GraphSTAGE: Channel-Preserving Graph Neural Networks for Time Series Forecasting

Tong Guan
Zhejiang University
Hangzhou, China
tguan@zju.edu.cn

Kaiyue Ma
Zhejiang University
Hangzhou, China
22332120@zju.edu.cn

Jiaheng Peng
Zhejiang University
Hangzhou, China
12232019@zju.edu.cn

Jun Liang
Zhejiang University
Hangzhou, China
jliang@zju.edu.cn

Bo Du
Griffith University
Gold Coast, Australia
bo.du@griffith.edu.au

Shirui Pan
Griffith University
Gold Coast, Australia
s.pan@griffith.edu.au

Abstract

Recent advancements in multivariate time series forecasting (MTSF) have increasingly focused on the core challenge of learning dependencies within sequences, specifically intra-series (temporal), inter-series (spatial), and cross-series dependencies. While extracting multiple types of dependencies can theoretically enhance the richness of learned correlations, it also increases computational complexity and may introduce additional noise. The trade-off between the variety of dependencies extracted and the potential interference has not yet been fully explored. To address this challenge, we propose GraphSTAGE, a purely graph neural network (GNN)-based model that decouples the learning of intra-series and inter-series dependencies. GraphSTAGE features a minimal architecture with a specially designed embedding and patching layer, along with the STAGE (Spatial-Temporal Aggregation Graph Encoder) blocks. Unlike channel-mixing approaches, GraphSTAGE is a channel-preserving method that maintains the shape of the input data throughout training, thereby avoiding the interference and noise typically caused by channel blending. Extensive experiments conducted on 13 real-world datasets demonstrate that our model achieves performance comparable to or surpassing state-of-the-art methods. Moreover, comparative experiments between our channel-preserving framework and channel-mixing designs show that excessive dependency extraction and channel blending can introduce noise and interference. As a purely GNN-based model, GraphSTAGE generates learnable graphs in both temporal and spatial dimensions, enabling the visualization of data periodicity and node correlations to enhance model interpretability.

CCS Concepts

• Computing methodologies → Neural networks; • Information systems → Data mining.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Keywords

Time Series Forecasting, Graph Neural Networks, Channel-Preserving

ACM Reference Format:
Tong Guan, Kaiyue Ma, Jiaheng Peng, Jun Liang, Bo Du, and Shirui Pan. 2018. GraphSTAGE: Channel-Preserving Graph Neural Networks for Time Series Forecasting. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Multivariate time series forecasting (MTSF) is pivotal in various domains such as traffic flow prediction and energy consumption forecasting. A key consideration in MTSF is effectively modeling the dependencies within the sequences—specifically the intra-series (temporal), inter-series (spatial), and potentially cross-series dependencies [18], as shown in Figure 2. Capturing these dependencies is crucial for understanding the underlying spatial and temporal relationships in the data, which directly impacts the accuracy of predictions.

However, many existing models focus on only one type of dependency. Common approaches employ channel-mixing techniques that project the original time series data $X_{\text{in}} \in \mathbb{R}^{N \times T}$ (where N is the number of nodes and T is the length of time series) into different representations. For instance, some methods transform X_{in} into $H_S \in \mathbb{R}^{N \times D}$ [22], which captures spatial dependencies among nodes, while others project it into $H_T \in \mathbb{R}^{T \times D}$ [15, 30, 39], emphasizing on temporal dependencies across time steps. These

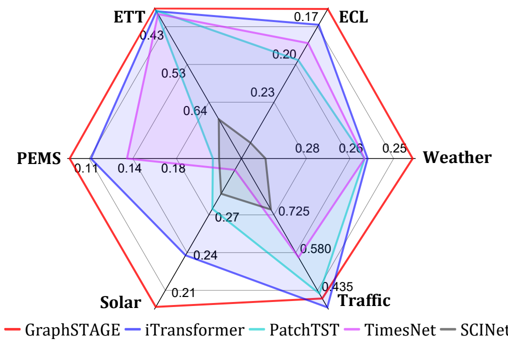

Figure 1: Performance of GraphSTAGE on average MSE.

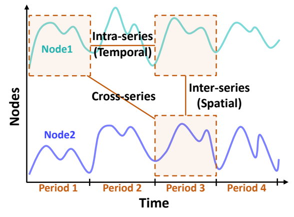

Figure 2: Dependencies between two sub-series in a multivariate time series.

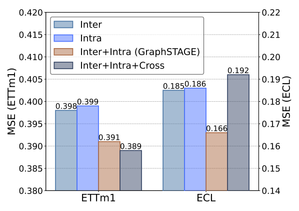

Figure 3: Performance of GraphSTAGE Variants on ETTm1 and ECL Datasets.

transformations often overlook at least one kind of dependency and fail to learn the underlying spatial or temporal graph structures [35], limiting the models' ability to extract inter-series or intra-series correlations effectively.

Recent models such as UniTST [18] and FourierGNN [34] attempt to capture multiple types of dependencies, including cross-series dependencies, by blending the temporal and spatial dimensions. They reshape the input data X_{in} from $\mathbb{R}^{N \times T}$ into a $\mathbb{R}^{NT \times 1}$ structure. While this approach theoretically allows for the simultaneous modeling of all dependencies, it also presents two significant challenges: (1) increased computational complexity and (2) a heightened risk of introducing additional noise.

First, mixing the channels may increases computational complexity. The complexity of weight multiplication operations escalates from $O(N^2)$ to $O((NT)^2)$ [18, 34], leading to exponentially higher computational costs. Consequently, these models often implement some compression mechanisms, such as router mechanism [38], to mitigate the computational burden. Despite these efforts, a trade-off between model size and performance persists. Achieving better performance frequently requires larger models, indicating that compression techniques may not fully address the efficiency concerns. To further illustrate this point, we conducted model variants experiments in Section 4.3. As shown in Table 5, our model outperforms VarC — a channel-mixing model similar to UniTST [18] and FourierGNN [34], as depicted in Figure 8, while also reducing memory usage by 83%.

Second, while blending channels allows these models to account for cross-series dependencies, it may introduce additional noise into the modeling process. Existing studies have often emphasized the benefits of capturing cross-series dependencies without fully considering the potential downsides of added noise. As shown in Figure 3, aggregating all dependencies may enhance predictive accuracy to some extent (as demonstrated by the improvement of performance on the ETTm1 dataset). However, it can also lead to overly complex models that struggle to compensate for the interference caused by the introduced noise, resulting in a sharp reduction in performance on the ECL dataset. This raises a crucial question: ***Is it truly necessary to model all these dependencies?***

We argue that modeling either a single type of dependency or multiple dependencies in a coupled manner is inefficient. Recently, channel-preserving approaches have demonstrated efficiency and effectiveness [20, 28]. To address the challenges of computational inefficiency and noise introduced by channel-mixing, we propose GraphSTAGE, a purely GNN-based model that decouples the learning of inter-series and intra-series dependencies while preserving the original channel structures. Unlike existing channel-mixing approaches, GraphSTAGE maintains the shape of the input data throughout the training process, thereby avoiding the interference caused by channel blending. To our knowledge, GraphSTAGE is the first purely graph-based, channel-preserving model. This design not only enhances computational efficiency but also reduces the noise associated with channel blending. More details about the potential noise introduced by channel blending can be found in Appendix A. Our contributions are threefold:

- We reflect on the extraction of dependencies in current time series models and emphasize that existing methods tend to overlook certain dependencies. Furthermore, we highlight that channel blending and excessive correlation extraction can introduce noise, and propose a channel-preserving framework to enable more accurate and robust dependencies modeling.
- We propose GraphSTAGE, a fully GNN-based method to effectively capture intra-series and inter-series dependencies, respectively, while generating interpretable correlation graphs. Moreover, its decoupled design allows for the independent extraction of specific dependencies as required.
- Experimentally, despite GraphSTAGE is structurally simple, it performs comparably to or surpasses state-of-the-art models across 13 MTSF benchmark datasets, as shown in Figure 1. Notably, GraphSTAGE ranks top-1 among 8 advanced models in 22 out of 30 comparisons, with results averaged across various prediction lengths.

