ForecastGrapher: Redefining Multivariate Time Series Forecasting with Graph Neural Networks

Wanlin Cai[1], **Kun Wang**[2], **Hao Wu**[2], **Xiaoxu Chen**[3], **Yuankai Wu**[1] *
[1]Sichuan University
[2]University of Science and Technology of China
[3]McGill University
caiwanlin@stu.scu.edu.cn, {wk520529,wuhao2022}@mail.ustc.edu.cn,
xiaoxu.chen@mail.mcgill.ca, wuyk0@scu.edu.cn

Abstract

The challenge of effectively learning inter-series correlations for multivariate time series forecasting remains a substantial and unresolved problem. Traditional deep learning models, which are largely dependent on the Transformer paradigm for modeling long sequences, often fail to integrate information from multiple time series into a coherent and universally applicable model. To bridge this gap, our paper presents ForecastGrapher, a framework reconceptualizes multivariate time series forecasting as a node regression task, providing a unique avenue for capturing the intricate temporal dynamics and inter-series correlations. Our approach is underpinned by three pivotal steps: firstly, generating custom node embeddings to reflect the temporal variations within each series; secondly, constructing an adaptive adjacency matrix to encode the inter-series correlations; and thirdly, augmenting the GNNs' expressive power by diversifying the node feature distribution. To enhance this expressive power, we introduce the Group Feature Convolution GNN (GFC-GNN). This model employs a learnable scaler to segment node features into multiple groups and applies one-dimensional convolutions with different kernel lengths to each group prior to the aggregation phase. Consequently, the GFC-GNN method enriches the diversity of node feature distribution in a fully end-to-end fashion. Through extensive experiments and ablation studies, we show that ForecastGrapher surpasses strong baselines and leading published techniques in the domain of multivariate time series forecasting.

1 Introduction

Multivariate time series forecasting is a critical component in predictive analytics, aiming to predict future values of interconnected time series based on their historical trends. Over the past decade, this intricate problem has been intensively tackled using various statistical and machine learning methods [19]. Recently, Various deep learning models, including Transformer-based [61] and non-attention mechanisms like MLP [58, 56, 12] and TCNs [48], have been proposed to address the challenges of time series forecasting, demonstrating competitive performance.

Different structures process time series in various ways. Essentially, they all utilize neural networks to capture both inter-series and intra-series correlations (temporal correlations) in time series data [6]. Earlier works often overlooked the inter-series correlation, treating all variables at the same time point as a single token. They employed Transformers [61, 49, 62], MLPs [58, 10, 56], and TCNs [47, 57] to capture the temporal correlation across these tokens. However, the importance of inter-series correlation is equally significant.

*Corresponding author

Preprint. Under review.

Recently, several studies [59, 30, 8] have been exploring the effectiveness of Transformers in modeling inter-series correlations, rather than focusing solely on temporal correlations. For example, iTransformer [30] reconceptualizes individual time series as distinct tokens and employs self-attention to capture inter-series correlations between these tokens. This approach demonstrates that Transformers are more adept at modeling inter-series correlations than temporal correlations.

The use of Transformers to model inter-series correlations is actually quite similar to graph structure learning in GNNs. This is because the attention mechanisms in Transformers, resembling GNNs' neighborhood aggregation, can be seen as employing dynamic adjacency matrices [21, 43]. This similarity highlights the intriguing parallels and potential for applying GNNs in areas traditionally dominated by Transformers, especially in multivariate time series analysis. At this point, we ask the following important question - *Can GNNs models yield superior performance for multivariate time series forecasting, and if so, what adaptations are necessary for multivariate time series data?*

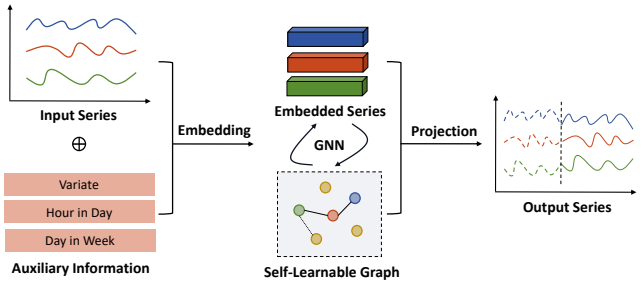


Figure 1: In ForecastGrapher, each variate is treated as a node within a graph, transforming the multivariate time series forecasting problem into a node regression task.

Towards this, we introduce **ForecastGrapher**, a GNN architecture with strong expressive power tailored for precise multivariate time series forecasting. Figure 1 illustrates the ForecastGrapher's approach to multivariate time series forecasting. ForecastGrapher conceptualizes each input time series as a graph node. Initially, it employs embedding techniques to encode the temporal variations of individual time series into a high-dimensional space. Subsequently, ForecastGrapher features self-learning graph structures to discern inter-series correlations among nodes. Ultimately, the forecasting results are generated by the form of node regression task after several layers of feature aggregation.

In addition to restructuring the forecasting problem as a node regression task, we also focus on the expressive power issue of GNNs. Notably, although GNNs are widely used in spatio-temporal forecasting (which can be understood as a special type of multivariate time series forecasting problem, mainly focusing on short-term predictions, such as 12-step ahead prediction) [20], the inherent limitations in their expressive power remain largely unaddressed in this field. It is well-known that typical GNNs, including Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), exhibit limited expressiveness. Several studies [53, 33] have shown that GNNs cannot distinguish between certain pairs of graphs. For instance, typical GCNs using mean value aggregators are unable to differentiate between node feature distributions with the same mean value but different standard deviations [9, 5]. This is particularly problematic for multivariate time series forecasting, as differing variances in the historical variables can lead to completely different future outcomes. To address this issue, we introduce a new GNN structure called Group Feature Convolution GNN (GFC-GNN) for ForecastGrapher. We employ learnable scalers to divide features into groups. Within these groups, we perform 1D convolutions across the feature dimension, using kernels of various lengths for differently scaled groups. As a result, GFC-GNN diversifies the distribution of node features in an end-to-end manner.

To summarize, the key contributions of our ForecastGrapher are outlined as follows:

- Framework: We discovered that a GNN architecture designed for the node regression task can effectively address the challenges of multivariate time series forecasting. The key lies in how to learn the graph structure, and in the design of node embeddings and the GNN framework itself.

- Expressive Power: Our findings indicate that the application of 1D convolutional layers, with varying kernel lengths to the feature dimensions prior to node aggregation, can effectively diversify the distribution of node features. This enhancement in diversity significantly improves forecasting accuracy.

- Performance: ForecastGrapher delivers performance that is comparable to or surpasses state-of-the-art methods across twelve benchmark datasets for multivariate time series forecasting.

2 Related Work

2.1 Backbone Networks Used in Time Series Forecasting

The dominant architecture for time series forecasting has traditionally been the Recurrent Neural Network (RNN) [39, 38, 46], the representative work is the introduction of DeepAR [39]. Another early representative network for time series forecasting is the Convolutional Neural Network (CNN), with the earliest work being Temporal Convolutional Network (TCN) [4], a type of CNN that does not leak future values. CNNs have been successfully utilized in time series forecasting, with notable works including SCInet [29] and TimesNet [48]. Since 2019, the Transformer has been introduced for multivariate time series forecasting, specifically for long sequence modeling [26, 61]. Subsequently, numerous Transformer variants have been developed to enhance performance in visual tasks. Key advancements include channel dependence [59, 30], time series patches [34], and the incorporation of frequency domain information [62, 49]. MLPs have also been explored in the context of time series forecasting [58]. With specially designed modules [10, 56, 12, 54], MLP can achieve competitive performance. However, most of those methods have not explicitly modeled the inter-series correlation between different variables. Some channel-dependent methods use the attention mechanism to capture relationships between variables, but the correlations obtained often change with time series fluctuations [59, 30], lacking interpretability.

2.2 Graph Neural Networks

The earliest graph neural networks were initially outlined in [15, 40]. In recent years, a variety of GNN variants have been introduced [24, 2, 35, 14, 44, 1, 9]. GNNs are typically applied to data with graph structures, such as social networks [17], citation networks [41] and biochemical graphs [45]. Despite their empirical successes in these fields, [53] and [33] demonstrated that GNNs cannot distinguish some pairs of graphs. To address this limitation, several studies have utilized hand-crafted aggregators to enhance the expressive power of GNNs [9, 11, 32].

The applications of GNNs in the field of spatio-temporal forecasting involve using GNNs to model spatial attributes, followed by the use of other modules to model temporal attributes. This approach has led to the development of a new Spatio-Temporal GNN (STGNN) structure [20]. For example, models such as DCRNN [27], ST-MetaNet [37], and AGCRN [3] combine GNNs with recurrent neural networks for their operations. Similarly, Graph WaveNet [51], MTGNN [50], and StemGNN [7] incorporate CNNs for temporal modeling. Additionally, the attention mechanism has become a widely used technique in STGNNs [16, 60]. However, these methods only address forecasting problems where both the input and output sequences are short, and they give little consideration to the expressive power of GNNs. Yi et al. [55] provided a purely GNN-based perspective on multivariate time series forecasting, treating both time points and variates as nodes within a graph. However, their approach is primarily suited to short-term forecasting; for long-term forecasting, treating time points as nodes becomes impractical due to computational complexity.

