MSCGrapher: Learning Multi-Scale Dynamic Correlations for Multivariate Time Series Forecasting

Xian Yang[1] **Zhenguo Zhang**[*1] **Shihao Lu**[1]

[1]Department of Computer Science and Technology, Yanbian University, 977 Gongyuan Road, Yanji, 133002, China

Abstract

Efficient learning intra-series and inter-series correlations is essential for multivariate time series forecasting (MTSF). However, in real-world scenarios, persistent and significant inter-series correlations are challenging to be represented in a static way and the strength of correlations varies across different time scales. In this paper, we address this challenge by modeling the complex inter-series relationships through dynamical correlations, considering the varying strengths of correlations. We propose a novel MTSF model: MSCGrapher, which leverages an adaptive correlation learning block to uncover inter-series correlations across different scales. Concretely, time series are first decomposed into different scales based on their periodicities. The graph representation of MTS is then constructed and an adaptive correlation learning method is introduced to capture the inter-series correlations across different scales. To quantify the strength of these correlations, we compute correlation scores based on the characteristics of the graph edges and classify correlations as either *Strong* or *Weak*. Finally, we employ a self-attention module to capture intra-series correlations and then fuse features from all scales to obtain the final representation. Extensive experiments on 12 real-world datasets show that MSCGrapher gains significant forecasting performance, highlighting the critical role of inter-series correlations in capturing implicit patterns for MTS.

1 INTRODUCTION

MTSF involves predicting the future based on multiple interrelated historical data, playing a significant role across various industries. Examples include predicting the prices of multiple assets in financial markets, multi-parameter weather in meteorology, equipment operation status in industrial manufacturing, and physiological indicators in healthcare [Chen et al., 2011, Wu et al., 2021, Fatima and Rahimi, 2024, Nguyen-Thai et al., 2024]. Due to its substantial applications, MTSF attracts widespread research interest. In the last decade, various deep learning models, such as methods based on CNNs [Zeng et al., 2023b, Wang et al., 2023], MLPs [Challu et al., 2023, Vijay et al., 2023], and Transformers [Zhou et al., 2021, Wu et al., 2021], have been proposed to tackle the challenges of time series forecasting and have achieved outstanding performance. Although these methods have different architectures, they fundamentally utilize neural networks to capture correlations: inter-series correlations and intra-series correlations [Cai et al., 2024].

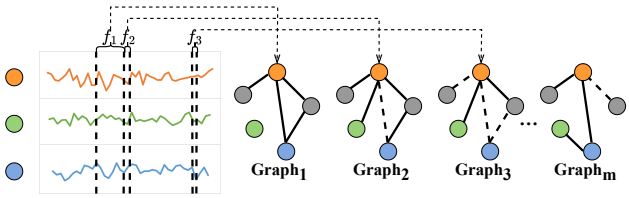


Figure 1: The relationships between different series vary at different time scales, resulting in different graph structures.

Early works primarily capture intra-series correlations but overlook inter-series correlations. This oversight significantly impacts the model's ability to capture complex dynamic relationships and prediction accuracy. In recent years, an increasing number of studies [Zhang and Yan, 2023, Yue et al., 2022] have focused on modeling inter-series correlations to reveal and leverage the complex interactions within MTS. One promising approach uses graph learning [Wu et al., 2020] to construct relationship graphs to model these correlations. While these methods can capture inter-series dependencies, they still have significant shortcomings in fully addressing the dynamically changing correlations across different time scales. Current MTSF method faces

[*]Corresponding author.

Proceedings of the 41st Conference on Uncertainty in Artificial Intelligence (UAI 2025), PMLR 244:4731–4751.

three limitations: (1) **Time Scale Insensitivity**: Most works primarily focus on single-scale correlation analysis, limiting their ability to reveal correlations across time scales and handle complex dynamic systems. For example, in climate science, climate change and extreme weather events are influenced by a combination of factors that exhibit different correlations at different time scales. The long-term global warming interacts with mid-term variations such as El Niño and short-term fluctuations like sudden extreme weather events (e.g., hurricanes, heavy rainfall) [Heede and Fedorov, 2023]. Figure 1 illustrates an example where a MTS is divided into three different time scales. Clearly, the relationships between nodes change at these scales. At scale f_1, the orange and blue series exhibit consistency. However, at scale f_3, they diverge, with the orange series affecting the green series, resulting in a different graph structure. From the above example, we can clearly identify the limitations of existing deep learning models in dynamic modeling of relationships for MTS. (2) **Dynamic Relationship Modeling**: While graph-based approaches represent MTS as nodes and their relationships as edges [Kipf and Welling, 2016], current graph structure learning in GNNs often lacks the adaptability needed to model dynamic inter-series correlations, particularly when these relationships evolve across different time scales. This rigidity limits their effectiveness in capturing the temporal evolution of complex systems. (3) **Correlation Strength Variability**: Another critical challenge lies in the scale-dependent variations in correlation strength, which significantly impact model performance. Existing methods often fail to account for these variations, further restricting their ability to accurately model and predict MTS behavior. These raise the question: Can graph learning accurately capture the correlations of MTS across different time scales? If so, what adjustments are needed for GNN's architecture?