By preserving the original data channels and decoupling dependencies learning, GraphSTAGE overcomes the key limitations of existing methods, providing a more efficient and interpretable solution for MTSF.

2 Related Works

Single Dependency Modeling. Traditional multivariate time series forecasting methods often focus on capturing a single type of dependency—either temporal (intra-series) or spatial (inter-series). Deep learning models such as CNNs, RNNs, GRUs and Formers [5, 7, 15, 21, 24, 30, 37, 39] excel at modeling sequential data by capturing temporal dynamics within each series. However, these models typically treat each spatial node independently, failing to account for inter-series dependency. On the other hand, models that focus solely on inter-series dependency, such as GNNs [1] and Formers [2, 13, 22], while effective at capturing spatial correlations, may not adequately model the temporal correlations within each series. Consequently, methods that concentrate on one type of dependency may fail to fully capture the complex correlations inherent in multivariate time series data.

Modeling Combined Dependencies. To address the limitations of single-dependency extracting models, several GNNs [12, 25, 31–33] have attempted to extract dependencies in both the temporal and spatial domains. However, these models often ignore global information extraction in either the spatial or temporal domain, focusing instead on local neighborhood information. Recent approaches have explored to capture multiple types of dependencies simultaneously

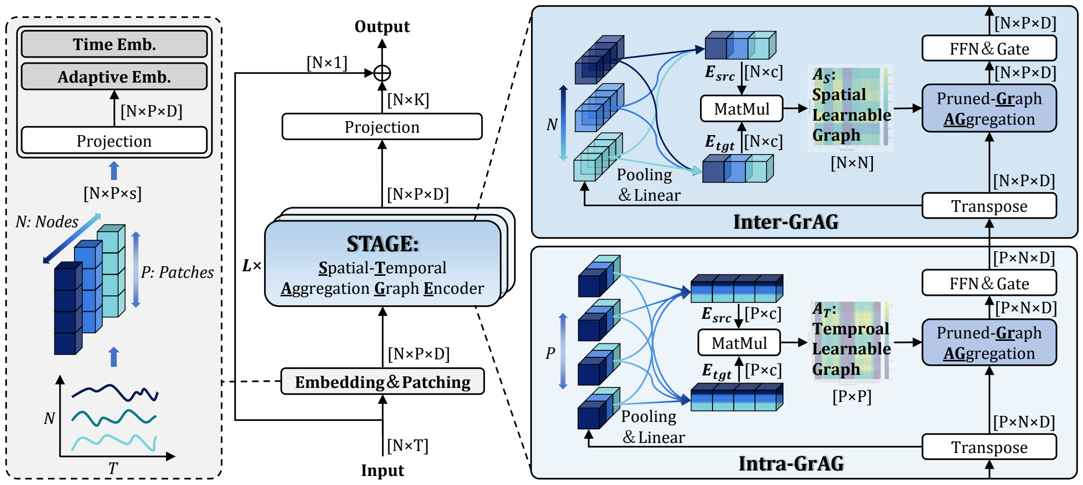

Figure 4: Overall Structure of GraphSTAGE. The model is composed of an Embedding & Patching layer followed by L stacked STAGE blocks. Each STAGE block employs a decoupled yet unified architecture integrating two key modules: the Intra-GrAG (Intra-series Pruned-Graph Aggregation), which captures temporal dependency and generates the temporal learnable graph A_T; the Inter-GrAG (Inter-series Pruned-Graph Aggregation), which captures spatial dependency and generates the spatial learnable graph A_S.

by blending the temporal and spatial dimensions. FourierGNN [34] and UniTST [18] construct hypervariate graph as input embeddings to represent time series with a unified view of spatial and temporal dynamics but overlook the potential interference caused by channel-mixing. Recognizing this issue, DGCformer [20] identifies irrelevant nodes in channel-mixing and adopts a grouping mechanism to focus attention on relevant nodes. Crossformer [38] and CARD [28] propose a two-stage framework to extract inter-series and intra-series dependencies, applying attention across both dimensions and then fuses the results. Building on these insights, we propose GraphSTAGE, a purely GNN-based model that decouples the learning of inter-series and intra-series dependencies while preserving the original input channels to avoid the interference introduced by channel blending.

3 GraphSTAGE

Problem Definition. Given the historical data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\} \in \mathbb{R}^{N\times T}$ with N nodes and T time steps, the multivariate time series forecasting task is to predict the future K time steps $\mathbf{Y} = \{\mathbf{x}_{T+1}, \dots, \mathbf{x}_{T+K}\} \in \mathbb{R}^{N\times K}$. This process can be given by:

$$\hat{\mathbf{Y}} = F_\theta(\mathbf{X}) = F_{\theta_t,\theta_s}(\mathbf{X}), \tag{1}$$

where $\hat{\mathbf{Y}}$ are the predictions corresponding to the ground truth $\mathbf{Y}$. The forecasting function is denoted as F_θ parameterized by θ. In practice, the channel-preserving model will be decoupled leverage a temporal network (parameterized by θ_t) to learn the intra-series dependency and a spatial network (parameterized by θ_s) to learn the inter-series dependency, respectively [28].

Overall Structure. Based on the motivation of using channel-preserving strategy to avoid interference introduced by channel-mixing, we propose GraphSTAGE—a purely GNN-based model with an architecture that decouples the learning of intra-series and inter-series dependencies, as illustrated in Figure 4. Our model comprises two key components: (1) a specially designed embedding and patching layer; and (2) the Spatial-Temporal Aggregation Graph Encoder (STAGE) block. In the embedding and patching layer, we introduce a more fine-grained time embedding to fully utilize the relative positions of data points within an hour as prior knowledge. In the STAGE block, we design a decoupled framework to respectively extract temporal and spatial dependencies, with corresponding learnable graphs that can be visualized to enhance interpretability. The pseudo-code can be found in Algorithm 1.