3 Methodology

3.1 Problem Formulation

In multivariate time series forecasting, given historical observations $\mathbf{X}_t = \{\mathbf{x}_{t-h}, \ldots, \mathbf{x}_{t-1}\} \in \mathbb{R}^{N \times h}$ with h time steps and N variates, we predict the future S time steps $\mathbf{Y}_t = \{\mathbf{x}_t, \ldots, \mathbf{x}_{t+S-1}\} \in \mathbb{R}^{N \times S}$. For convenience, we denote $\mathbf{x}_t \in \mathbb{R}^N$ as the time series data collected at time point t. Furthermore, we denote $\mathbf{X}_{t,n} \in \mathbb{R}^h$ as the complete time series of the variate indexed by n, collected from time point $t-h$ to $t-1$.

To generate $\mathbf{Y}_t$, we conceptualize the generation process as a node regression problem within the context of graph data. Specifically, each $\mathbf{X}_{t,n}$ in the input $\mathbf{X}_t$ is treated as a dynamic feature of node n in a graph. We assume the existence of additional auxiliary node features $\mathbf{E}_{t,n}$. Consequently, the

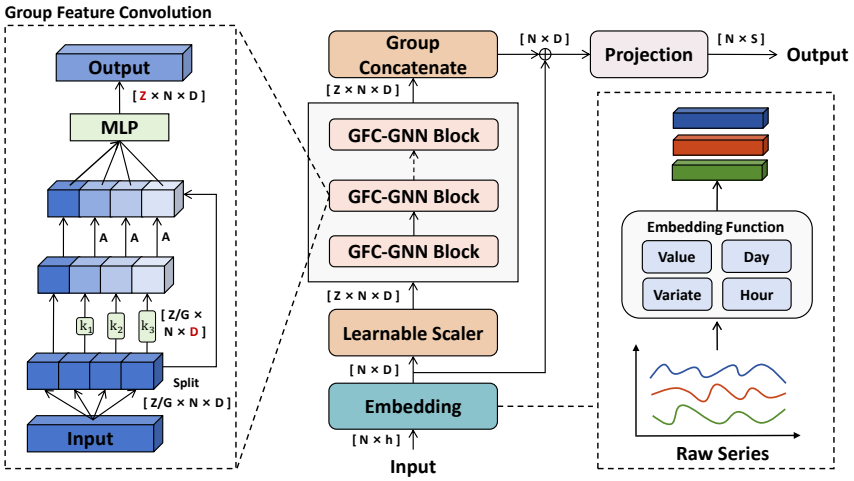


Figure 2: The overall structure of ForecastGrapher is designed to address a node regression task. The model considers each time series as a node and generates a corresponding node embedding. Next, it employs learnable scalers to partition the node embedding into multiple groups. Subsequently, several layers of GFC-GNN are stacked (the red color indicates the dimension to which the corresponding neural networks are applied). Finally, ForecastGrapher utilizes node projection for forecasting.

multivariate time series forecasting problem can be concisely formulated as follows:

$$\begin{aligned} \mathbf{h}_{t,n}^{0} &= \text{Embedding}\left(\mathbf{X}_{t,n}, \mathbf{E}_{t,n}\right), \quad n = 1, \ldots, N, \\ \mathbf{H}_{t}^{l+1} &= \text{GNN}\left(\mathbf{H}_{t}^{l}, \mathbf{A}^{l}\right), \quad l = 1, \ldots, L, \\ \mathbf{Y}_{t,n} &= \text{Projection}\left(\mathbf{h}_{t,n}^{L}\right). \end{aligned} \tag{1}$$

Here, $\mathbf{H}_t = \{\mathbf{h}_{t,1}, \ldots, \mathbf{h}_{t,N}\} \in \mathbb{R}^{N \times D}$ represents N node embeddings, each of dimension D, where the subscript t denotes the time step and the superscript l refers to the layer index. The term $\mathbf{A}$ represents the self-learned adjacency matrix. The whole ForecastGrapher framework is illustrated in Figure 2. In subsequent sections, we provide a detailed introduction to the design principles of Embedding layer, GNN layer, and Projection layer.

3.2 Embedding the Time Series

In our study, we treat each single variate as a node within a graph. The initial step involves integrating the temporal dynamics of time series into our node embeddings. There are many architectures like Temporal Convolutional Networks (TCN) and Transformers for this purpose. Surprisingly, simpler linear models have demonstrated superior performance in capturing these temporal patterns [58]. Therefore, in this research, we utilize a straightforward linear model to create embeddings that accurately reflect the temporal changes in individual time series.

In various forecasting contexts, dynamic and static covariates that are known in advance play a significant role. Key among these are indicators related to "where" and "when." [42]. For instance, global covariates, common to all time series like time of day and day of the week, or specific ones such as a sensor's location, are crucial. In the datasets examined in our study, we have included three additional embeddings (variate, hour and day) to enrich the node embeddings. These embeddings are designed to capture and incorporate these essential covariate aspects effectively. In summary, the node embedding $\mathbf{h}_{t,n}^{0}$ is calculated by the following equation:

$$\mathbf{h}_{t,n}^{0} = \text{Linear}(\mathbf{X}_{t,n}) + \mathbf{e}_{n}^{\text{variate}} + \mathbf{e}_{\pi(t)}^{HiD} + \mathbf{e}_{\pi(t)}^{DiW}, \tag{2}$$

where Linear denotes a straightforward linear layer. The term $\mathbf{e}_{n}^{\text{variate}} \in \mathbb{R}^{D}$ represents a learnable embedding associated with the n-th variate, $\mathbf{e}_{\pi(t)}^{HiD} \in \mathbb{R}^{D}$ and $\mathbf{e}_{\pi(t)}^{DiW} \in \mathbb{R}^{D}$ are learnable embeddings for Hour in Day and Day in Week, respectively, π indicates the use of the corresponding granularity timestamp as the index for these temporal embedding matrices (In the experiments, if the data lacks "Hour in Day" and "Day in Week" information, we will not utilize these information).

3.3 Graph Neural Networks

3.3.1 Self-Learnable Adjacency Matrix

Defining an adjacency matrix for target time series is often challenging, and there are various methods to learn one from data. In our ForecastGrapher model, we employ the popular and straightforward method proposed by [51]. This involves two trainable parameters, $\mathbf{E}_1^l$ and $\mathbf{E}_2^l \in \mathbb{R}^{N\times c}$, representing the source and target nodes respectively. The adjacency matrix is computed as follows:

$$\mathbf{A}^l = \text{SoftMax}\left(\text{ReLU}\left(\mathbf{E}_1^l \left(\mathbf{E}_2^l\right)^T\right)\right), \tag{3}$$

using the ReLU activation function to prune weak connections and the SoftMax function to normalize the adjacency matrix of the graph. It's important to highlight that in ForecastGrapher, we learn a new adjacency matrix at each layer. This strategy is implemented with the aim of capturing varying inter-series correlations across different layers.

3.3.2 Learnable Scaler and Group Feature Convolution

Departing from traditional human-crafted aggregators and scalers [9], our approach to enhancing the expressive power of GNNs involves a pure end-to-end strategy. This is inspired by the Bayesian neural network perspective of Convolutional Neural Networks (CNNs), which suggests that CNNs can autonomously modify the distribution of features [52, 36]. This insight forms the basis of our intuition that an automatic adjustment of feature distributions can be achieved within CNNs. We visualize the GFC mechanism in Figure 3.

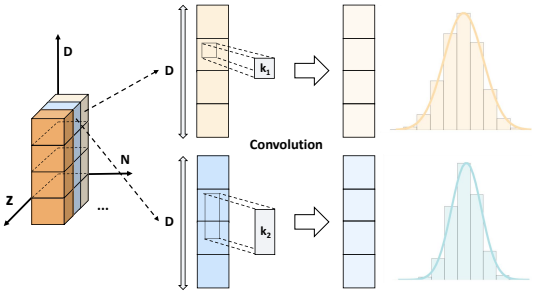


Figure 3: The GFC mechanism enhances the diversity of node embedding distributions: Convoluting the node feature with two distinct kernel lengths results in two distinct distributions.

