou, and Wei Chen. 2022. Multivariate time-series forecasting with temporal polynomial graph neural networks. *Advances in Neural Information Processing Systems* 35 (2022), 19414–19426.

[26] John A Miller, Mohammed Aldosari, Farah Saeed, Nasid Habib Barna, Subas Rana, I Budak Arpinar, and Ninghao Liu. 2024. A survey of deep learning and foundation models for time series forecasting. arXiv:2401.13912

[27] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1ecqn4YwB

[28] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *ACM International Conference on Knowledge Discovery & Data Mining*. 1720–1730.

[29] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S Jensen, Zhenli Sheng, et al. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. *Proceedings of the VLDB Endowment* 17, 9 (2024), 2363–2377.

[30] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. 2023. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993* (2023).

[31] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete Graph Structure Learning for Forecasting Multiple Time Series. In *International Conference on Learning Representations*.

[32] Zhiqi Shao, Dai Shi, Andi Han, Yi Guo, Qibin Zhao, and Junbin Gao. 2023. Unifying over-smoothing and over-squashing in graph neural networks: A physics informed approach and beyond. *ArXiv:2309.02769* (2023).

[33] Zezhi Shao, Fei Wang, Yongjun Xu, Wei Wei, Chengqing Yu, Zhao Zhang, Di Yao, Guangyin Jin, Xin Cao, Gao Cong, et al. 2023. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. arXiv:2310.06119

[34] Dai Shi, Yi Guo, Zhiqi Shao, and Junbin Gao. 2023. How curvature enhance the adaptation power of framelet gcns. *ArXiv:2307.09768* (2023).

[35] Kamile Stankeviciute, Ahmed M Alaa, and Mihaela van der Schaar. 2021. Conformal time-series forecasting. *Advances in neural information processing systems* 34 (2021), 6216–6228.

[36] Matthew Thorpe, Tan Minh Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. 2022. GRAND++: Graph Neural Diffusion with A Source Term. In *International Conference on Learning Representations*.

[37] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *ACM International World Wide Web Conference*. 1082–1092.

[38] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. CoST: Contrastive Learning of Disentangled Seasonal-Trend Representations for Time Series Forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=PilZY3omXV2

[39] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.

[40] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *ACM International Conference on Knowledge Discovery & Data Mining*. 753–763.

[41] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *International Joint Conference on Artificial Intelligence*. 1907–1913.

[42] Yutong Xia, Yuxuan Liang, Haomin Wen, Xu Liu, Kun Wang, Zhengyang Zhou, and Roger Zimmermann. 2024. Deciphering spatio-temporal graph forecasting: A causal lens and treatment. *Advances in Neural Information Processing Systems* 36 (2024).

[43] Ling Yang and Shenda Hong. 2022. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *International conference on machine learning*. PMLR, 25038–25054.

[44] Mengxi Yang, Xuebin Zheng, Jie Yin, and Junbin Gao. 2022. Quasi-Framelets: Another Improvement to Graph Neural Networks. *arXiv:2201.04728* (2022).

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

[46] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *The Joint Conference on Artificial Intelligence*.

[47] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. 2020. Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction. In *European Conference on Computer Vision*. 507–523.

[48] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *AAAI Conference on Artificial Intelligence*, Vol. 37. 11121–11128.

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

Conference on Learning Representations.

[50] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *AAAI Conference on Artificial Intelligence*, Vol. 34. 1234–1241.

[51] Xuebin Zheng, Bingxin Zhou, Junbin Gao, Yu Guang Wang, Pietro Lió, Ming Li, and Guido Montúfar. 2021. How framelets enhance graph neural networks. *In International Conference of Machine Learning* (2021).

[52] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI Conference on Artificial Intelligence*, Vol. 35. 11106–11115.