3.1 Tokenization via Embedding and Patching

Channel-preserving Embedding Strategy. Most signal intra-series dependency modeling models regard multiple nodes of the same time as the (temporal) token. As a result, they project the input data shaped as $X_{\text{in}} \in \mathbb{R}^{N\times T}$ into $\mathbb{R}^{T\times D}$, where D is the hidden dimension, and the original spatial dimension N is not preserved. Inspired by inter-series oriented models [22] in MTSF, we preserve the nodes dimension throughout the model, which proven competent by previous works [2]. Given a time series with N nodes, $X \in \mathbb{R}^{N\times T}$, we divide each univariate time series x_i into patches $x_p^i \in \mathbb{R}^{P\times s}$, with stride s and number of patches P [23]. A projection layer is then applied to map all the series into $X_p \in \mathbb{R}^{N\times P\times D}$, where D is the embedding dimension.

Refined Time Embedding to Enhance Relative Positioning. The effectiveness of static covariates that are available in advance has been validated in several MTSF models [9, 10, 17]. However, for datasets with a fixed sampling frequency below one hour (e.g., five minutes or fifteen minutes), previous models only embedded the 'Hour of Day' and 'Day of Week' information [2], which is insufficient to reflect the relative position within an hour. To address this limitation, we modify existing embedding methods by replacing the 'Hour of Day' embedding with a 'Timestamp of Day' embedding. This allows the embedding layer to adapt to the sample frequency, providing a more fine-grained time embedding that fully utilize the relative positions of data points within an hour as prior knowledge. Additionally, we introduce an learnable embedding to adaptively capture underlying dependencies. The process is presented below:

$$H = \text{Embedding}(X_p) = X_p + \mathbf{e}_{tod} + \mathbf{e}_{dow} + \mathbf{e}_{adp}{}^1, \tag{2}$$

where $H \in \mathbb{R}^{N\times P\times D}$ contains N embedded tokens of dimension D, $\mathbf{e}_{tod} \in \mathbb{R}^{P\times D}$ and $\mathbf{e}_{dow} \in \mathbb{R}^{P\times D}$ are learnable embeddings for 'Timestamp of Day' and 'Day of Week', respectively. $\mathbf{e}_{adp} \in \mathbb{R}^{P\times D}$ is generated using a random tensor method.

3.2 Spatial-Temporal Aggregation Graph Encoder

Our proposed STAGE block is illustrated in Figure 4. STAGE employs a decoupled yet unified architecture to aggregate information learned by Temporal Learnable Graph (A_T) and Spatial Learnable Graph (A_S). The Intra-series Pruned-Graph AGgregation module (Intra-GrAG) is responsible for extracting intra-series (temporal) dependencies and generating the A_T. Similarly, the Inter-series Pruned-Graph AGgregation module (Inter-GrAG) extracts inter-series (spatial) dependencies and generates the A_S.

Decoupled Spatial-Temporal Extraction with Unified Aggregation. STAGE is capable of learning intra-series and inter-series dependencies separately within a single block by utilizing a decoupled architecture composed of Intra-GrAG and Inter-GrAG modules. In STAGE block, the input tensor has dimensions $H \in \mathbb{R}^{N\times P\times D}$, where N is the number of nodes, P is the number of patches, and D is the embedding dimension. To learn intra-series dependencies, we first transpose the input tensor to shape $\mathbb{R}^{P\times N\times D}$, swapping the spatial and temporal dimensions. This restructure allows the model to focus on temporal relationships within each node across different time steps. After learning the intra-dependencies, we transpose the tensor back to its original shape $\mathbb{R}^{N\times P\times D}$ to learn inter-series dependencies, concentrating on the relationships between different nodes at each time step. By adopting this approach, we can employ a unified architecture for both intra-dependency and inter-dependency learning, simply by changing the order of the input dimensions.

Furthermore, since STAGE is a purely GNN-based method, the correlations among nodes or patches (time steps) learned by the model can be directly visualized, enhancing interpretability and providing insights into the data periodicity and node correlations.

Learnable Graph Generator for Temporal and Spatial Dimensions. Learnable Graphs are essential for characterizing both temporal and spatial similarities. STAGE adaptively learns the graph structures by generating separate adjacency matrices: A_T for patches (temporal dimension) and A_S for nodes (spatial dimension).

Since STAGE employs a unified aggregation mechanism, the principles of the Inter-GrAG and Intra-GrAG modules are analogous. Therefore, to avoid redundancy, the subsequent discussion will focus only on the components of the Inter-GrAG module. First, a Pooling layer downsamples the extracted temporal information. We can choose any pooling mechanisms in the temporal dimension as the Pool operation, such as max-pooling and mean-pooling. To capture directed similarities among nodes, we apply two Linear mappings to each node:

$$\begin{aligned} E_{src} &= \text{L2Norm}(H_{pool}W_{p1}), E_{tgt} = \text{L2Norm}(H_{pool}W_{p2}), \\ H_{pool} &= \text{Pool}(H_{in}) \end{aligned} \tag{3}$$

where $H_{pool} \in \mathbb{R}^{N\times D}$. Here, $H_{in} \in \mathbb{R}^{N\times P\times D}$ is obtained by transposing the output of intra-GrAG module, which originally has the shape $\mathbb{R}^{P\times N\times D}$. $W_{p1} \in \mathbb{R}^{D\times c}$, $W_{p2} \in \mathbb{R}^{D\times c}$ are two trainable matrices, and $E_{src} \in \mathbb{R}^{N\times c}$ and $E_{tgt} \in \mathbb{R}^{N\times c}$ are the source and target embedding matrices of all nodes, respectively. The L2 normalization ensures that each embedding matrices has a unit norm, facilitating stable training and enhancing model performance.

The directed similarities between each pair of nodes can be extracted as follows [31]:

$$A_S = \text{SoftMax}(\text{ReLU}(E_{src} \cdot E_{tgt}^T)). \tag{4}$$

The ReLU activation is used to avoid negative values. SoftMax function is employed to normalize values in the matrix. In this way, we obtain the spatial learnable graph $A_S \in \mathbb{R}^{N\times N}$, which serves as a global similarities matrix. It should be noted that the parameters of this similarity matrix are derived for each individual sample. Consequently, when the sample changes, the similarity weights among different nodes also change.

Pruned-Graph Aggregation Mechanism. In the Intra-GrAG module, this mechanism performs graph convolutions on the learned graph A_T. In the Inter-GrAG module, it performs graph convolutions on the learned graph A_S, aggregating information from global nodes while pruning irrelevant or weak connections. The pruning operation reduces noise and enhances the model's ability to