To address the above issues, we propose MSCGrapher, which can effectively enhance graph learning's ability to capture dynamic varying correlations across various time scales in MTS, and accurately characterize the strength and weakness of correlations at different scales. First, MSCGrapher encodes the temporal variations of each series into a high-dimensional space and represents them as nodes in graph. For the multi-periodic characteristic of time series, we use Fast Fourier Transform (FFT) to extract periodic components at different frequencies, which reveals the underlying patterns and trends. Next, we design an adaptive correlation graph learning block that uses an adaptive GNN to dynamically learn adjacency matrices for each time scale. It computes relationship strength scores from edge features and partitions the matrices based on these scores to identify correlations and capture the complex dynamic changes in the data. For intra-series relationships, a multi-head attention module is employed to capture the dependencies at different time points by computing correlations between time steps. Finally, after multiple layers of feature aggregation, we generate the final prediction results. Our contributions include the following key aspects:

- **Overall Framework:** We propose the MSCGrapher framework, which effectively handles MTS and captures both multi-scale inter-series correlations and intra-series temporal correlations.
- **Effective Modules:** Our research shows that using an adaptive GNN can more accurately capture the complex dynamic changes hidden in MTS.
- **Performance:** Extensive experiments on various real-world datasets show that MSCGrapher outperforms existing models. Additionally, we perform transferability experiments with the correlation learning method, verifying its generalization capability across different datasets and models.

2 RELATED WORKS

2.1 TIME SERIES FORECASTING

Early time series forecasting are generally based on traditional statistical or machine learning methods. Recent advancements in deep learning architectures have shown significant advantages in time series forecasting [Miller et al., 2024]. CNNs have succeeded in MTSF, as seen in works like [Zeng et al., 2023b, Wang et al., 2023]. TCNs, a type of CNN that prevents future value leakage, effectively preserve the temporal order of time series [Bai et al., 2018]. MLPs encode temporal dependencies into their layers using the MLP structure [Vijay et al., 2023, Challu et al., 2023]. Transformers are used in MTSF due to their ability to capture long-range dependencies. Crossformer [Zhang and Yan, 2023] and Informer [Zhou et al., 2021] enhance model performance by employing cross-attention mechanisms and probabilistic sparse self-attention mechanisms to capture temporal dependencies. However, these methods fail to consider inter-series correlations at different time scales in MTS. While some methods address periodicity as a key factor in time series [Wu et al., 2022, Fan et al., 2022], they still fall short in modeling complex correlations and multi-scale dependencies.

2.2 CORRELATIONS LEARNING WITH GNNS

Graph Neural Networks (GNNs) demonstrate their importance in various fields by effectively modeling complex interactions in graph-structured data. Initially, GNNs were applied to tasks like traffic prediction [Wu et al., 2023] and skeleton-based action recognition [Shi et al., 2019]. In recent years, many studies start applying GNNs in MTS modeling to capture the dependencies between variables. These methods [Yu et al., 2017, Li et al., 2017] often use fixed graph structures to model inter-series correlations. For example,

in traffic prediction, a graph structure is constructed based on the spatial distance between sensors, with sensors as nodes and roads as edges connecting the nodes. However, constructing a graph structure based on prior knowledge is challenging in MTS modeling. To address this, researchers propose learnable graph structures to dynamically model relationships between series, offering new perspectives [Wu et al., 2020]. Recently, some approaches attempt to use dynamic or time-varying graph structures to model correlations [Zheng et al., 2020, Chen et al., 2023, Cai et al., 2024], but they overlook a key factor: as time progresses, the inter-series correlations change dynamically across different time scales, and the strength of these correlations fluctuates. Failure to adequately consider the varying strength of inter-series correlations leads to insufficient accuracy in capturing these important dependencies.

3 PRELIMINARIES

3.1 PROBLEM DEFINITION

Given MTS $\mathbf{X}_t = \{x_{t-L}, \ldots, x_{t-1}\} \in \mathbb{R}^{N \times L}$, where L denotes the size of the historical review window and N is the number of variables, the MTSF task is to predict the values of N variables over the future T time steps. The future values are denoted as $\mathbf{Y}_t = \{x_t, \ldots, x_{t+T-1}\} \in \mathbb{R}^{N \times T}$, where T denotes the size of the future prediction window. Here, $\mathbf{X}_{t,:} \in \mathbb{R}^N$ denotes the time series collected at time step t, and $\mathbf{X}_{:,n} \in \mathbb{R}^L$ represents the entire times series of each variable indexed by n.

3.2 GRAPH REPRESENTATION FOR MTS INTER-SERIES CORRELATIONS

We use graphs to represent the inter-series correlations of MTS at different scales, referred to as the strong correlation graph and the weak correlation graph. The graph is defined as $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ represents the set of nodes, $|\mathbf{V}| = N$ and $\mathbf{E}$ is the set of edges. We consider the i-th series as nodes v_i, and the weighted edges representing relationships between different time series are denoted by E_i. Strong correlation refers to variable pairs that exhibit consistently similar trends at the same time scale, with corresponding weights close to 1 in the learnable adjacency matrix. Weak correlation refers to dissimilar or noise-influenced trends, with weights close to 0. The strong correlation graph is denoted as G_{strong}, and the the weak correlation graph is denoted as G_{weak}. Different time scales identified from the MTS are represented as $f = \{f_1, \ldots, f_k\}$, assuming there are k different scales. The adjacency matrix corresponding to each scale is denoted by $\{\mathbf{A}^1, \ldots, \mathbf{A}^k\}$, where $\mathbf{A}^k \in \mathbb{R}^{N \times N}$. $\mathbf{A}^k_{\text{strong}}$ and $\mathbf{A}^k_{\text{weak}}$ represent the adjacency matrices of the strong correlation graph and weak correlation graph at scale k.

4 MSCGRAPHER

Our MSCGrapher, with residual connections, consists of an Embedding Layer, Multi-scale Correlation Learning Block (MSCL), Multi-head Attention Layer (MAL), Multi-scale Aggregation Layer, and Projection Layer. The Embedding Layer processes time series into suitable representations, MSCL and MAL capture inter-series correlations and intra-series dependencies. Finally, the multi-scale aggregation layer integrates features, and the projection layer outputs the final representation required for downstream tasks. The overall framework is illustrated in Figure 2.

4.1 TIME SERIES EMBEDDING REPRESENTATION

For each series of MTS, we treat it as a node of graph. The first step is to integrate the temporal dynamics of each series into a proper embedding representation.