Initially, we augment the initial embedding $\mathbf{H}_t^0$ by multiplying it with z learnable scalers. This process results in $\hat{\mathbf{H}}_t^0 \in \mathbb{R}^{z\times N\times D}$, achieved by introducing a new dimension. In each GNN layer, we partition $\hat{\mathbf{H}}_t$ into G groups, resulting in $\hat{\mathbf{H}}_{g,t} \in \mathbb{R}^{\frac{z}{G}\times N\times D}$, as illustrated in Figure 2. Here, g signifies the g-th group. For an integer division scenario, where $z \bmod G$ equals 0, we simply split into equal segments. Otherwise, we allocate int $\left(\frac{z}{G}\right) + z \bmod G$ dimensions to the first group, while maintaining int $\left(\frac{z}{G}\right)$ for the others. Subsequently, one-dimensional convolutions are applied on the feature dimension of $\hat{\mathbf{H}}_{g,t}$. To diversify the feature distribution of each group, different convolutional kernel sizes are used for each group, and one group is left unchanged without convolution. The formulation of GFC-GNN can be articulated as follows:

$$\begin{aligned}
\mathbf{T}_{g,t}^l &= \text{Conv1d}_g\left(\hat{\mathbf{H}}_{g,t}^l, k_g\right), \quad g = 2,\ldots,G,\\
\mathbf{V}_{g,t}^l &= \mathbf{A}^l\hat{\mathbf{T}}_{g,t}^l, \quad g = 2,\ldots,G,\\
\mathbf{H}_t^{l+1} &= \text{MLP}\left(\hat{\mathbf{H}}_{1,t}^l \mid \mathbf{V}_{2,t}^l \mid \ldots \mid \mathbf{V}_{G,t}^l\right),
\end{aligned} \tag{4}$$

where Conv1d_g represents the 1DCNN for the g-th group in the feature dimension, k_g is the kernel length of the g-th group and MLP denotes the multi-layer perceptron that is applied to the concatenated outputs of all groups. Notably, each 1DCNN is characterized by different kernel lengths, allowing for varied and specialized processing across different groups. We provide a theoretical analysis using Monte Carlo sampling to demonstrate that GFC-GNN can transform distributions with the same mean but different variances into distributions with different means, and it can generate diverse feature distributions, as shown in the Appendix A.

3.4 Combining Groups and Generating Forecast Results

After undergoing L layers of graph convolution, we obtain the grouped representation $\mathbf{H}_t^{L-1}$. To generate the forecast results via node regression, it's necessary to amalgamate the representations

of the G groups. To ensure that the representations of each group are fully utilized, we employ a learnable concatenation method. Additionally, we use a residual learning approach [18] to fuse the final representation with the initial representation. The finally representation is given by

$$\mathbf{H}_t^L = \mathbf{H}_t^0 + \mathbf{W}_{\text{concat}}\mathbf{H}_t^{L-1}, \tag{5}$$

where $\mathbf{W}_{\text{concat}} \in \mathbb{R}^{z\times 1}$ is the learnable weight matrix for group concatenation.

Finally, the regression layer performs forecasting based on

$$\mathbf{Y}_t = \mathbf{H}_t^L\mathbf{W}_{\text{reg}} + \mathbf{b}_{\text{reg}}, \tag{6}$$

where $\mathbf{W}_{\text{reg}} \in \mathbb{R}^{D\times S}$ and $\mathbf{b}_{\text{reg}} \in \mathbb{R}^S$ are the learnable weights and bias, respectively.

4 Experiments

4.1 Experimental Setup

Datasets We evaluate the performance of ForecastGrapher on twelve widely used datasets. These include ETT (h1, h2, m1, m2) [61], Electricity, Exchange [25], Traffic, Weather, and PEMS (03, 04, 07, 08), as evaluated in [30, 29]. Additional information about these datasets can be found in the Appendix B.1.

Baselines We have selected several well-known forecasting models as our benchmarks, including **(i) Transformer-based models:** iTransformer [30], PatchTST [34], Crossformer [59]; **(ii) MLP-based models:** DLinear [58], RLinear [28]; **(iii) TCN-based models:** TimesNet [48], SCINet [29]. **(iv) GNN-based models:** FourierGNN[55], StemGNN[7]. Additionally, we include a Naive method, which repeats the last 24 values in the review window.

Parameter Settings We maintained identical dataset partitioning and historical window length with $h = 96$ following [30]. The prediction window length S is set within $\{96, 192, 336, 720\}$ or $\{12, 24, 48, 96\}$. For the PEMS dataset, we selected $\{12, 24, 48, 96\}$ time steps as the long-term prediction length, in contrast to short-term traffic forecasts, e.g., $\{3, 6, 12\}$ in traditional studies [27]. The batch size is fixed at $batch = 32$, though it is reduced to 16 in cases of insufficient memory. We limit the number of training epochs to $epochs = 10$, using mean squared error (MSE) as the training loss function. Additional details on experimental parameters are provided in the Appendix B.2. For the hardware, we employ 4 RTX 4090 24GB GPUs for our experiments.

4.2 Forecasting Results

The main prediction results are shown in Table 1, and we compare the performance with the benchmarks using average MSE and MAE of all output lengths, which the lower, the better. The outcomes reveal that ForecastGrapher demonstrates outstanding performance across all datasets. Specifically, ForecastGrapher achieves the top spot in terms of MSE and MAE a total of 16 times. Compared with the recent SOTA iTransformer, the error of ETT, Electricity, Weather, PEMS datasets significantly decreased by 4.21%, 7.25%, 4.47% and 14.65% respectively. Specifically, ForecastGrapher has demonstrated superior performance compared to other methods, notably iTransformer, on high-dimensional datasets such as Electricity and PEMS. iTransformer leverages the Transformer model to capture inter-series correlations by essentially creating a dynamic graph structure. Our research, however, suggests that GNNs utilizing static graph structures are more effective at capturing these inter-series correlations. Contrary to what one might intuitively expect, dynamic inter-series correlations may not provide benefits for high-dimensional, long-term forecasting tasks. To enable an intuitive comparison, we selected representative models for visualization on the Electricity dataset. As illustrated in Figure 4, ForecastGraper

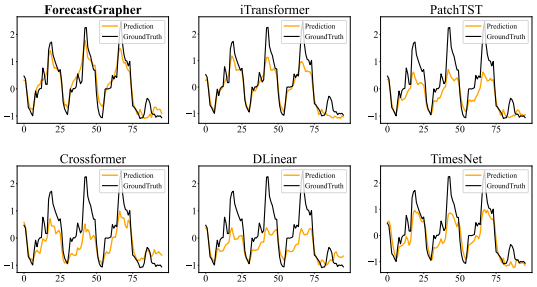


Figure 4: Visualization of input 96 and output 96 prediction results on the Electricity dataset.

exhibits a superior grasp of data fluctuations, surpassing other models in accuracy. When compared with other advanced GNNs and Naive method, ForecastGrapher also has a significant advantage in long-term prediction, as shown in Table 2.

Table 1: Multivariate time series prediction results, with input length 96, output lengths in {12,24,48,96} for PEMS, {96,192,336,720} for others. Results are averaged from all output lengths. Use bold to indicate the best, and underline to indicate the second. The benchmarks are reported from [30]. Full results are listed in Appendix G.

Model	**Ours**		iTransformer		PatchTST		Crossformer		DLinear		RLinear		TimesNet		SCINet	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.383**	**0.397**	0.407	0.410	0.387	0.400	0.513	0.496	0.403	0.407	0.414	0.408	0.400	0.406	0.485	0.481
ETTm2	**0.276**	**0.323**	0.288	0.332	0.281	0.326	0.757	0.610	0.350	0.401	0.286	0.327	0.291	0.333	0.571	0.537
ETTh1	**0.437**	0.437	0.454	0.447	0.469	0.454	0.529	0.522	0.456	0.452	0.446	**0.434**	0.458	0.450	0.747	0.647
ETTh2	**0.372**	0.402	0.383	0.407	0.387	0.407	0.942	0.684	0.559	0.515	0.374	**0.399**	0.414	0.427	0.954	0.723
Electricity	**0.165**	**0.260**	0.178	0.270	0.205	0.290	0.244	0.334	0.212	0.300	0.219	0.298	0.193	0.295	0.268	0.365
Exchange	0.367	0.407	0.360	**0.403**	0.367	0.404	0.940	0.707	**0.354**	0.414	0.378	0.417	0.416	0.443	0.750	0.626
Traffic	0.458	0.292	**0.428**	**0.282**	0.481	0.304	0.550	0.304	0.625	0.383	0.626	0.378	0.620	0.336	0.804	0.509
Weather	**0.246**	**0.274**	0.258	0.279	0.259	0.281	0.259	0.315	0.265	0.317	0.272	0.291	0.259	0.287	0.292	0.363
PEMS03	**0.098**	**0.205**	0.113	0.221	0.180	0.291	0.169	0.281	0.278	0.375	0.495	0.472	0.147	0.248	0.114	0.224
PEMS04	0.093	0.204	0.111	0.221	0.195	0.307	0.209	0.314	0.295	0.388	0.526	0.491	0.129	0.241	**0.092**	**0.202**
PEMS07	**0.079**	**0.172**	0.101	0.204	0.211	0.303	0.235	0.315	0.329	0.395	0.504	0.478	0.124	0.225	0.119	0.217
PEMS08	**0.140**	**0.212**	0.150	0.226	0.280	0.321	0.268	0.307	0.379	0.416	0.529	0.487	0.193	0.271	0.158	0.244
1^{st} Count	**16**		3		0		0		1		2		0		2	

Table 2: Comparison with GNN and Naive method for multivariate time series prediction with input length 96. Results are averaged from all output lengths. Full results are listed in Appendix G.