Algorithm 1 The learning algorithm of GraphSTAGE.

```
Require: Input historical time series $\mathbf{X} \in \mathbb{R}^{N\times T}$; input length $T$;
    prediction length $K$; nodes number $N$; patches number $P$; patch
    stride $s$;embedding dimension $D$; STAGE block number $L$.
 1: $\mathbf{Base} = \mathtt{Mean}(\mathbf{X})$                          ▷ $\mathbf{Base} \in \mathbb{R}^{N\times 1}$
 2: $\mathbf{X} = \mathtt{Patching}(\mathbf{X})$                         ▷ $\mathbf{X} \in \mathbb{R}^{N\times P\times s}$
 3: ▷ Projecton maps series into embedding dimension $D$.
 4: $\mathbf{X_p} = \mathtt{Projecton}(\mathbf{X})$                     ▷ $\mathbf{X_p} \in \mathbb{R}^{N\times P\times D}$
 5: ▷ Refined time embedding to enhance relative positioning.
 6: $\mathbf{H}^0 = \mathtt{Embedding}(\mathbf{X_p})$                   ▷ $\mathbf{H}^0 \in \mathbb{R}^{N\times P\times D}$
 7: for $l$ in $\{1, \ldots, L\}$:            ▷ Run through stacked STAGE blocks.
 8:     ▷ Intra-GrAG module to capture temporal dependency.
 9:     $\mathbf{H_t}^{l-1} = \mathtt{IntraGrAG}(\mathbf{H}^{l-1}.\mathtt{transpose})$ ▷ $\mathbf{H_t}^{l-1} \in \mathbb{R}^{P\times N\times D}$
10:     ▷ Inter-GrAG module to capture spatial dependency.
11:     $\mathbf{H}^l = \mathtt{InterGrAG}(\mathbf{H_t}^{l-1}.\mathtt{transpose})$    ▷ $\mathbf{H}^l \in \mathbb{R}^{N\times P\times D}$
12: End for
13: $\hat{\mathbf{Y}} = \mathtt{Projecton}(\mathbf{H}^L)$ ▷ Project tokens to prediction, $\hat{\mathbf{Y}} \in \mathbb{R}^{N\times K}$
14: $\hat{\mathbf{Y}} = \hat{\mathbf{Y}} + \mathbf{Base}$                     ▷ $\hat{\mathbf{Y}} \in \mathbb{R}^{N\times K}$
15: Return $\hat{\mathbf{Y}}$                    ▷ Return the prediction result $\hat{\mathbf{Y}}$
```

[1]The process utilizes the broadcasting mechanism in PyTorch.

Table 1: Multivariate forecasting results with prediction lengths $K \in \{12, 24, 48, 96\}$ for PEMS and $K \in \{96, 192, 336, 720\}$ for others and fixed lookback length $T = 96$. Results are averaged from all prediction lengths. *AVG* means further averaged by subsets.

Models	Ours		iTransformer		RLinear		PatchTST		Crossformer		TimesNet		DLinear		SCINet	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	**0.166**	**0.263**	0.178	0.270	0.219	0.298	0.205	0.290	0.244	0.334	0.192	0.295	0.212	0.300	0.268	0.365
ETTm1	0.391	**0.394**	0.407	0.410	0.414	0.407	**0.387**	0.400	0.513	0.496	0.400	0.406	0.403	0.407	0.485	0.481
ETTm2	**0.278**	**0.325**	0.288	0.332	0.286	0.327	0.281	0.326	0.757	0.610	0.291	0.333	0.350	0.401	0.571	0.537
ETTh1	**0.445**	**0.430**	0.454	0.447	0.446	0.434	0.469	0.454	0.529	0.522	0.458	0.450	0.456	0.452	0.747	0.647
ETTh2	0.387	0.407	0.383	0.407	**0.374**	**0.398**	0.387	0.407	0.942	0.684	0.414	0.427	0.559	0.515	0.954	0.723
ETT (AVG)	**0.375**	**0.388**	0.383	0.399	0.380	0.392	0.381	0.397	0.685	0.578	0.391	0.404	0.442	0.444	0.689	0.597
Exchange	0.376	0.409	0.360	**0.403**	0.378	0.417	0.367	0.404	0.940	0.707	0.416	0.443	**0.354**	0.414	0.750	0.626
Traffic	0.462	0.294	**0.428**	**0.282**	0.626	0.378	0.481	0.304	0.550	0.304	0.620	0.336	0.625	0.383	0.804	0.509
Weather	**0.243**	**0.274**	0.258	0.278	0.272	0.291	0.259	0.281	0.259	0.315	0.259	0.287	0.265	0.317	0.292	0.363
Solar-Energy	**0.192**	0.267	0.233	**0.262**	0.369	0.356	0.270	0.307	0.641	0.639	0.301	0.319	0.330	0.401	0.282	0.375
PEMS03	**0.097**	**0.210**	0.113	0.221	0.495	0.472	0.180	0.291	0.169	0.281	0.147	0.248	0.278	0.375	0.114	0.224
PEMS04	**0.090**	**0.200**	0.111	0.221	0.526	0.491	0.195	0.307	0.209	0.314	0.129	0.241	0.295	0.388	0.092	0.202
PEMS07	**0.080**	**0.179**	0.101	0.204	0.504	0.478	0.211	0.303	0.235	0.315	0.124	0.225	0.329	0.395	0.119	0.234
PEMS08	**0.139**	**0.220**	0.150	0.226	0.529	0.487	0.280	0.321	0.268	0.307	0.193	0.271	0.379	0.416	0.158	0.244
PEMS (AVG)	**0.102**	**0.203**	0.119	0.218	0.514	0.482	0.217	0.305	0.220	0.304	0.148	0.246	0.320	0.394	0.121	0.222
1st Count	**22**		4		2		1		0		0		1		0	

focus on the most significant correlations. To avoid redundancy and for simplicity, the subsequent discussion will focus only on the components of the Inter-GrAG module.

Graph attention network (GAT) [27] is a powerful model for extracting spatial dependencies, allocating different weights to neighbor nodes. Pruned-Graph Aggregation (PGA) can be regarded as a Special GAT with three specific improvements: 1) input embeddings are the extracted temporal embeddings rather than the original features; 2) the input nodes learnable graph will be pruned to make the model concentrate on the most significant connections; 3) the spatial dependencies among nodes is global rather than localized in neighborhoods. In this way, PGA incorporates spatial information effectively and aggregates global information without any prior knowledge, such as pre-defined static graph. The whole process can be formulated as below:

$$H_{ag} = H_{in}W_1 + \text{Prune}(A_S)H_{in}W_2 + \text{Prune}(A_S)^T H_{in}W_3, \quad (5)$$

where $W_1, W_2, W_3 \in \mathbb{R}^{D\times D}$ are trainable matrices and $H_{ag} \in \mathbb{R}^{N\times P\times D}$. The Prune operation retains the top-k values to focus on the most significant connections, where $k = N \times \alpha$ for Inter-GrAG module and $k = P\times\alpha$ for Intra-GrAG module, with a coefficient α between 0 and 1 (e.g., 0.7). After that, a Feed-Forward Network (FFN) and Gate is employed to obtain the output of Encoders H_E. The FFN processes the aggregated features to capture nonlinear transformations, while the gating mechanism controls the flow of information. This gating enhances the model's capacity to capture complex dependencies by adaptively weighing the importance of different features.

In summary, STAGE decouples intra-series and inter-series dependencies within a unified pruned-graph aggregation mechanism, avoiding computational overhead and potential noise introduced by channel blending. Its fully graph-based mechanism enhances interpretability. Further discussion about the variants of STAGE will be delivered in the Section 4.3.