Local features in time series reflect short-term changes and behaviors. We use 1D convolution to transform the input MTS into high-dimensional embedded representations: $\mathbf{emb}_{\text{Token}} = \text{Conv}_{1d}(\mathbf{X}_t, \mathbf{W})$, where $\mathbf{emb}_{\text{Token}} \in \mathbb{R}^{c_{\text{dim}} \times L}$, c_{dim} is the feature dimension and $\mathbf{W}$ is the weight matrix. Additionally, temporal features often contain important information that explains periodicity, trends, and other time-related patterns. We employ an embedding operation to enhance temporal context information: $\mathbf{emb}_{\text{Temporal}} = \mathbf{E}_m + \mathbf{E}_d + \mathbf{E}_w + \mathbf{E}_h + \mathbf{E}_t$, where $\mathbf{E}_{i \in \{m,d,w,h,t\}} \in \mathbb{R}^{c_{\text{dim}} \times L}$ represents embeddings for month, day, week, hour, and minute. In the forecasting scenario, position features are also crucial. Therefore, positional information is added in the series through position encoding: $\mathbf{emb}_{\text{Position}} = \text{PE}(L, i)$, where $i \in \{0, c_{\text{dim}} - 1\}$ represents the index of the dimension in the embedding vector. In summary, Embedding layer of MSCGrapher consists of three main parts:

$$\mathbf{H}_{\text{emb}} = \mathbf{emb}_{\text{Token}} + \mathbf{emb}_{\text{Temoporal}} + \mathbf{emb}_{\text{Position}} \quad (1)$$

4.2 MULTI-SCALE INTER-SERIES CORRELATIONS LEARNING

To effectively capture the correlations of MTS at different time scales, we design a Multi-scale Correlation Learning Block (named MSCL), which consists of Multi-scale Segmentation Layer and Adaptive Correlation Graph Learning Layer. The former divides time series into different time scales based on their periodic characteristics while the latter learns the dependencies between time series at the corresponding scales to capture correlations. k parallel blocks are used to learn correlations of k time scales.

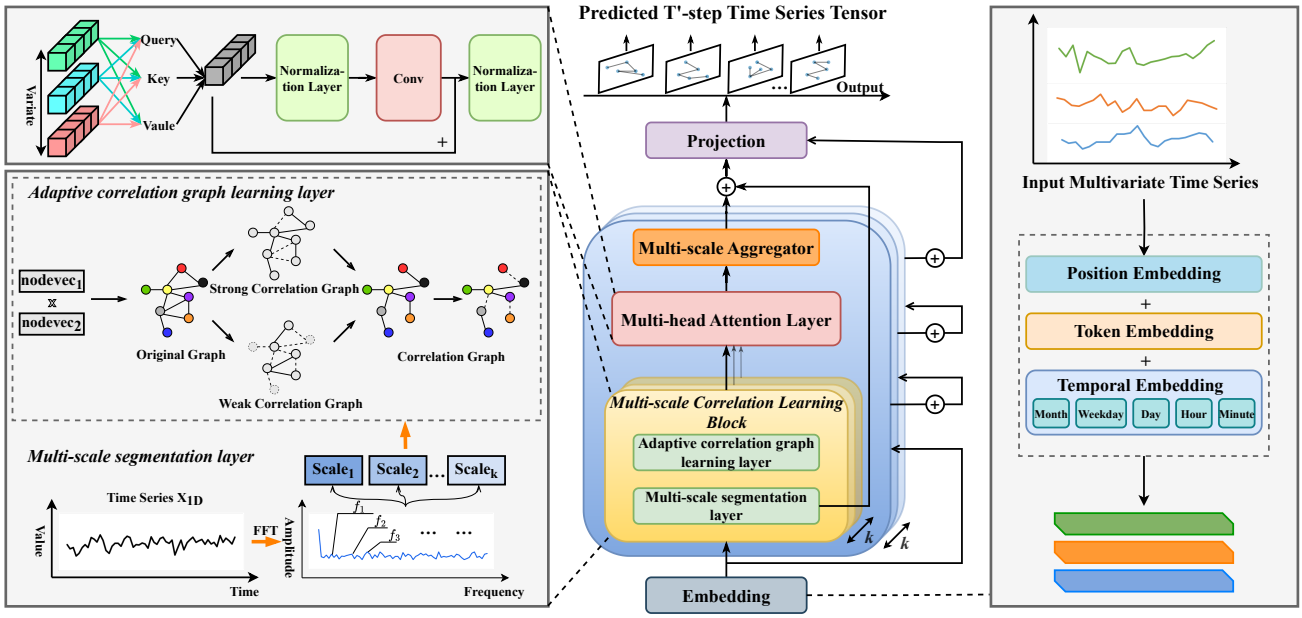


Figure 2: The overall framework of MSCGrapher. The core is the Multi-scale Correlation Learning Block, which includes the Multi-scale Segmentation Layer and Adaptive Correlation Graph Learning Layer.

4.2.1 Multi-scale Segmentation of MTS

Generally, different scales uncover various patterns. For example, in financial markets, short-term price fluctuations may be influenced by news events and trading behaviors, while long-term trends may be driven by economic cycles and policy. To identify periodicities of time series as time scales, we transform the representation of MTS to frequency domain by using Fast Fourier Transform (FFT). The process is as follows:

$$\begin{aligned} \mathbf{X}_f &= \mathrm{FFT}(\mathbf{H}_{\mathrm{emb}}), \quad \mathbf{F} = \mathrm{Avg}(\mathrm{Amp}(\mathbf{X}_f)), \\ &\underset{f_* \in \{1,\ldots,\lfloor \frac{L}{2} \rfloor\}}{\mathrm{argTopk}}(\mathbf{F}) = \{f_1, \ldots, f_k\}, \quad p_i = \frac{L}{f_i}. \end{aligned} \tag{2}$$