Dataset	Electricity		Traffic		Weather		PEMS03		PEMS04		PEMS07		PEMS08	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Ours	**0.165**	**0.260**	**0.458**	**0.292**	**0.246**	**0.274**	**0.098**	**0.205**	**0.093**	**0.204**	**0.079**	**0.172**	**0.140**	**0.212**
FourierGNN	0.228	0.324	0.557	0.342	0.249	0.302	0.151	0.267	0.180	0.294	0.123	0.237	0.216	0.312
StemGNN	0.197	0.300	0.612	0.356	0.268	0.321	0.187	0.302	0.217	0.333	0.184	0.289	0.303	0.351
Naive	0.329	0.341	1.156	0.477	0.371	0.336	0.901	0.703	0.966	0.735	0.966	0.720	0.997	0.745

4.3 Ablation on GNNs

We substitute our uniquely designed GNNs with simpler alternatives such as GCN [24], GAT [44], and Mixhop [1]. Details on how to implement other GNN variants are provided in the Appendix C. Additionally, we assess the impact of the GFC methodology on enhancing the performance of these GNN models by comparing their performance with and without the GFC component.

The results on the Exchange, Traffic and Weather datasets are given in Table 3. The results indicate that the GFC module typically enhances the performance of all GNN models, particularly with more complex datasets. For instance, on the Traffic dataset, removing the GFC module led to a 7.59% increase in error (from 0.461 to 0.496). In contrast, for simpler datasets like the Exchange, the differences with and without the GFC module are relatively minor. Interestingly, Mixhop aggregates multiple representations by performing multi-hop neighbor aggregation at each layer, which is akin to grouping using multi-hop neighbors at every layer. The performance does not fluctuate significantly after removing the GFC, suggesting that the grouping mechanism of GFC may not be fully compatible with Mixhop's approach of aggregating multi-hop neighbors. This discrepancy warrants further in-depth research in the future. In summary, all vari-

Table 3: Ablation Study on GNNs and GFC: The prediction results are averaged across all prediction lengths.

Dataset	Exchange		Traffic		Weather	
Metric	MSE	MAE	MSE	MAE	MSE	MAE
ForecastGrapher	0.367	0.407	0.458	0.292	0.246	0.274
GCN	0.376	0.412	**0.462**	**0.293**	**0.245**	**0.275**
GCN-w/o-GFC	**0.370**	**0.409**	0.487	0.328	0.253	0.279
GAT	**0.369**	**0.408**	**0.461**	**0.303**	**0.246**	**0.274**
GAT-w/o-GFC	0.375	0.410	0.496	0.340	0.260	0.283
Mixhop	**0.367**	**0.408**	**0.451**	0.294	**0.248**	**0.275**
Mixhop-w/o-GFC	0.372	0.411	0.451	**0.290**	0.249	0.276

ants of GNNs demonstrate competitive performance, indicating that using GNNs for node regression is highly suitable for addressing multivariate time series forecasting problems.

4.4 Ablation on Inter-series Correlation Learning Mechanism

Our model primarily relies on self-learnable variate embeddings and a self-learnable adjacency matrix to capture inter-series correlations. The PEMS datasets provide adjacency matrices associated with actual distances, offering us a valuable opportunity to validate the effectiveness of the self-learnable adjacency matrix. To evaluate the influence of self-learnable variate embeddings and self-learnable adjacency matrices in ForecastGrapher, we design two distinct variants:

1. 'w/o-variate' eliminates self-learnable variate embeddings of nodes.
2. 'w/o-adp' replaces the adaptive adjacency matrix with the original distance-based adjacency matrix.

The results on PEMS are presented in Table 4. We have found that both self-learnable variate embeddings and adaptive adjacency matrices are equally important. For short-term prediction tasks (prediction length = 12), removing these components does not significantly reduce the model's performance. However, when the prediction horizon is extended, removing these components results in a much worse performance. This also indicates that distance-based adjacency matrices are insufficient to fully leverage the inter-series correlations within the traffic dataset. In addition, a visual analysis of the adjacency matrix can be seen in the Appendix E.

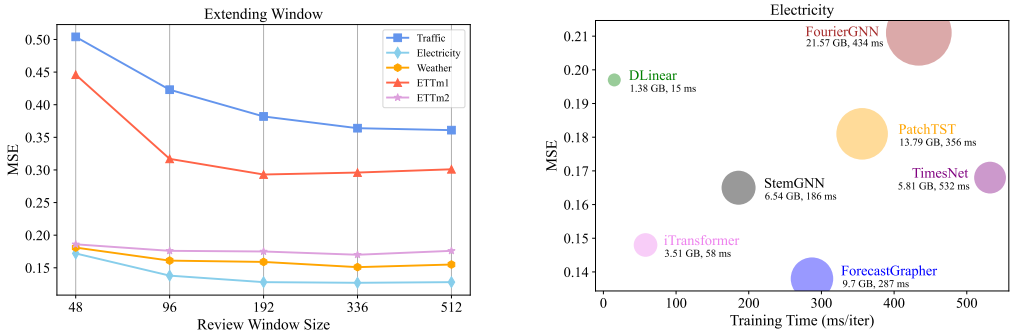
Table 4: Ablation on PEMS dataset, which includes ablation experiments of variable embedding and self-learning adjacency matrix.

Dataset		PEMS03		PEMS04		PEMS07		PEMS08	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
12	ForecastGrapher	**0.065**	**0.168**	**0.075**	**0.181**	**0.058**	**0.152**	**0.081**	**0.184**
	w/o-variate	0.069	0.174	0.079	0.187	0.063	0.161	0.083	0.187
	w/o-adp	0.068	0.173	0.078	0.185	0.063	0.159	0.086	0.191
96	ForecastGrapher	**0.134**	**0.244**	**0.112**	**0.227**	**0.103**	**0.194**	**0.197**	**0.234**
	w/o-variate	0.158	0.269	0.132	0.250	0.119	0.220	0.221	0.265
	w/o-adp	0.152	0.257	0.137	0.250	0.147	0.238	0.213	0.253

4.5 Extending the Historical Review Window and Model Efficiency

Previous research [58] has shown that a Transformer model may not necessarily be able to effectively extract information from longer review windows. It is natural to inquire whether the GNN-based ForecastGrapher can still capture sufficient temporal correlations from a longer review window. A robust model should exhibit improved performance as the review window extends, rather than displaying significant fluctuations.

To evaluate whether ForecastGrapher can leverage extended historical review windows, we conduct experiments on ETT, Weather, Traffic and Electricity datasets. The input length is varied from shorter to longer as $\{48, 96, 192, 336, 512\}$, and the model is assigned the task of forecasting the values for the next 96 time steps. Figure 5a illustrates that ForecastGrapher effectively captures temporal correlations from these extended review windows, yielding superior results as the review windows lengthen, with minimal fluctuations toward optimal performance. This demonstrates that the node regression framework provided by ForecastGrapher remains robust when dealing with multivariate time series containing long sequence inputs. Additionally, we conduct a comprehensive comparison of the performance, training speed, and memory usage of ForecastGrapher and other models on Electricity, as shown in Figure 5b. Although ForecastGrapher does not achieve the best results in terms of training speed and memory usage, it still outperforms models like PatchTST and FourierGNN. While TimesNet has relatively low memory usage, its training speed is the slowest. Overall, our model achieves the best performance at an acceptable cost.



(a) Forecasting results with output length 96 and input length in {48,96,192,336,512}.

(b) Model efficiency comparison on Electricity with input length 96 and output length 96.

Figure 5: Analysis of the model robustness and efficiency.

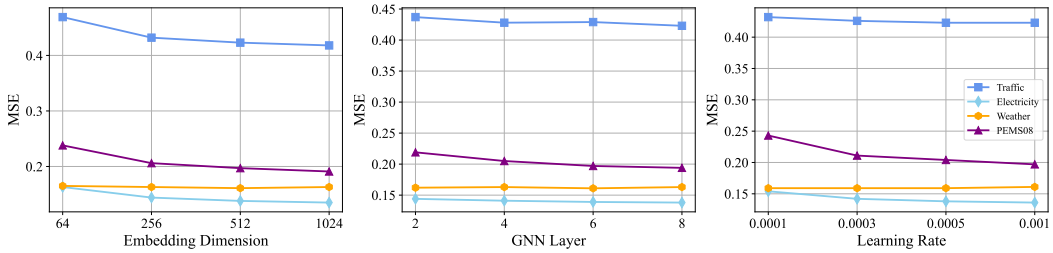


Figure 6: Sensitivity analysis in embedding dimension, number of GNN layers and learning rate. The results are recorded with input length 96 and output length 96.

4.6 Hyperparameter Sensitivity

We evaluate the hyperparameter sensitivity of ForecastGrapher, focusing specifically on three factors: the embedding dimension D, the number of GNN layers $Layer$, and the learning rate LR. The results are presented in Figure 6. We observe that for datasets with large variables such as Traffic and Electricity, the error decreases as D, $Layer$, and LR increase. The Weather dataset exhibits less sensitivity to these hyperparameters, maintaining stable performance across varying configurations.