4 Experiments

4.1 Experimental Setup

Datasets. To validate the performance of GraphSTAGE, we conduct extensive benchmarks on 13 real-world datasets, including ETT (4 subsets), ECL, Exchange, Traffic, Weather, Solar-Energy datasets proposed in LSTNet [14], and PEMS (4 subsets) collected by the Performance Measurement System (PeMS) [4] and proposed in ASTGCN [6].

Experimental Settings. All experiments are conducted on a single RTX 4090 24GB GPU, and we utilize the Adam [11] optimizer to optimize the training process. All experiments are repeated five times and we report the averaged results. The batch size of GraphSTAGE is consistently set to 16, and the number of training epochs is fixed to 10. We conduct a grid search to determine the best configuration. We consistently set the embedding dimension D to 64, and the number of STAGE layers between 1 and 2. Normalization is skipped before the embedding process for the PEMS and Solar-Energy datasets, and performed in advance for all other datasets. We partition the dataset for train-validation-test following the methodology established in TimesNet [29], to ensure the comparability of subsequent experiments. For the forecasting settings, the lookback length for all datasets is set to 96. The prediction horizon varies across $\{12, 24, 48, 96\}$ for the PEMS datasets and $\{96, 192, 336, 720\}$ for the other datasets.

Baselines. We have selected seven well-known forecasting models as our benchmarks, including (1) Transformer-based methods: iTransformer [22], Crossformer [38], PatchTST [23]; (2) Linear-based methods: DLinear [36], RLinear [16]; and (3) TCN-based methods: SCINet [19], TimesNet [29]. Additional comparisons with four advanced GNNs are provided in Table 2.

4.2 Main Results

Outstanding Performance of GraphSTAGE Across 13 Datasets: Ranking First in 22 out of 30 Comparisons. Comprehensive forecasting results are presented in Table 1, with the best performances in **red** and the second in blue. Lower MSE/MAE values indicate better prediction performance. The quantitative results reveal that GraphSTAGE demonstrates outstanding performance across all datasets, including node-based multivariate time series datasets (e.g., PEMS, Solar-Energy) and attribute-based multivariate time series datasets (e.g., ETT, Weather, ECL). GraphSTAGE achieves the best performance in 22 out of 30 cases, significantly outperforming the recent state-of-the-art (SOTA) iTransformer, which ranks first in only 4 instances. Compared to iTransformer, the MSE on the ECL, ETT (AVG), Weather, Solar-Energy, and PEMS (AVG) datasets is significantly reduced by 6.7%, 2.1%, 5.8%, 17.6%, and 14.3%, respectively. Specifically, on the PEMS07 dataset, which has the largest number of nodes, GraphSTAGE outperforms the recent SOTA iTransformer by 20.8%, indicating its potential for application to larger-scale MTSF tasks, such as extensive grid management. Moreover, the recent SOTA iTransformer performs poorly on attribute-based multivariate time series datasets (e.g., ETT) because it is a single-dependency learning model that focuses solely on inter-series (spatial) dependencies. In attribute-based datasets, there is generally no strong direct interaction or correlation between the attributes (e.g., temperature, wind speed), which makes it more necessary to extract intra-series (temporal) dependencies. This observation further validates the effectiveness of GraphSTAGE in capturing both intra-series and inter-series dependencies, leading to superior forecasting accuracy across diverse types of multivariate time series data.

Supplementary Visualization and Additional Baseline Comparisons. In order to better compare the models, we present supplementary prediction results visualization for PEMS07 dataset, which has the largest number of nodes (883 nodes) among all the datasets. For all baselines, the input length is set to 96, with a forecasting horizon of 96 time steps. As shown in Figures 5, our predictions perfectly match the trend of the GroundTruth.

Table 2 contains additional comparison results with advanced GNNs [3, 8, 31, 34]. The results indicate that GraphSTAGE achieves top-1 performance in all cases. Notably, on the largest-scale dataset (ECL with 321 nodes), it outperforms the second-best model (CrossGNN) by significant margins, with reductions in MSE and MAE exceeding 17.4% and 12.3%, respectively.

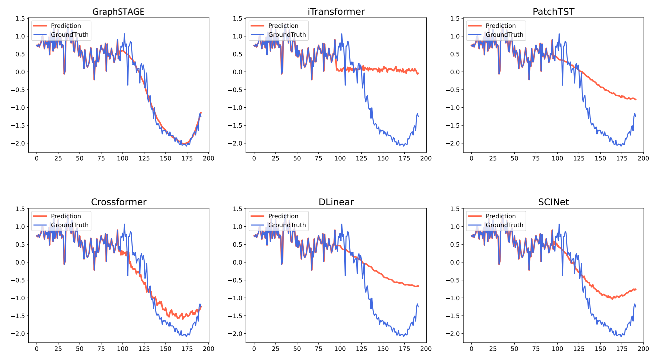

Figure 5: Sample visualization across models on PEMS07 dataset, with forecast horizon 96.

Table 2: Additional comparison with advanced GNNs, following the setting of TimesNet [29]. The input sequence length is set to 96 for all baselines. *AVG* means the average results from all four prediction lengths: $\{96, 192, 336, 720\}$**.**

Models	Ours		FourierGNN [34]		CrossGNN [8]		StemGNN [3]		MTGNN [31]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.391**	**0.394**	0.453	0.448	0.393	0.404	0.550	0.537	0.469	0.446
ETTh2	**0.387**	**0.407**	0.543	0.517	0.393	0.418	1.158	0.812	0.465	0.509
Weather	**0.243**	**0.274**	0.257	0.305	0.247	0.289	0.289	0.342	0.314	0.355
ECL	**0.166**	**0.263**	0.221	0.318	0.201	0.300	0.215	0.316	0.251	0.347
Average	**0.297**	**0.335**	0.369	0.397	0.309	0.353	0.553	0.502	0.375	0.414

Model Efficiency and Increasing lookback length. We conducted a comprehensive comparison of the performance, training speed, and memory usage of GraphSTAGE against other models on the ECL dataset, as shown in Figure 6. While GraphSTAGE may not achieve the best results in terms of training speed and memory usage, it delivers the best predictive performance. To ensure a fair comparison, we followed the settings in [2] and set the batch size of GraphSTAGE to 32. Compared with Crossformer [38], the only baseline model that learns multiple dependencies, GraphSTAGE's memory usage decreased by 47.0%, training time decreased by 60.9%,

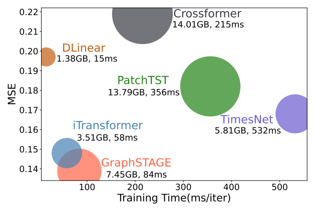

Figure 6: Model efficiency comparison on ECL dataset with input length 96 and output length 96.

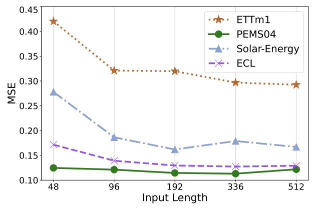

Figure 7: Forecasting with output length 96 and input length in {48, 96, 192, 336, 512} across four datasets.

and predictive performance improved by 36.5%. This significant reduction in computational resources, combined with an improvement in accuracy, highlights GraphSTAGE's efficiency. Therefore, GraphSTAGE effectively balances model size, computational speed, and predictive accuracy. Our model achieves superior performance at an acceptable computational cost, demonstrating its practicality for real-world MTSF tasks.