where $\mathrm{FFT}()$ and $\mathrm{Amp}()$ represent FFT and amplitude calculations, p_i is the period corresponding to different scales. We first extract the k most significant frequency components $\{f_1, \ldots, f_k\}$, and compute their corresponding period $\{p_1, \ldots, p_k\}$. Then, we reshape the original input $\mathbf{X}_{\mathrm{input}}$ based on the extracted period p_i and frequency f_i:

$$\mathbf{X}^i = \mathrm{Reshape}_{p_i, f_i}(\mathrm{Padding}(\mathbf{X}_{\mathrm{input}})), \tag{3}$$

where $\mathrm{Padding}()$ extends the time series with zero padding along the time dimension to fit $\mathrm{Reshape}_{p_i,f_i}()$, $i \in \{1, \ldots, k\}$. Note that $\mathbf{X}^i \in \mathbb{R}^{c_{\mathrm{dim}} \times p_i \times f_i}$ represents the i-th reshaped time series for time scale i.

4.2.2 Adaptive Correlation Graph Learning

Two trainable matrices, $\mathbf{E}_1^l \in \mathbb{R}^{c \times N}$ and $\mathbf{E}_2^l \in \mathbb{R}^{N \times c}$, are employed to learn the adaptive adjacency matrix at time scale l:

$$\mathbf{A}^l = \mathrm{SoftMax}\left(\mathrm{ReLU}\left(\mathbf{E}_1^l (\mathbf{E}_2^l)^T\right)\right), \tag{4}$$

i.e., we learn a new adjacency matrix at each time scale to capture differences in correlations across different scales. After obtaining the l-th time scale adjacency matrices $\mathbf{A}^l$, we can generate new adjacency matrices $\mathbf{A}^l_{\mathrm{strong}}$ and $\mathbf{A}^l_{\mathrm{weak}}$ based on the changes in correlation strength. The process is illustrated in Figure 3.

We first construct the edge index matrix $\mathbf{E}^l_{\mathrm{index}}$ and edge attribute matrix $\mathbf{A}^l_{\mathrm{edge}}$ based on $\mathbf{A}^l$:

$$\begin{aligned} \mathbf{E}^l_{\mathrm{index}} &= \mathrm{Transpose}(\mathrm{Nonzero}(\mathbf{A}^l)), \\ \mathbf{A}^l_{\mathrm{edge}} &= \mathrm{Reshape}_c(\mathbf{A}^l(\mathbf{A}^l \neq 0)), \end{aligned} \tag{5}$$

Then, we obtain graph representation by two parameters:

$$\begin{aligned} \widehat{\mathbf{X}} &= \mathrm{Reshape}_{L, c_{\mathrm{dim}}}(\mathbf{X}_{\mathrm{input}}), \\ \mathbf{G}_{\mathrm{data}} &= \mathrm{Data}(\widehat{\mathbf{X}}, \mathbf{E}^l_{\mathrm{index}}, \mathbf{A}^l_{\mathrm{edge}}, \mathbf{B}^l). \end{aligned} \tag{6}$$

where $\mathbf{B}^l$ is a zero-filled batch tensor, $\mathrm{Data}()$ is used to create a graph object.

Next, we apply convolution and non-linear transformations on the node features in $\mathbf{G}_{\mathrm{data}}$ to extract the start and end node indices, denoted as r and c, respectively, from $\mathbf{E}^l_{\mathrm{index}}$,

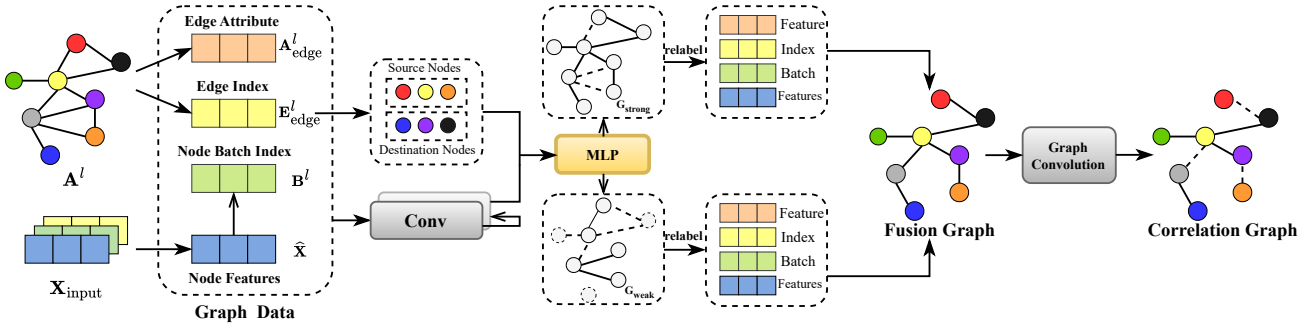


Figure 3: An overview of the Adaptive Correlation Graph Learning Layer. It utilizes a correlation graph learning method to obtain the strong and weak correlations between time series.

where r corresponds to the source node and c to the target node. We then obtain representations for each edge and compute edge scores using an MLP:

$$\begin{aligned} \widehat{\mathbf{G}}_{\text{data}} &= \text{Conv}_2(\text{ReLU}(\text{Conv}_1(\mathbf{G}_{\text{data}}))), \\ \text{r} = \mathbf{E}^l_{\text{index}}[0,:], \text{c} &= \mathbf{E}^l_{\text{index}}[1,:], \\ \mathbf{E}_{\text{rep}} = \text{concat}(\widehat{\mathbf{X}}[\text{r}], \widehat{\mathbf{X}}[\text{c}]), &\mathbf{S}_{\text{edge}} = \text{MLP}(\mathbf{E}_{\text{rep}}). \end{aligned} \tag{7}$$

We concatenate the node features corresponding to the start and end node indices to obtain $\mathbf{E}_{\text{rep}}$ and use $\text{MLP}()$ to compute edge scores $\mathbf{S}_{\text{edge}}$. After obtaining the score for each edge, a higher score indicates a stronger relationship between the two nodes. Specifically, a higher score means that the edge is more important in the graph structure, and the correlations between two nodes is stronger. We partition the edges of the graph into strong correlation graph $\mathbf{G}_{\text{strong}}$ and weak correlation graph $\mathbf{G}_{\text{weak}}$ based on the edge scores and a ratio.