5 Limitations

Our work has some limitations, as we have only focused on time series forecasting tasks within this framework. Research on other time series tasks, such as time series classification and anomaly detection, has not been conducted. Although ForecastGrapher outperforms models like iTransformer and DLinear in terms of prediction accuracy, its computational cost is higher. The main issue may lie in the self-learning graph structure, which still has room for optimization.

6 Conclusion

In this study, we take the innovative approach of framing multivariate time series forecasting as a node regression task within graph data, employing GNNs to address this challenge. We treat each variate in the dataset as a node, forming a graph that effectively captures the 'where' and 'when' information of the target time series. However, applying GNNs directly to this graph structure encounters limitations in expressive power. To overcome this, we introduce learnable scalars and 1D convolutions on the feature dimensions within each node to enhance information diversity. Leveraging this node regression framework and an enhanced GNN block, we develop the ForecastGrapher architecture. Through extensive testing across twelve datasets, the superiority of ForecastGrapher has been clearly demonstrated. We envision this pioneering effort as a foundational architecture for a broad range of time series analysis tasks.

References

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

[2] James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[3] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33:17804–17815, 2020.

[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[5] Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Mm-gnn: Mix-moment graph neural network towards modeling neighborhood feature distribution. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 132–140, 2023.

[6] Wanlin Cai, Yuxuan Liang, Xianggen Liu, Jianshuai Feng, and Yuankai Wu. Msgnet: Learning multi-scale inter-series correlations for multivariate time series forecasting. *arXiv preprint arXiv:2401.00423*, 2023.

[7] 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.

[8] Jinguo Cheng, Ke Li, Yuxuan Liang, Lijun Sun, Junchi Yan, and Yuankai Wu. Rethinking urban mobility prediction: A super-multivariate time series forecasting approach. *arXiv preprint arXiv:2312.01699*, 2023.

[9] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

[10] Abhimanyu Das, Weihao Kong, Andrew Leach, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023.

[11] Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. *Advances in Neural Information Processing Systems*, 32, 2019.

[12] Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 459–469, New York, NY, USA, 2023. Association for Computing Machinery.

[13] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[15] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[16] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 922–929, 2019.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[20] Guangyin Jin, Yuxuan Liang, Yuchen Fang, Zezhi Shao, Jincai Huang, Junbo Zhang, and Yu Zheng. Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[21] Chaitanya Joshi. Transformers are graph neural networks. *The Gradient*, 7, 2020.

[22] 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.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[25] 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.

[26] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[27] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[28] Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. Revisiting long-term time series forecasting: An investigation on linear mapping. *ArXiv*, abs/2305.10721, 2023.

[29] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.

[30] 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.

[31] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

[32] Xiaojun Ma, Qin Chen, Yuanyi Ren, Guojie Song, and Liang Wang. Meta-weight graph neural network: Push the limits beyond global homophily. In *Proceedings of the ACM Web Conference 2022*, pages 1270–1280, 2022.

[33] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[34] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[35] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR, 2016.

[36] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2018.

[37] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1720–1730, 2019.

[38] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

[39] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[40] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[41] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[42] Zezhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4454–4458, 2022.

[43] Petar Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster.

[45] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375, 2008.

[46] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *International conference on machine learning*, pages 6607–6617. PMLR, 2019.

[47] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International Conference on Learning Representations*, 2021.

[48] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

[49] 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.

[50] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020.

[51] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019.

[52] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pages 5393–5402. PMLR, 2018.

[53] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

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

[55] 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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[56] 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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[57] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987, 2022.

[58] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[59] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023.

[60] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1234–1241, 2020.

[61] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[62] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pages 27268–27286. PMLR, 2022.

A Deep Analysis of GFC-GNN

We first analyze the limitation of conventional GCNs following [5]. Our assumption is that there are two node classes on the graph, which exhibit significant differences in the distribution of their future values. However, their node features share similarities in distribution, such as having equal means. For a standard GCN using weighted mean aggregator [24], we identify the following limitations.

Theorem 1. *Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we denote the nodes belonging to class C_i as $\{v_i \mid v_i \in C_i\}$. Assume the feature distribution $\mathbf{h}_i$ of nodes in C_i follows an i.i.d. Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$. For any two distinct classes C_i and C_j, if $\mu_i = \mu_j$ and $\sigma_i \neq \sigma_j$, then the p-norm distance between the expectation of GCN outputs of these two classes is zero: $\left\|\mathbb{E}_{v_k \sim C_i}\left(GCN\left(\mathbf{h}_k\right)\right) - \mathbb{E}_{v_k \sim C_j}\left(GCN\left(\mathbf{h}_k\right)\right)\right\|_p = 0$.*

Proof. To simplify the analysis, we omit the activation function and learnable weights. The update function of a GCN with a mean aggregator is:

$$\mathbf{h}_k^{(l+1)} = \frac{1}{d_k} \sum_{n \in \mathcal{N}(v_k)} a_{nk} \mathbf{h}_n^{(l)}, \tag{7}$$

where d_k is the degree of node k, defined as $d_k = \sum_{n \in \mathcal{N}(v_k)} a_{nk}$. Here, a_{nk} represents the element of the adjacency matrix $\mathbf{A}$. Therefore, we have:

$$\mathbb{E}_{v_k \sim C_i}\left(\mathbf{h}_k^{(l+1)}\right) = \mathbb{E}_{v_k \sim C_j}\left(\mathbf{h}_k^{(l+1)}\right). \tag{8}$$

□

Now we analyze the shift in nodes' feature distribution of GFC. We focus on a simplified scenario in which the 1D CNN employs circularly-padded activations. Additionally, both the weights of this CNN and the nodes' features prior to convolution are assumed to be independently and identically distributed (i.i.d.) and drawn from a Gaussian distribution (similar to [52, 36]).

Let $\mathbf{h}^l(\alpha)$ denote the output at layer l and spatial location α. Assume $\mathbf{h}^l(\alpha)$ is independently and identically distributed (i.i.d.) from the Gaussian distribution $\mathcal{N}\left(\mu, \sigma^2\right)$. For the j-th value in the given group g, we scale $\mathbf{h}^l(\alpha)$ by a factor of s_j to obtain $\mathbf{h}_j^l(\alpha)$. Consequently, $\mathbf{h}_j^l(\alpha)$ follows a distribution of $\mathcal{N}\left(s_j\mu, s_j^2\sigma^2\right)$. Consider a 1D periodic CNN with a filter size of k_g, a channel size of $\frac{z}{G}$, and a spatial size of D (where convolution is performed over the feature dimension of size D). Assume the weights $\mathbf{W}_g^l \in \mathbb{R}^{k_g \times \frac{z}{G} \times \frac{z}{G}}$ are i.i.d. from $\mathcal{N}\left(\mu_w, \sigma_w^2\right)$, and we have ACT as the activation function. The forward-propagation dynamics are described by the following recurrence relation:

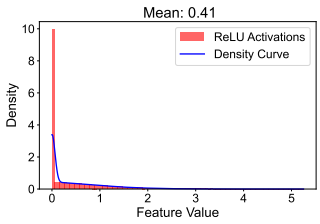
$$\mathbf{h}_i^{l+1}(\alpha) = \text{ACT}\left(\sum_{j=1}^{\frac{z}{G}} \sum_{\beta=1}^{k_g} \mathbf{h}_j^l\left(\alpha + \beta\right) \mathbf{W}_g^l(\beta, i, j)\right). \tag{9}$$

Providing an analytical form for $\mathbf{h}_i^{l+1}(\alpha)$ is challenging, but its mean and distribution depend not only on the expected value μ and variance σ of the input distribution but also on the convolution kernel length k_g and the value of the learnable scaler s_j. We utilized the Monte Carlo method to analyze the distribution of input features following the distributions $\mathcal{N}(1, 1)$ and $\mathcal{N}(1, 3)$ under the influence of two different sets of kernel lengths k_g and learnable scalers. We set the activation function as ReLU [13] function. The results are illustrated in Figure 7. From the figure, it is evident that even if the distributions of $\mathbf{h}^l(\alpha)$ and the learned 1D CNN weights are identical, varying the values of k_g and s_j can lead to entirely distinct output distributions. Moreover, when distributions with the same mean but different variances undergo group feature convolution, the expected value of the feature also changes.