Additionally, to evaluate the ability of GraphSTAGE to leverage increasing lookback length, we conducted experiments on the ETTm1, PEMS04, Solar-Energy, and ECL datasets. The input lengths were varied from shorter to longer as 48, 96, 192, 336, 512, while the forecasting horizon was fixed at the next 96 time steps. As shown in Figure 7, the model's performance steadily improves as the input length increases. Notably, when the input length expands from 48 to 96, the MSE decreases most significantly. This demonstrates that the *Intra-GrAG* module of GraphSTAGE effectively captures intra-series dependencies, enabling it to learn more temporal correlations from longer input series.

4.3 Model Analysis

Ablation on Correlation Learning Mechanism. To verify the effectiveness of GraphSTAGE components, we provide detailed ablation studies covering both removing components (w/o) and replacing components (Replace) experiments. The averaged results are listed in Table 3. In the replacement experiments, we use the attention from Crossformer [38], which has been proved more accurate than vanilla Transformer [26]. Removing any component from GraphSTAGE results in performance degradation. GraphSTAGE utilizes Inter-GrAG module on the spatial dimension and Intra-GrAG module on the time dimension, generally achieving better performance than when replaced by the attention from Crossformer.

Table 3: Ablations on the Correlation Learning Mechanism. We remove or replace components along spatial and temporal dimensions. The average results of all predicted lengths $K \in \{96, 192, 336, 720\}$ are listed here .

Design	Spatial	Temporal	ETTm1		ECL		Traffic		Solar-Energy	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
GraphSTAGE	**Inter-GrAG**	**Intra-GrAG**	**0.391**	**0.394**	**0.166**	**0.263**	0.462	**0.294**	**0.192**	0.267
w/o	Inter-GrAG	w/o	0.398	0.400	0.185	0.277	0.478	0.312	0.225	0.292
	w/o	Intra-GrAG	0.399	0.400	0.186	0.276	0.509	0.320	0.239	0.294
Replace	Inter-GrAG	Attention	0.395	0.401	0.168	0.265	0.478	0.303	0.206	0.270
	Attention	Intra-GrAG	0.403	0.406	0.171	0.268	0.459	0.305	0.206	0.276
	Attention	Attention	0.395	0.404	0.171	0.269	**0.453**	0.300	0.204	**0.264**

Ablation on Embedding&Patching Mechanism. As shown in Table 4, we test the components of the Embedding&Patching module

Table 4: Ablations on the Embedding&Patching Mechanism. The average results of all predicted lengths $K \in \{12, 24, 48, 96\}$ are listed here.

Design	PEMS03		PEMS04		PEMS07		PEMS08	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
GraphSTAGE	**0.097**	**0.210**	**0.090**	**0.200**	**0.080**	**0.179**	**0.139**	**0.220**
w/o Patching	0.110	0.222	0.100	0.215	0.096	0.199	0.176	0.253
w/o Time Emb.	0.114	0.223	0.099	0.211	0.091	0.193	0.199	0.264
w/o Adaptive Emb.	0.121	0.257	0.098	0.211	0.116	0.221	0.203	0.260

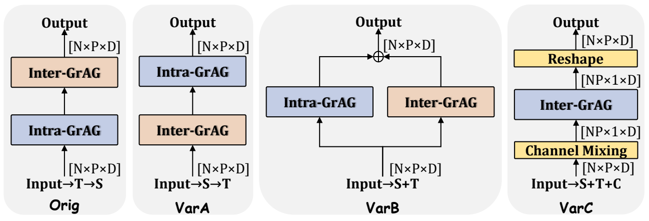

Figure 8: Model Variants. Orig (GraphSTAGE) follows an input→T→S structure, sequentially extracting temporal and then spatial dependencies. VarA uses input→S→T, reversing the order but remaining sequential. VarB employs input→S+T, a parallel structure that decouples temporal and spatial extraction before fusion. VarC utilizes input→S+T+C (C represents cross-series dependency as shown in Figure 2), incorporating channel-mixing with a unified architecture similar to FourierGNN [34], extracting all three types of dependencies within a unified framework.

through three ablation studies: w/o Patching, w/o Time Embedding, and w/o Adaptive Embedding. The performance of GraphSTAGE consistently surpasses all of the ablation variants, indicating that accurate prediction relies not only on the dependency extraction module but also importantly on the use of prior knowledge.

Variants Comparison. We designed three model variants to validate the effectiveness of our framework. As illustrated in Figure 8, the proposed GraphSTAGE model is referred to as **Orig**.

In Variant **VarA**, we swapped the positions of the *Inter-GrAG* and *Intra-GrAG* modules. The *Inter-GrAG* module now processes the original features, rather than the temporal embeddings extracted by the *Intra-GrAG* module. The swap aims to validate the rationale of the proposed sequential architecture. VarA's performance in Table 5, shows that the original sequence—inputting the extracted temporal embeddings into the *Inter-GrAG*—contributes positively to the model's effectiveness.

In Variant **VarB**, the *Inter-GrAG* and *Intra-GrAG* modules are connected in parallel rather than sequentially. This configuration investigates whether simultaneous processing of inter-series and intra-series dependencies impacts model performance compared to the original sequential architecture. VarB's performance in Table 5 confirms the sequential structure is more effective than the parallel.

Table 5: Model variants. All models are evaluated on 4 predication lengths. The best results are in red, the second results are in blue, and the highest memory usage is in bold.

Models		Orig (GraphSTAGE)			VarA			VarB			VarC		
Metric		MSE	MAE	Mem (GB)	MSE	MAE	Mem (GB)	MSE	MAE	Mem (GB)	MSE	MAE	Mem (GB)
ETTm1	96	0.319	**0.356**	0.522	0.326	0.361	0.522	**0.316**	0.357	0.522	0.325	0.361	**0.558**
	192	0.367	0.381	0.522	**0.365**	**0.383**	0.522	0.373	0.390	0.522	0.370	0.387	**0.578**
	336	**0.394**	**0.400**	0.522	0.403	0.413	0.522	0.401	0.409	0.522	0.402	0.410	**0.578**
	720	0.482	**0.441**	0.544	**0.456**	0.444	0.544	0.476	0.450	0.544	0.458	0.443	**0.597**
	AVG	0.391	**0.394**	0.528	**0.388**	0.400	0.528	0.392	0.402	0.528	0.389	0.400	**0.578**
ECL	96	**0.139**	**0.237**	4.066	0.166	0.257	3.920	0.156	0.250	4.110	0.170	0.265	**23.703**
	192	**0.155**	**0.251**	4.080	0.172	0.265	3.920	0.169	0.262	4.124	0.175	0.267	**23.725**
	336	**0.175**	**0.272**	4.086	0.193	0.285	4.100	0.184	0.277	4.186	0.192	0.285	**23.749**
	720	**0.196**	**0.292**	4.144	0.235	0.319	4.120	0.225	0.313	4.200	0.231	0.317	**23.794**
	AVG	**0.166**	**0.263**	4.094	0.192	0.282	4.015	0.184	0.276	4.155	0.192	0.284	**23.743**