Then, we relabel the nodes of $\mathbf{G}_{\text{strong}}$ and $\mathbf{G}_{\text{weak}}$ to obtain new node features:

$$\begin{aligned} \mathbf{G}^l_{\text{strong}}, \mathbf{G}^l_{\text{weak}} &= \text{Divide}(\mathbf{S}_{\text{edge}}, \text{ratio}), \\ \mathbf{A}^l_{\text{strong}}, \mathbf{A}^l_{\text{weak}} &= \text{Relabel}(\mathbf{G}^l_{\text{strong}}, \mathbf{G}^l_{\text{weak}}), \end{aligned} \tag{8}$$

where $\text{Relabel}()$ represents the function for relabeling node features. After that, we assign corresponding weights to the strong and weak adjacency matrices, perform a weighted sum, and fuse the two graphs to generate a new representation $\mathbf{A}^l_f$.

$$\mathbf{A}^l_f = W \cdot \mathbf{A}^l_{\text{strong}} + (1 - W) \cdot \mathbf{A}^l_{\text{weak}}, \tag{9}$$

where W is a weighting parameter used to control the fusion ratio of $\mathbf{A}^l_{\text{strong}}$ and $\mathbf{A}^l_{\text{weak}}$. We assign a higher weight to the $\mathbf{A}^l_{\text{strong}}$ to direct the model's attention toward strong correlation information, thereby capturing key node relationships more effectively while avoiding noise interference caused by weak correlation information. Finally, we use the Mix-hop graph convolution method [Abu-El-Haija et al., 2019] to capture the dependencies in the Fusion graph between time series:

$$\mathbf{X}^i_{\text{out}} = \sigma\left(\Big\Vert_{j \in P} (\mathbf{A}^l_f)^j \mathbf{X}^i \right). \tag{10}$$

where, $\mathbf{X}^i_{\text{out}}$ is the output after fusion at scale i. The hyper-parameter P is a set of integers representing the powers of the adjacency matrix. $(\mathbf{A}^l_f)^j$ denotes the j-th power of the learned Fusion adjacency matrix $\mathbf{A}^l_f$, and $\Vert$ concatenates the intermediate results generated in each iteration along the column direction. Finally, we use an MLP to project $\mathbf{X}^i_{\text{out}}$ into a 3D tensor $\widehat{\mathbf{X}}^i_{\text{out}} \in \mathbb{R}^{c_{\text{dim}} \times p_i \times f_i}$.

4.3 EXTRACTION OF INTRA-SERIES CORRELATIONS

A multi-head attention based module is proposed to capture the intra-series correlations within time series at different time scales. Specifically, we project the input series $\widehat{\mathbf{X}}^i_{\text{out}}$ through a linear layer into different spaces to obtain queries($\mathbf{Q}$), keys($\mathbf{K}$), and values($\mathbf{V}$). They are then projected onto multiple attention heads, where each head learns different temporal dependencies. Finally, we combine the outputs of different heads and extract local features through Conv_{1d} to generate a comprehensive representation $\widehat{\mathbf{H}}^i_{\text{out}} \in \mathbb{R}^{Bf_i \times c_{\text{dim}} \times p_i}$, where B is the batch size:

$$\begin{aligned} \mathbf{O} &= \text{Linear}\left(\text{Concat}\left(\text{head}_1, \ldots, \text{head}_H\right)\right), \\ \mathbf{X}_{\text{attn}} &= \text{LayerNorm}\left(\widehat{\mathbf{X}}^i_{\text{out}} + \text{Dropout}(\mathbf{O})\right), \\ \mathbf{Y} &= \text{Dropout}\left(\text{Conv}_{1d}\left(\text{Conv}_{1d}(\mathbf{X}^T_{\text{attn}})^T\right)^T\right), \\ \widehat{\mathbf{H}}^i_{\text{out}} &= \text{LayerNorm}\left(\mathbf{X}_{\text{attn}} + \mathbf{Y}\right). \end{aligned} \tag{11}$$

4.4 MULTI-SCALE AGGREGATOR AND PROJECTION

After handling k scales, we obtain the representations $\widehat{\mathbf{H}}^i_{\text{out}}$ for each scale. To generate predictions through node regression, we need to aggregate the tensors from the k different scales. Each tensor is first reshaped to obtain new $\widehat{\mathbf{H}}^i_{\text{out}} \in \mathbb{R}^{c_{\text{dim}} \times L}$, and then k scales based on their respective amplitudes are aggregated:

$$\widehat{\mathbf{X}}_{\text{out}} = \sum_{i=1}^{k} \text{Softmax}(\mathbf{W})\widehat{\mathbf{H}}^i_{\text{out}}, \tag{12}$$

$\mathbf{W} \in \mathbb{R}^{B \times k}$ is the learnable scale weight matrix composed of amplitudes from each time scale, which represents the relative importance. Thus, we can adaptively integrate the information from different scales based on the learned weights. The final prediction is completed by a regression process:

$$\mathbf{Y}_t = \mathbf{W}_l \widehat{\mathbf{X}}_{\text{out}} \mathbf{W}_t + \mathbf{b}. \tag{13}$$

where $\mathbf{W}_l \in \mathbb{R}^{N \times c_{\text{dim}}}$ and $\mathbf{W}_t \in \mathbb{R}^{L \times T}$ are learnable weights. $\mathbf{W}_l$ and $\mathbf{W}_t$ perform linear mapping on the variable dimension and time dimension, respectively.