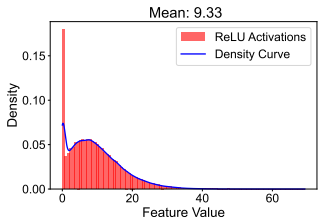
B Implementation Details

B.1 Datasets

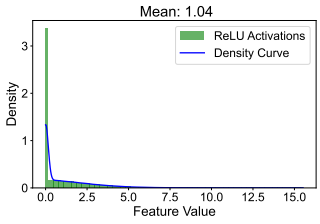
We utilized 12 datasets in our experiment, all of which are extensively employed for benchmark testing. The datasets encompass a diverse range of applications and scenarios, ensuring a comprehensive



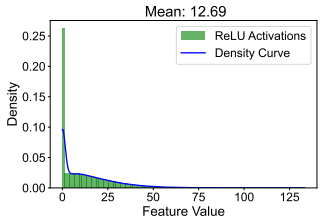
(a) Distribution shift for $\mathcal{N}(1,1)$ with $k_g = 2$, $\frac{z}{G} = 2$, $s_1 = -0.25$, $s_2 = 0.25$



(b) Distribution shift for $\mathcal{N}(1,1)$ with $k_g = 3$, $\frac{z}{G} = 2$, $s_1 = 2.0$, $s_2 = 2.5$



(c) Distribution shift for $\mathcal{N}(1,3)$ with $k_g = 2$, $\frac{z}{G} = 2$, $s_1 = -0.25$, $s_2 = 0.25$



(d) Distribution shift for $\mathcal{N}(1,3)$ with $k_g = 3$, $\frac{z}{G} = 2$, $s_1 = 2.0$, $s_2 = 2.5$

Figure 7: Monte Carlo simulations of the distribution and mean value of $\mathbf{h}_i^{l+1}(\alpha)$.

evaluation of our method. The following provides a detailed overview of each dataset: (1) **ETT** [61] collects 7 features data at two distinct time scales: hourly and every 15 minutes. These data are gathered from two regions, resulting in a total of four datasets: h1, h2, m1, and m2. (2) **Electricity**[2] records the hourly electricity consumption of 321 customers. (3) **Exchange**[25] records daily exchange rates for 8 countries from 1990 to 2016. (4) **Traffic**[3] collects the road occupancy rate measured by 862 sensors on San Francisco freeways every hour since January 2015. (5) **Weather**[4] gathers 21 meteorological indicators, including air temperature, with a ten-minute time granularity. (6) **PEMS** collects traffic flow data in California through multiple sensors and we use four datasets including 03, 04, 07, 08 used by iTransformer[30].

We set the input length to 96, the output lengths for the PEMS dataset are $\{12, 24, 48, 96\}$, and for others are $\{96, 192, 336, 720\}$. Table 5 presents the number of variate, prediction length, dataset partition size, and frequency information for each dataset, providing a overview of the datasets used in our experiment. This information is essential for understanding the scale and characteristics of the datasets.

B.2 Settings and Hyperparameters

All experiments are performed utilizing RTX 4090 24GB GPU devices, and the training process is refined through the Adam[23] optimizer, with MSE loss function. Regarding $batch$ size, 16 is chosen for the traffic dataset due to memory constraints, while 32 is maintained for all others. We set the embedding dimension D within the range of $\{128, 512, 1024\}$, the learning rate within $\{0.0001, 0.0005, 0.001\}$, and the number of GNN layers from 1 to 9. The scaling number z in the learnable scaler is selected from $\{8, 32\}$. We perform a grid search within these parameter ranges to find the optimal settings. Furthermore, we offer different standardization [22, 31] options for different datasets, and we forego standardization for PEMS. For the PEMS datasets, we do not perform normalization prior to embedding. However, for all other datasets, normalization is applied beforehand. We provide specific hyperparameters for different datasets in Table 6.

[2]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[3]http://pems.dot.ca.gov

[4]https://www.bgc-jena.mpg.de/wetter/

Table 5: Description of all datasets.

Datasets	Nodes	Prediction Length	Dataset Size	Frequency
ETTm1	7	{96, 192, 336, 720}	(34465, 11521, 11521)	15 minutes
ETTm2	7	{96, 192, 336, 720}	(34465, 11521, 11521)	15 minutes
ETTh1	7	{96, 192, 336, 720}	(8545, 2881, 2881)	Hourly
ETTh2	7	{96, 192, 336, 720}	(8545, 2881, 2881)	Hourly
Electricity	321	{96, 192, 336, 720}	(18317, 2633, 5261)	Hourly
Exchange	8	{96, 192, 336, 720}	(5120, 665, 1422)	Daily
Traffic	862	{96, 192, 336, 720}	(12185,1757,3590)	Hourly
Weather	21	{96, 192, 336, 720}	(36792, 5271, 10540)	10 minutes
PEMS03	358	{12, 24, 48, 96}	(15629, 5147, 5147)	5 minutes
PEMS04	307	{12, 24, 48, 96}	(10100, 3303, 3304)	5 minutes
PEMS07	883	{12, 24, 48, 96}	(16839, 5550, 5550)	5 minutes
PEMS08	170	{12, 24, 48, 96}	(10618, 3476, 3477)	5 minutes

Table 6: Hyperparameters of ForecastGrapher on different datasets.

Datasets	ETTm1	ETTm2	ETTh1	ETTh2	ECL	Exchange	Weather	Traffic	PEMS03	PEMS04	PEMS07	PEMS08
Epochs	10											
Batch	32							16	32			
Loss	MSE											
Learning Rate	1e-4	1e-4	1e-4	1e-4	5e-4	1e-4	1e-4	1e-3	1e-3	5e-4	1e-3	1e-3
Layer	1	1	2	1	8	1	6	8	6	9	6	6
D	512	512	128	512	512	128	512	512	512			
z	32	32	32	32	8	8	32	8	8			
c	10							1000	10			
k_g	3, 5, 7											
group	4											
Optimizer	Adam [23]											

C Other GNN Variants

For the standard GCN, we apply graph convolution to each group. The formula undergoes only minor changes compared to Equation (4):

$$\begin{aligned} \mathbf{T}_{g,t}^{l} &= \text{Conv1d}_g\left(\hat{\mathbf{H}}_{g,t}^{l}, k_g\right), \quad g = 2, \ldots, G, \\ \mathbf{V}_{g,t}^{l} &= \mathbf{A}^{l}\hat{\mathbf{T}}_{g,t}^{l}, \quad g = 1, \ldots, G, \\ \mathbf{H}_{t}^{l+1} &= \text{MLP}\left(\hat{\mathbf{V}}_{1,t}^{l} \mid \mathbf{V}_{2,t}^{l} \mid \ldots \mid \mathbf{V}_{G,t}^{l}\right). \end{aligned} \tag{10}$$

The only difference is that we do not retain a module that bypasses the graph convolution. If we do not use GFC, the model abandons the grouping mechanism and operates directly on the ungrouped $\mathbf{H}_t^l$. In this case, $\mathbf{H}_t^{l+1} = \mathbf{A}^l\mathbf{H}_t^l\mathbf{W}_{\text{GCN}}^l$. When using GAT and MixHop, we replace $\mathbf{V}_{g,t}^l = \mathbf{A}^l\hat{\mathbf{T}}_{g,t}^l$ with the corresponding networks.

D Ablation on Embedding

To assess the impact of different embedding components in ForecastGrapher, we develop three distinct variants:

1. 'w/o-variate' eliminates variate embeddings of nodes, removing information about 'where.'.
2. 'w/o-hid' omits hour in day embeddings, thus eliminating temporal information about 'when' within a single day.

3. 'w/o-diw' removes day in week embeddings, which eliminates information about the 'when' across a week.

The results on ETTm1, ETTm2 and ETTh2 (The ETT datasets include comprehensive calendar information) presented in Table 7 generally show that models achieve optimal performance when all embedding components are incorporated. However, it's important to note that not all datasets respond equally to various embedding strategies. For example, ETTm1 exhibits robustness even in the absence of variate and hid embeddings.

Table 7: Ablation on embedding. The results are obtained from the mean of all prediction lengths, and the best results are highlighted in bold.

Dataset	ETTm1		ETTm2		ETTh2	
Metric	MSE	MAE	MSE	MAE	MSE	MAE
ForecastGrapher	**0.383**	**0.397**	**0.276**	**0.323**	**0.372**	**0.402**
w/o-variate	0.384	0.397	0.280	0.325	0.377	0.403
w/o-hid	0.384	0.398	0.278	0.324	0.376	0.404
w/o-diw	0.395	0.408	0.278	0.324	0.384	0.407

E Learned Graph Visualization

To enhance the interpretability of our analysis, we visualize the learned partial adjacency matrix on the PEMS04 dataset in Figure 8. Specifically, we generate a heatmap to visualize the associations among the top 50 nodes in the dataset. This heatmap provides a quick overview of the relationships between these nodes. We also visualized the time series corresponding to node pairs with higher values in the learnable adjacency matrix. Moreover, we compare the learned adjacency matrix with the preset distance-based adjacency matrix. Firstly, the adjacency matrix learned by ForecastGrapher is sparser, indicating that the model requires fewer inter-series correlations for predictions. Furthermore, the learnable adjacency matrix captures connections between time series that are distant yet exhibit strong similarities, for example, nodes 0 and 36, 8 and 35, as well as 34 and 44, which are challenging to represent in distance-based static adjacency matrices.