In Variant **VarC**, we adopt the same channel-mixing architecture as UniTST [18] and FourierGNN [34], which reshapes the input data X_{in} from $\mathbb{R}^{N\times T}$ to a $\mathbb{R}^{NT\times 1}$ structure. This reshaping enables the coupled learning of three types of dependencies within a unified structure. By comparing Orig with VarC, we are able to evaluate the effectiveness of our proposed channel-preserving framework. From the results in Table 5, we observe that although channel-mixing demonstrates stronger results in some cases—e.g., on the ETTm1 dataset with an input length of 96 and forecast length of 720, it outperforms Orig by 5.8%—this improvement comes at the cost of increased memory usage. Moreover, on larger datasets like ECL, channel blending leads to an exponential increase in parameters and a sharp decrease in prediction accuracy. By treating the original multivariate time series as a univariate time series of length $N \times T$, the coupled dependencies learning introduces more interference and noise compared to the proposed decoupled framework. This highlights the advantages of our channel-preserving strategy, which maintains computational efficiency and reduces noise while effectively capturing the essential dependencies.

The comparisons among these variants validate the design of GraphSTAGE. The sequential structure in **Orig** (GraphSTAGE) proves to be more effective than altering the module order (**VarA**) or processing dependencies in parallel (**VarB**). Additionally, our channel-preserving framework demonstrates superior scalability and efficiency compared to the channel-mixing strategy in **VarC**, especially on larger datasets. This underscores the importance of preserving the original data structure and decoupling the learning of inter-series and intra-series dependencies in MTSF models.

Visualization of Learned Dependencies. We conducted heatmap visualizations of dependencies on three datasets with different sampling frequencies: ETTm1, ECL, and PEMS04. For ETTm1, the input length is set to 288, corresponding to 3 days of data, as the sampling frequency is 15 minutes (288 × 15 minutes = 3 days). For ECL, the input length is 96, meaning each sample contains 4 days of data, given the sampling frequency of 1 hour (96 × 1 hour = 4 days). For PEMS04 with 5-minute intervals, the input length is set to 576, meaning each sample contains 2 days of input data.

In experiments, we set the patch stride to 2 and randomly selected one Temporal Learnable Graph (A_T) for each dataset, as shown in Figure 9. In ETTm1's $A_T^{(1)}$, peaks occur every 48 patches, corresponding to 24 hours. Similarly, ECL's $A_T^{(2)}$ shows peak every 12 patches (24 hours), and PEMS04's $A_T^{(3)}$ peaks every 144 patches (24 hours). These visualizations demonstrate that the periodicity extracted by the Inter-GrAG module matches the inherent daily periodicity of each dataset. This match confirms our method effectively captures and visualizes the daily patterns in the data.

Figure 10 presents a randomly selected sample of the Spatial Learnable Graph (A_S) along with the corresponding ground truth of the nodes. In A_S, we observe that nodes 184 and 282 exhibit a high correlation—as indicated by a bright spot within the green square in Figure 10 (left), representing a correlation coefficient close to 1. Conversely, nodes 184 and 83 show almost zero correlation—there is no bright spot within the orange square in Figure 10 (left), indicating a correlation coefficient close to 0. Correspondingly, as shown in Figure 10 (right), the ground truth for nodes 184 and 282 behaves very similarly, whereas node 83 displays completely different trends. This randomly selected visualization demonstrates that the correlations among nodes in A_S learned by the Inter-GrAG module match the ground truth, confirming the effectiveness of GraphSTAGE in capturing inter-series dependencies.

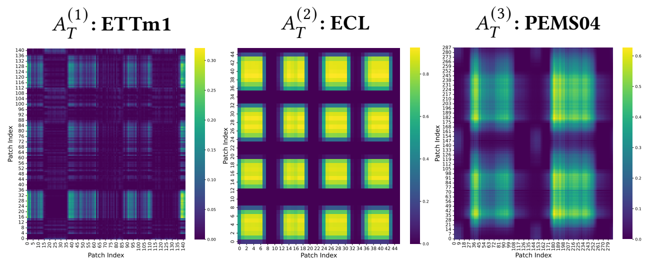

Figure 9: Visualization of Temporal Learnable Graphs (A_T) across different datasets (ETTm1, ECL, PEMS04). Each column represents a randomly selected A_T from the results of GraphSTAGE.

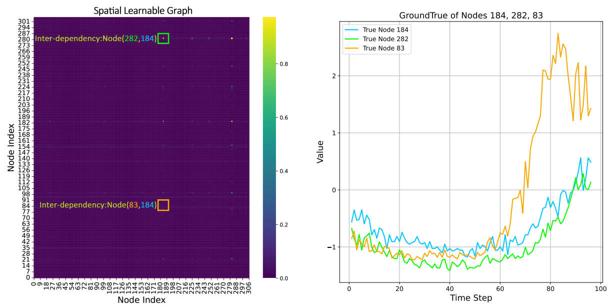

Figure 10: A randomly selected sample of the Spatial Learnable Graph (A_S) on the PEMS04 dataset (left), along with the corresponding GroundTruth of the nodes (right).

5 Conclusion

Current models primarily focus on the advantages of channel-mixing methods for extracting multiple dependencies, often neglecting the noise these approaches can introduce. GraphSTAGE is the first model to directly address this issue. Through the model variants experiments in Section 4.3, we validated the presence of such interference, underscoring the limitations of excessive dependency extraction. To mitigate these challenges, GraphSTAGE utilizes a decoupled architecture that independently extracts inter-series and intra-series dependencies. As a fully graph-based, channel-preserving framework, GraphSTAGE maintains the integrity of the original channel structures, effectively avoiding the interference and noise associated with channel blending. Extensive experiments conducted on 13 real-world datasets demonstrate that GraphSTAGE achieves performance on par with, or surpassing, state-of-the-art methods. Future research could explore decoupled extraction of cross-series dependencies and develop inductive models that maintain channel preservation.