5 EXPERIMENTS

We conduct a comprehensive experiments of MSCGrapher on MTSF across multiple real-world datasets to validate its generalization ability in various scenarios. We also explore the potential of integrating correlation learning block into other models to assess their transferability and performance.

5.1 EXPERIMENT SETUP

5.1.1 Datasets and Baselines

12 real-world MTS datasets are employed, including Flight, ETT(h1,h2,m1,m2)[Wu et al., 2022], Weather, Electricity, Exchange-Rate[Lai et al., 2018] and PEMS(03,04,07,08)[Liu et al., 2022a]. 13 well-established forecasting models are selected as baselines, including (1) Transformer-based models: Informer [Zhou et al., 2021], Autoformer [Wu et al., 2021], Pyraformer [Liu et al., 2021], FEDformer [Zhou et al., 2022], and Stationary [Liu et al., 2022b]; (2) Linear methods: TiDE [Das et al., 2023] and Dlinear [Zeng et al., 2023a]; (3) TCN-based methods: TimesNet [Wu et al., 2022] and MSGNet [Cai et al., 2024]; (4) GNN-based methods: MSHyper [Shang and Chen, 2024], CrossGNN [Huang et al., 2023], StemGNN [Cao et al., 2020] and FourierGNN [Yi et al., 2024].

5.1.2 Implementation details

All experiments are conducted on an NVIDIA GeForce RTX 4090 24GB GPU. We use the Adam optimizer with a learning rate set to 10^{-4} and a batch size of 32. The loss function is MSE. We set the historical review window L to 96 and the forecasting window T to {96, 192, 336, 720} or {12, 24, 48, 96}.

5.2 FORECASTING RESULTS AND ANALYSIS

We present the forecasting results in Table 1, which compare the MSE and MAE across all output lengths with 9 non GNN-based baselines. The best results are highlighted in red **bold** and the second best results are underlined in blue. Compared to other models, MSCGrapher wins 13 times across various frequencies, numbers of variables, and real-world scenarios, while the second baselines only win 7 times. To assess the model's generalization ability, we also calculate the average rank, where MSCGrapher gains 1.50 and consistently outperforms other models. Compared to Transformer-based methods, MSCGrapher has a significant performance improvement, which demonstrate that inter-series relationships cannot be ignored for MTS. Although linear methods are advantageous for long-term forecasting, MSCGrapher still achieved performance improvement on most datasets. Compared to the latest SOTA model (MSGNet), which also leverages multi-scale information, MSCGrapher achieves superior performance across all datasets. For example, on the Flight dataset, MSCGrapher reduces the MSE by 3.4%; on the ETTh datasets, the MSE drops by 4.8% and 2.2%, respectively; and on the PEMS dataset, the improvements are 1.5% and 7.3%. The Flight dataset contains highly volatile air traffic data with frequent short-term fluctuations and certain periodic patterns. By integrating both strong and weak correlations, MSCGrapher enhances forecasting accuracy. The ETTh dataset exhibits clear seasonal trends and periodic fluctuations—strong correlations capture the inertial behavior of power loads, while weak correlations reflect contextual factors such as ambient temperature and temporal cycles. MSCGrapher effectively distinguishes and adaptively fuses correlations of different strengths, resulting in lower forecasting errors. In the PEMS datasets, traffic flow is characterized by spatial heterogeneity and abrupt temporal changes, making inter-node relationships particularly complex. MSCGrapher demonstrates its advantage by modeling the correlations of dynamic variations.

Additionally, we compare MSCGrapher with GNN-based methods using datasets consistent with the baselines. The detailed results are shown in Table 2 and 3. As shown in Table 2, MSCGrapher significantly outperforms GNN-based methods across five datasets from different domains, especially on three traffic-related datasets, where the average MSE decreases by 18.6%, 30.38%, and 23.87%, respectively. Ta-

Table 1: The forecasting results of our MSCGrpaher and baselines. Complete results are referred to Supplements.

Models	MSCGrpaher		MSGNet(2024)		Dlinear(2023)		TimesNet(2023)		TiDE(2023)		Stationary(2022)		FEDformer(2022)		Pyraformer(2022)		Autoformer(2021)		Informer(2021)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Flight	**0.201**	**0.315**	0.208	0.321	0.233	0.345	0.265	0.372	0.239	0.350	0.560	0.542	0.418	0.485	0.448	0.496	0.238	0.344	0.391	0.439
ETTh1	**0.450**	**0.452**	0.472	0.477	0.456	0.452	0.458	0.452	0.518	0.517	0.570	0.537	0.498	0.484	0.827	0.703	0.496	0.487	1.040	0.820
ETTh2	0.393	**0.415**	0.402	0.431	0.559	0.515	0.414	0.427	**0.387**	0.419	0.526	0.516	0.437	0.449	0.826	0.703	0.450	0.459	4.431	1.729
ETTm1	**0.400**	0.412	0.400	0.412	0.403	0.407	0.400	**0.406**	0.413	0.415	0.471	0.456	0.448	0.452	0.618	0.607	0.588	0.517	0.961	0.734
ETTm2	**0.287**	**0.329**	0.290	0.331	0.350	0.401	0.291	0.333	0.293	0.338	0.306	0.347	0.305	0.349	1.498	0.869	0.327	0.371	1.410	0.810
weather	**0.255**	**0.283**	0.257	0.284	0.265	0.317	0.259	0.287	0.271	0.320	0.288	0.314	0.309	0.360	0.815	0.717	0.338	0.382	0.634	0.548
electricity	0.196	0.302	0.196	0.303	0.212	0.300	**0.193**	**0.295**	0.209	0.295	0.193	0.296	0.214	0.327	0.382	0.445	0.227	0.338	0.336	0.397
exchange	0.396	0.429	0.403	0.432	**0.354**	**0.414**	0.416	0.443	0.400	0.431	0.461	0.454	0.519	0.500	1.377	1.018	0.613	0.539	1.550	0.998
PEMS03	**0.136**	**0.241**	0.138	0.243	0.278	0.375	0.147	0.248	0.326	0.420	0.147	0.249	0.213	0.327	0.360	0.414	0.667	0.601	0.201	0.300
PEMS08	**0.192**	0.276	0.206	0.323	0.379	0.416	0.193	**0.271**	0.441	0.464	0.201	0.276	0.286	0.358	0.269	0.292	0.814	0.659	0.313	0.325
1st Count	**13**		0		2		4		1		0		0		0		0		0	
Avg Rank	**1.50**		2.60		4.95		2.90		5.30		5.60		6.25		8.85		7.35		8.75	