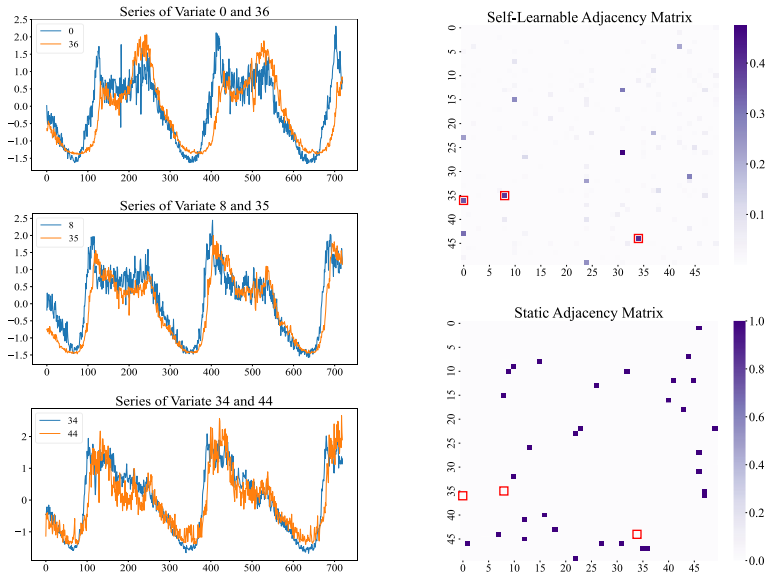


Figure 8: Visualization of the adjacency matrix for the top 50 nodes in the PEMS04 dataset, showcasing both the third-layer learnable adjacency matrix and the preset static adjacency matrix. The preset static adjacency matrix fails to capture the correlations between time series with strong similarities.

F More Forecasting Results Visualization

To facilitate a comprehensive comparison of model performances, we include additional prediction outcomes in Figure 9 to Figure 11. We utilize iTransformer[30], PatchTST[34], Crossformer[59], DLinear[58], TimesNet[48] as benchmarks for comparison. It becomes evident that our model excels in predicting future trends, thus demonstrating its superior performance.

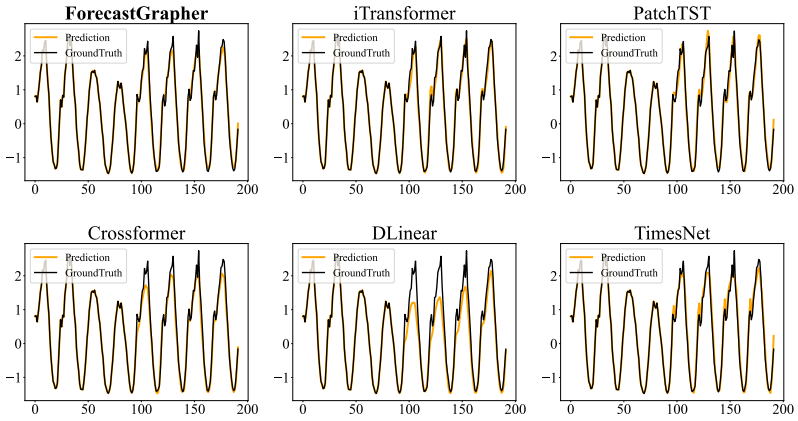


Figure 9: Visualization of input 96 and output 96 prediction results on the Traffic dataset.

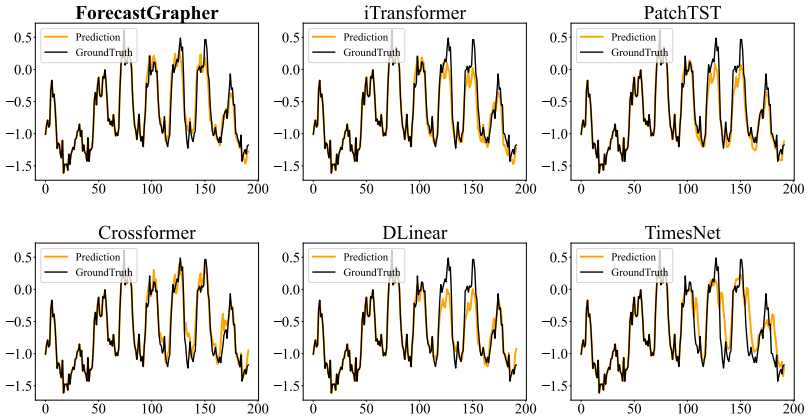


Figure 10: Visualization of input 96 and output 96 prediction results on the Electrcity dataset.

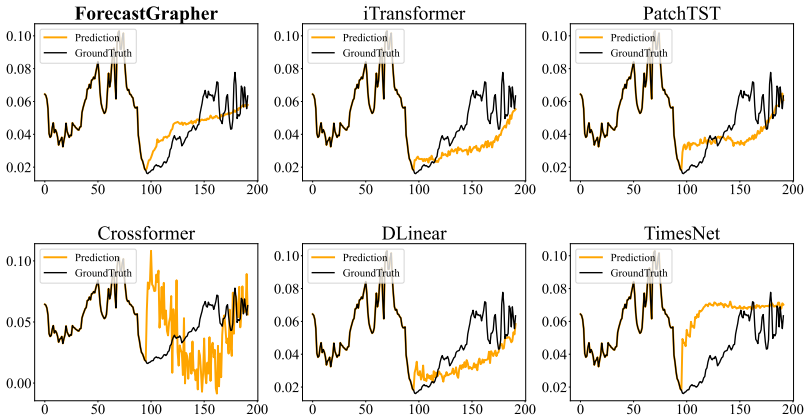


Figure 11: Visualization of input 96 and output 96 prediction results on the Weather dataset.

G Full Forecasting Results

We offer comprehensive multivariate prediction outcomes in this section. Tables 8 encompasses test across 12 benchmark datasets. The outcomes reveal that ForecastGrapher demonstrates outstanding performance across all datasets. Specifically, ForecastGrapher achieves the top spot in terms of MSE and MAE a total of 31 and 26 times, respectively. Table 9 contains comparison results with advanced GNNs and Naive method.

Table 8: Full multivariate time series prediction results, with input length 96, output lengths in {12,24,48,96} for PEMS, {96,192,336,720} for others. Use bold to indicate the best result, and underline to indicate the second. The benchmarks are reported from [30]