References

[1] 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.
[2] Wanlin Cai, Kun Wang, Hao Wu, Xiaoxu Chen, and Yuankai Wu. 2024. ForecastGrapher: Redefining Multivariate Time Series Forecasting with Graph Neural Networks. *arXiv preprint arXiv:2405.18036* (2024).
[3] 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.
[4] Tom Choe, Alexander Skabardonis, and Pravin Varaiya. 2002. Freeway Performance Measurement System: Operational Analysis Tool. *Transportation Research Record* 1811 (01 2002). https://doi.org/10.3141/1811-08
[5] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014.*
[6] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.
[7] S Hochreiter. 1997. Long Short-term Memory. *Neural Computation MIT-Press* (1997).
[8] Qihe Huang, Lei Shen, Ruixin Zhang, Shouhong Ding, Binwu Wang, Zhengyang Zhou, and Yang Wang. 2023. Crossgnn: Confronting noisy multivariate time series via cross interaction refinement. *Advances in Neural Information Processing Systems* 36 (2023), 46885–46902.
[9] Yuanpei Huang and Nanfeng Xiao. 2024. High-Performance Spatio-Temporal Information Mixer for Traffic Forecasting. In *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[10] Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. 2023. PDFormer: Propagation Delay-aware Dynamic Long-range Transformer for Traffic Flow Prediction. In *AAAI*. AAAI Press.
[11] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR* (2015).
[12] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural relational inference for interacting systems. In *International conference on machine learning*. PMLR, 2688–2697.
[13] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations.*
[14] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. *SIGIR* (2018).
[15] Jianxin Li, Xiong Hui, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. *arXiv: 2012.07436* (2021).
[16] Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. 2023. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721* (2023).
[17] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.
[18] Juncheng Liu, Chenghao Liu, Gerald Woo, Yiwei Wang, Bryan Hooi, Caiming Xiong, and Doyen Sahoo. 2024. UniTST: Effectively Modeling Inter-Series and Intra-Series Dependencies for Multivariate Time Series Forecasting. *arXiv preprint arXiv:2406.04975* (2024).
[19] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. 2022. SCINet: time series modeling and forecasting with sample convolution and interaction. *NeurIPS* (2022).
[20] Qinshuo Liu, Yanwen Fang, Pengtao Jiang, and Guodong Li. 2024. DGCformer: Deep Graph Clustering Transformer for Multivariate Time Series Forecasting. *arXiv preprint arXiv:2405.08440* (2024).
[21] 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.*
[22] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations.*
[23] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *ICLR* (2023).
[24] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep state space models for time series forecasting. *Advances in neural information processing systems* 31 (2018).
[25] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete Graph Structure Learning for Forecasting Multiple Time Series. In *International Conference on Learning Representations.*
[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. *NeurIPS* (2017).
[27] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.
[28] Xue Wang, Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. 2024. CARD: Channel aligned robust blend transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations.*
[29] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. *ICLR* (2023).
[30] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *NeurIPS* (2021).
[31] 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 *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 753–763.
[32] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. AAAI Press, 1907–1913.
[33] Nancy Xu, Chrysoula Kosma, and Michalis Vazirgiannis. 2023. TimeGNN: Temporal Dynamic Graph Learning for Time Series Forecasting. In *International Conference on Complex Networks and Their Applications*. Springer, 87–99.
[34] 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).
[35] Guoqi Yu, Jing Zou, Xiaowei Hu, Angelica I Aviles-Rivero, Jing Qin, and Shujun Wang. 2024. Revitalizing Multivariate Time Series Forecasting: Learnable Decomposition with Inter-Series Dependencies and Intra-Series Variations Modeling. In *Forty-first International Conference on Machine Learning.*
[36] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting? *AAAI* (2023).
[37] Yunhao Zhang, Minghao Liu, Shengyang Zhou, and Junchi Yan. 2024. UP2ME: Univariate Pre-training to Multivariate Fine-tuning as a General-purpose Framework for Multivariate Time Series Analysis. In *Forty-first International Conference on Machine Learning.*
[38] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. *ICLR* (2023).
[39] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. *ICML* (2022).

A Adverse Effects of Channel-Mixing on Overfitting and Predictive Performance

Noise and interference in channel-mixing methods arise from the potential learning of *pseudo connections* when employing a Hypervariate Graph structure to capture dependencies. In particular, the Hypervariate Graph ($\mathbb{R}^{NT \times NT}$) is exponentially larger than our proposed learnable graphs, namely the Temporal Learnable Graph $A_T \in \mathbb{R}^{T \times T}$ and the Spatial Learnable Graph $A_S \in \mathbb{R}^{N \times N}$. Consequently, channel-mixing models (e.g., FourierGNN [34] and UniTST [18]) exhibit a higher degree of freedom compared to our channel-preserving model (GraphSTAGE). This broader flexibility can lead to numerous pseudo connections (connections between different nodes across different time steps) and diminish the weights of genuine inter-series and intra-series connections, often resulting in *overfitting* and *deteriorated predictive performance*.

To verify the hypothesis that channel-mixing can cause *overfitting* and *reduced predictive accuracy*, we conducted experiments on two datasets. Since the dependency-learning components in FourierGNN [34] and UniTST [18] are different from those in the proposed GraphSTAGE, we used the model variant **VarC** (see Figure 8) as the baseline. This variant replaces only the channel-preserving

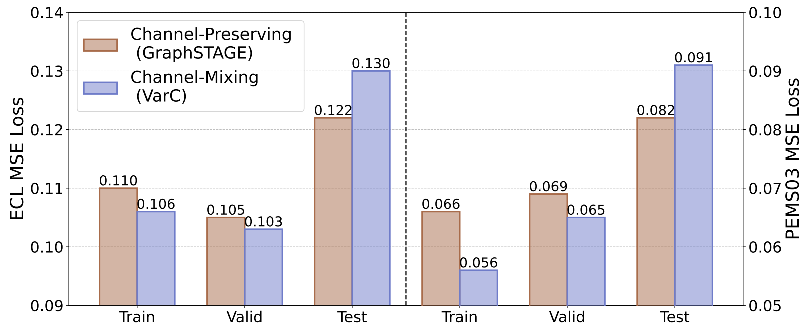

Figure 11: Comparison of training and test MSE. The channel-mixing model (VarC) shows lower training MSE but higher test MSE on the ECL dataset (left) and PEMS03 dataset (right), indicating overfitting.
Legend:
-GraphSTAGE: Channel-preserving model. -VarC: Channel-mixing variant.

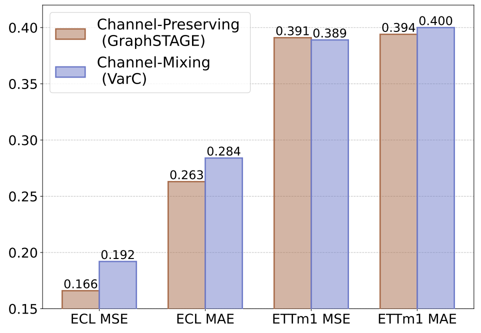

Figure 12: Average results over four prediction lengths. GraphSTAGE (channel-preserving) outperforms VarC (channel-mixing), validating that channel-mixing strategies can compromise accuracy.

strategy in the proposed GraphSTAGE with a channel-mixing strategy, while keeping other modules intact.

In our experiments, we trained for 30 epochs with a fixed learning rate of 1×10^{-3} on the ECL and PEMS03 datasets, both using an input length T equal to the prediction length K ($T = K = 24$). We ran each experiment five times, holding hyperparameters constant. As shown in Figure 11, the channel-mixing model (**VarC**) achieves lower training MSE but higher test MSE, revealing a clear sign of overfitting compared to the proposed channel-preserving model (**GraphSTAGE**).

To further demonstrate the decrease in predictive performance on the test set caused by the channel-mixing strategy, we summarize the average results across four prediction lengths on ECL and ETTm1 datasets. The input sequence length T is set to 96, and the average results are obtained from prediction lengths $K \in \{96, 192, 336, 720\}$. Figure 12 illustrate that the proposed channel-preserving model (**GraphSTAGE**) outperforms its channel-mixing variant (**VarC**), reinforcing our conclusion that channel mixing can degrade predictive performance.