Table 2: Forecasting results compared with GNN methods.

Datasets	Electricity		Weather		PEMS03		PEMS04		PEMS08	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MSCGrpaher	**0.196**	0.302	0.255	**0.283**	**0.136**	**0.241**	**0.137**	**0.255**	**0.192**	**0.276**
FourierGNN	0.228	0.325	**0.249**	0.302	0.151	0.267	0.180	0.295	0.216	0.313
StemGNN	0.197	**0.300**	0.268	0.321	0.187	0.302	0.217	0.333	0.303	0.351

Table 3: GNN methods results on PEMS datasets.

Dataset		PEMS03		PEMS04		PEMS07		PEMS08	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MSCGrpaher	12	**0.078**	**0.184**	**0.092**	0.207	**0.070**	**0.175**	**0.116**	**0.223**
	24	**0.103**	0.214	**0.109**	0.228	**0.093**	**0.195**	**0.149**	**0.255**
	48	**0.151**	**0.257**	**0.144**	**0.265**	**0.125**	**0.234**	**0.189**	**0.275**
MSHyper	12	0.106	0.207	0.103	**0.197**	0.137	0.256	0.113	0.209
	24	0.126	**0.207**	0.148	**0.148**	0.245	0.225	0.230	0.248
	48	0.138	0.265	0.191	0.308	0.137	0.221	0.317	0.324
CrossGNN	12	0.094	0.208	0.158	0.270	0.085	0.198	0.148	0.262
	24	0.131	0.248	0.231	0.322	0.185	0.293	0.277	0.363
	48	0.242	0.343	0.468	0.475	0.340	0.414	0.336	0.407
FourierGNN	12	0.087	0.202	0.112	0.231	0.073	0.182	0.143	0.263
	24	0.120	0.240	0.153	0.272	0.100	0.215	0.210	0.320
	48	0.177	0.294	0.209	0.321	0.140	0.258	0.216	0.311

ble 3 further compares MSCGrapher with state-of-the-art GNN methods on the PEMS datasets, where MSCGrapher achieves superior performance on most metrics. Due to the pronounced spatial heterogeneity and temporal abrupt changes in traffic flow within the PEMS datasets, the relationships among nodes are highly complex and dynamic. This fully demonstrates that the inter-series correlations in multivariate time series evolve over time, and our multi-scale correlation learning method effectively captures and handles these dynamic features.

To more clearly show the capability of MSCGrapher in modeling the inter-series correlations of MTS, we illustrate the forecasting results of a single-variate series on flight dataset with 5 baselines on Figure 4. The significant prediction deviations are marked with circles and yellow shaded areas. As observed, MSCGrapher fits nearly all key change regions well, whereas other baselines struggle in scenarios involving drastic changes. Specifically, MSCGrapher demonstrates better visualization of prediction results than MSGNet in the low-value range of 20–100. Moreover, within the peak range of 20–40, MSCGrapher more accurately captures the dynamic trends of the true value curve. MSCGrapher also outperforms DLinear in predicting certain extreme points, producing results closer to the true values. In contrast, DLinear tends to exhibit lag or smoothing effects when handling sharp fluctuations, making it difficult to precisely capture sudden changes. Furthermore, MSCGrapher surpasses TiDE in predicting extreme points and fitting trends within the 30–40 and 50–60 intervals, demonstrating higher fitting accuracy and improved dynamic consistency. Pyraformer performs poorly during peak periods and in regions with significant fluctuations, failing to accurately track pronounced changes in the data. Autoformer underperforms in low-value regions, struggling to capture subtle variations and resulting in significant deviations between predicted and true values.

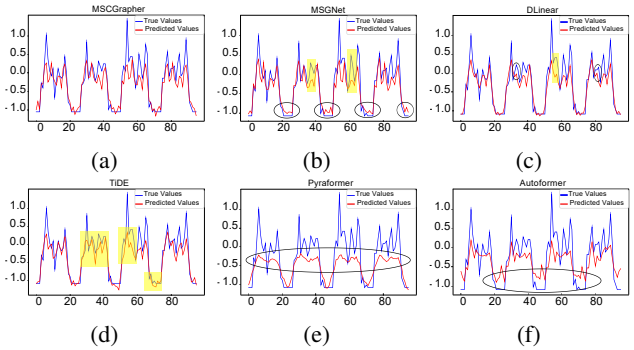


Figure 4: Visualization of the prediction results on the flight dataset with an output length of 96.

5.3 ABLATION STUDY

To investigate the impact of different modules in MSCGrapher, we design the following variants:

1. w/o-ACLayer: Adaptive correlation learning layer is instead by convolutions.
2. w/o-CorGraph: The process of correlation graph learning does not have a strong or weak degree division.
3. w/o-MSLayer: It removes the multi-scale modeling part.
4. w/o-Attention: The Multi-Head self-Attention Layer is removed.