Models		**Ours**		iTransformer		PatchTST		Crossformer		DLinear		RLinear		TimesNet		SCINet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.317**	**0.357**	0.334	0.368	0.329	0.367	0.404	0.426	0.345	0.372	0.355	0.376	0.338	0.375	0.418	0.438
	192	**0.365**	**0.383**	0.377	0.391	0.367	0.385	0.450	0.451	0.380	0.389	0.391	0.392	0.374	0.387	0.439	0.450
	336	**0.393**	**0.405**	0.426	0.420	0.399	0.410	0.532	0.515	0.413	0.413	0.424	0.415	0.410	0.411	0.490	0.485
	720	0.456	0.442	0.491	0.459	**0.454**	**0.439**	0.666	0.589	0.474	0.453	0.487	0.450	0.478	0.450	0.595	0.550
ETTm2	96	0.176	**0.259**	0.180	0.264	**0.175**	0.259	0.287	0.366	0.193	0.292	0.182	0.265	0.187	0.267	0.286	0.377
	192	**0.238**	**0.300**	0.250	0.309	0.241	0.302	0.414	0.492	0.284	0.362	0.246	0.304	0.249	0.309	0.399	0.445
	336	**0.296**	**0.338**	0.311	0.348	0.305	0.343	0.597	0.542	0.369	0.427	0.307	0.342	0.321	0.351	0.637	0.591
	720	**0.395**	**0.396**	0.412	0.407	0.402	0.400	1.730	1.042	0.554	0.522	0.407	0.398	0.408	0.403	0.960	0.735
ETTh1	96	**0.373**	0.397	0.386	0.405	0.414	0.419	0.423	0.448	0.386	0.400	0.386	**0.395**	0.384	0.402	0.654	0.599
	192	**0.424**	0.427	0.441	0.436	0.460	0.445	0.471	0.474	0.437	0.432	0.437	**0.424**	0.436	0.429	0.719	0.631
	336	**0.472**	0.448	0.487	0.458	0.501	0.466	0.570	0.546	0.481	0.459	0.479	**0.446**	0.491	0.469	0.778	0.659
	720	**0.479**	0.475	0.503	0.491	0.500	0.488	0.653	0.621	0.519	0.516	0.481	**0.470**	0.521	0.500	0.836	0.699
ETTh2	96	0.294	0.345	0.297	0.349	0.302	0.348	0.745	0.584	0.333	0.387	**0.288**	**0.338**	0.340	0.374	0.707	0.621
	192	**0.367**	0.396	0.380	0.400	0.388	0.400	0.877	0.656	0.477	0.476	0.374	**0.390**	0.402	0.414	0.860	0.689
	336	**0.407**	0.428	0.428	0.432	0.426	0.433	1.043	0.731	0.594	0.541	0.415	**0.426**	0.452	0.452	1.000	0.744
	720	**0.420**	0.441	0.427	0.445	0.431	0.446	1.104	0.763	0.831	0.657	0.420	**0.440**	0.462	0.468	1.249	0.838
Electricity	96	**0.138**	**0.235**	0.148	0.240	0.181	0.270	0.219	0.314	0.197	0.282	0.201	0.281	0.168	0.272	0.247	0.345
	192	**0.154**	**0.249**	0.162	0.253	0.188	0.274	0.231	0.322	0.196	0.285	0.201	0.283	0.184	0.289	0.257	0.355
	336	**0.169**	**0.264**	0.178	0.269	0.204	0.293	0.246	0.337	0.209	0.301	0.215	0.298	0.198	0.300	0.269	0.369
	720	**0.199**	**0.294**	0.225	0.317	0.246	0.324	0.280	0.363	0.245	0.333	0.257	0.331	0.220	0.320	0.299	0.390
Exchange	96	**0.086**	0.206	0.086	0.206	0.088	**0.205**	0.256	0.367	0.088	0.218	0.093	0.217	0.107	0.234	0.267	0.396
	192	0.181	0.303	0.177	0.299	**0.176**	**0.299**	0.470	0.509	0.176	0.315	0.184	0.307	0.226	0.344	0.351	0.459
	336	0.334	0.418	0.331	0.417	**0.301**	**0.397**	1.268	0.883	0.313	0.427	0.351	0.432	0.367	0.448	1.324	0.853
	720	0.869	0.702	0.847	**0.691**	0.901	0.714	1.767	1.068	**0.839**	0.695	0.886	0.714	0.964	0.746	1.058	0.797
Traffic	96	0.423	0.278	**0.395**	**0.268**	0.462	0.295	0.522	0.290	0.650	0.396	0.649	0.389	0.593	0.321	0.788	0.499
	192	0.445	0.285	**0.417**	**0.276**	0.466	0.296	0.530	0.293	0.598	0.370	0.601	0.366	0.617	0.336	0.789	0.505
	336	0.460	0.292	**0.433**	**0.283**	0.482	0.304	0.558	0.305	0.605	0.373	0.609	0.369	0.629	0.336	0.797	0.508
	720	0.503	0.312	**0.467**	**0.302**	0.514	0.322	0.589	0.328	0.645	0.394	0.647	0.387	0.640	0.350	0.841	0.523
Weather	96	0.161	**0.206**	0.174	0.214	0.177	0.218	**0.158**	0.230	0.196	0.255	0.192	0.232	0.172	0.220	0.222	0.306
	192	0.209	**0.251**	0.221	0.254	0.225	0.259	**0.206**	0.277	0.237	0.296	0.240	0.271	0.219	0.261	0.261	0.340
	336	**0.268**	**0.295**	0.278	0.296	0.278	0.297	0.272	0.335	0.283	0.335	0.292	0.307	0.280	0.306	0.309	0.378
	720	0.348	**0.345**	0.358	0.349	0.354	0.348	0.398	0.418	**0.345**	0.381	0.364	0.353	0.365	0.359	0.377	0.427
PEMS03	12	**0.065**	**0.168**	0.071	0.174	0.099	0.216	0.090	0.203	0.122	0.243	0.126	0.236	0.085	0.192	0.066	0.172
	24	**0.081**	**0.186**	0.093	0.201	0.142	0.259	0.121	0.240	0.201	0.317	0.246	0.334	0.118	0.223	0.085	0.198
	48	**0.111**	**0.220**	0.125	0.236	0.211	0.319	0.202	0.317	0.333	0.425	0.551	0.529	0.155	0.260	0.127	0.238
	96	**0.134**	**0.244**	0.164	0.275	0.269	0.370	0.262	0.367	0.457	0.515	1.057	0.787	0.228	0.317	0.178	0.287
PEMS04	12	0.075	0.181	0.078	0.183	0.105	0.224	0.098	0.218	0.148	0.272	0.138	0.252	0.087	0.195	**0.073**	**0.177**
	24	0.085	0.194	0.095	0.205	0.153	0.275	0.131	0.256	0.224	0.340	0.258	0.348	0.103	0.215	**0.084**	**0.193**
	48	0.099	0.213	0.120	0.233	0.229	0.339	0.205	0.326	0.355	0.437	0.572	0.544	0.136	0.250	**0.099**	**0.211**
	96	**0.112**	**0.227**	0.150	0.262	0.291	0.389	0.402	0.457	0.452	0.504	1.137	0.820	0.190	0.303	0.114	0.227
PEMS07	12	**0.058**	**0.152**	0.067	0.165	0.095	0.207	0.094	0.200	0.115	0.242	0.118	0.235	0.082	0.181	0.068	0.171
	24	**0.069**	**0.163**	0.088	0.190	0.150	0.262	0.139	0.247	0.210	0.329	0.242	0.341	0.101	0.204	0.119	0.225
	48	**0.085**	**0.179**	0.110	0.215	0.253	0.340	0.311	0.369	0.398	0.458	0.562	0.541	0.134	0.238	0.149	0.237
	96	**0.103**	**0.194**	0.139	0.245	0.346	0.404	0.396	0.442	0.594	0.553	1.096	0.795	0.181	0.279	0.141	0.234
PEMS08	12	0.081	0.184	**0.079**	**0.182**	0.168	0.232	0.165	0.214	0.154	0.276	0.133	0.247	0.112	0.212	0.087	0.184
	24	**0.115**	0.220	0.115	**0.219**	0.224	0.281	0.215	0.260	0.248	0.353	0.249	0.343	0.141	0.238	0.122	0.221
	48	**0.169**	**0.211**	0.186	0.235	0.321	0.354	0.315	0.355	0.440	0.470	0.569	0.544	0.198	0.283	0.189	0.270
	96	**0.197**	**0.234**	0.221	0.267	0.408	0.417	0.377	0.397	0.674	0.565	1.166	0.814	0.320	0.351	0.236	0.300
1^{st} Count		**31**	**26**	5	7	4	4	2	0	2	0	1	8	0	0	3	3

Table 9: Full comparison with GNNs and Naive method for multivariate time series prediction, with input length 96, output lengths in {12,24,48,96} for PEMS, {96,192,336,720} for others. Use bold to indicate the best result and the symbol '-' indicates exceeding memory.

Dataset		Electricity		Traffic		Weather		PEMS03		PEMS04		PEMS07		PEMS08	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Ours	96(12)	**0.138**	**0.235**	**0.423**	**0.278**	**0.161**	**0.206**	**0.065**	**0.168**	**0.075**	**0.181**	**0.058**	**0.152**	**0.081**	**0.184**
	192(24)	**0.154**	**0.249**	**0.445**	**0.285**	**0.209**	**0.251**	**0.081**	**0.186**	**0.085**	**0.194**	**0.069**	**0.163**	**0.115**	**0.220**
	336(48)	**0.169**	**0.264**	**0.460**	**0.292**	0.268	**0.295**	**0.111**	**0.220**	**0.099**	**0.213**	**0.085**	**0.179**	**0.169**	**0.211**
	720(96)	**0.199**	**0.294**	**0.503**	**0.312**	0.348	**0.345**	**0.134**	**0.244**	**0.112**	**0.227**	**0.103**	**0.194**	**0.197**	**0.234**
FourierGNN	96(12)	0.211	0.307	0.538	0.335	0.177	0.240	0.087	0.202	0.112	0.231	0.073	0.182	0.143	0.263
	192(24)	0.214	0.312	0.536	0.334	0.218	0.279	0.120	0.240	0.153	0.272	0.100	0.215	0.210	0.320
	336(48)	0.227	0.325	0.556	0.340	**0.265**	0.318	0.177	0.294	0.209	0.321	0.140	0.258	0.216	0.311
	720(96)	0.260	0.354	0.597	0.358	**0.336**	0.370	0.218	0.333	0.247	0.354	0.177	0.292	0.294	0.356
StemGNN	96(12)	0.165	0.267	0.576	0.339	0.181	0.250	0.119	0.244	0.144	0.276	0.120	0.242	0.246	0.319
	192(24)	0.180	0.283	0.593	0.345	0.226	0.289	0.179	0.305	0.188	0.317	0.168	0.282	0.281	0.337
	336(48)	0.200	0.306	0.624	0.366	0.287	0.338	0.191	0.303	0.234	0.342	0.184	0.285	0.305	0.356
	720(96)	0.243	0.345	0.655	0.373	0.379	0.406	0.258	0.355	0.303	0.396	0.265	0.346	0.380	0.393
Naive	96(12)	0.321	0.326	1.222	0.499	0.290	0.284	0.541	0.542	0.573	0.566	0.581	0.559	0.577	0.575
	192(24)	0.304	0.323	1.095	0.458	0.331	0.311	0.541	0.543	0.573	0.567	0.582	0.560	0.577	0.573
	336(48)	0.326	0.342	1.151	0.475	0.392	0.350	0.914	0.722	0.977	0.754	1.003	0.746	1.019	0.770
	720(96)	0.366	0.373	-	-	0.472	0.399	1.610	1.004	1.742	1.053	1.698	1.016	1.816	1.061

H Broader Impact

The proposed model, ForecastGrapher holds some potential impacts in multivariate time series forecasting and machine learning domains. It introduces a GNN framework specifically tailored for multivariate time series forecasting, enhancing the the ability to capture and express complex time series correlations. It achieves state-of-the-art performance on real-world datasets, making it more promising for practical applications like weather and electricity forecasting. Moreover, it serves as a valuable, unified modeling framework for time series correlations, offering a new path for future research. However, predictions about stocks, which involve significant uncertainty, may yield incorrect outcomes, harming the profit of investors.