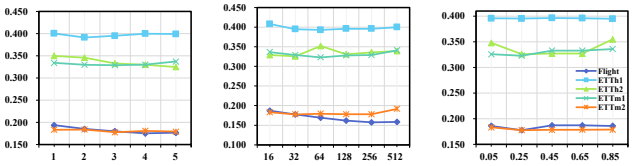
We do the ablation study on 5 datasets which are from different domain. Table 4 shows the average results of these variants across output lengths. From Table 4, we can find theses variants all have a increase in MSE and MAE. When removing the adaptive correlation learning layer, the performance is most affected on all datasets. On Flight and PEMS08, MSE drops by 35.33% and 24.41%, and MAE by 19.30% and 22.76%, which demonstrate that the inter-series correlations is a key factor for MTSF task. Furthermore, the results of variant 2 also indicate that manipulating the strength of the inter-series correlations in different ways can more accurately capture the implicit information. When using a single scale instead of multi-scale partitioning, MSCGrapher has a significant performance degradation, where MSE drops by 12.02% and MAE by 8.07% on the Flight dataset. It proves that the periodicity is a core characteristic and multi-scale helps to extract the complex periodic patterns hidden in MTS. The results of variant 4 show that intra-series correlations are also a crucial factor.

Table 4: Results of the ablation study.

Datasets	Flight		ETTh1		Exchange		Weather		PEMS08	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MSCGrapher	**0.201**	**0.315**	**0.450**	**0.452**	**0.393**	**0.429**	**0.255**	**0.283**	**0.192**	**0.276**
w/o-ACLayer	0.317	0.413	0.500	0.478	0.430	0.448	0.265	0.292	0.254	0.342
w/o-CorGraph	0.208	0.320	0.465	0.460	0.417	0.442	0.257	0.284	0.204	0.277
w/o-MSLayer	0.233	0.347	0.467	0.461	0.399	0.429	0.256	0.284	0.198	0.283
w/o-Attention	0.206	0.320	0.455	0.455	0.412	0.436	0.257	0.284	0.198	0.281

5.4 SENSITIVITY TO HYPERPARAMETERS

We evaluate the impact of following hyperparameters on different datasets: scale k, embedding dimension c_{dim} and ratio. In this experiments, the length of historical review window and the prediction length are set to 96. The results are presented in Figure 5. For k, we can find the MSE gradually decreases on all datasets as k increases. When k increases to a certain extent, the performance begins to decline. For these datasets, the best choice of k is 3 or 5, which indicates that MTS can be represented well from several period in most cases. Similar to k, c_{dim} also presents the same trend. Concerning the ratio, although increasing the ratio helps capture more potential strong correlations, it also raises the risk of misinterpreting weak correlations as strong correlations, which may affect the overall performance of the model.



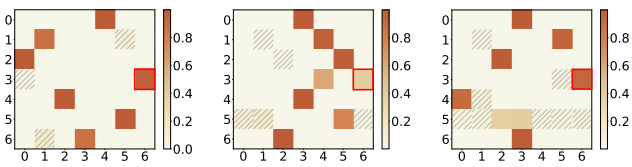
(a) k (b) c_{dim} (c) ratio

Figure 5: Sensitivity tests for k, c_{dim}, and ratio.

5.5 LEARNED CORRELATION GRAPH VISUALIZATION

To clearly exhibit the learned correlations, we visualize a subset of the correlation adjacency matrices from the Flight dataset in Figure 6.

Specifically, we choose three different time scales: 24h, 12h, and 6h, and compare the different correlation adjacency matrices for each time scale. As shown in Figure 6, the correlation adjacency matrices are sparse on all time scales, which says that our method can effectively find the intrinsic relationships hidden in MTS. At different time scales, we observe that the strength of correlations changes. For example, the correlation between nodes 3 and 6 (marked in red boxes) is strong at 24h and 6h. However, at a 12h time scale, the correlation becomes weak. This suggests that the variation in correlations across different time scales reveals the complex dynamic evolution of the system. If we can focus more on the strong correlations and reduce the influence of weak correlations during the evolution process, we can more effectively capture the core dynamics and key changes of the system. By increasing the weight of strong correlations, we can direct the model's attention to relationships that remain consistently stable across multiple time scales, rather than being distracted by weak correlations that only exist briefly at specific time scales.



(a) $24h$ (b) $12h$ (c) $6h$

Figure 6: Visualization of the first layer correlation adjacency matrix on Flight dataset for different time scales: (a) 24h, (b) 12h, and (c) 6h.

Table 5: Comparative results of integrating our correlation learning methods into MTGNN and TEGNN.

Datasets		Solar-Energy				Traffic				Electricity				Exchange-Rate			
Model	Metric	3(30min)	6(60min)	12(120min)	24(240min)	3(3h)	6(6h)	12(12h)	24(24h)	3(3h)	6(6h)	12(12h)	24(24h)	3(3d)	6(6d)	12(12d)	24(24d)
MTGNN	RSE	0.1778	0.2348	0.3109	0.4270	0.4162	0.4754	0.4461	0.4535	0.0745	0.0878	0.0916	0.0953	0.0194	0.0259	0.0349	0.0456
	RSE(Cor)	0.1782	0.2362	**0.3102**	**0.4230**	0.4237	**0.4688**	0.4531	**0.4531**	**0.0744**	**0.0856**	**0.0911**	**0.0950**	**0.0192**	**0.0258**	**0.0344**	**0.0446**
	CORR	0.9852	0.9726	0.9509	0.9031	0.8963	0.8667	0.8794	0.8810	0.9474	0.9316	0.9278	0.9234	0.9786	0.9708	0.9551	0.9372
	CORR(Cor)	0.9851	0.9723	**0.9512**	**0.9058**	0.8934	**0.8670**	0.8764	**0.8818**	**0.9475**	**0.9327**	0.9273	**0.9237**	**0.9789**	0.9705	**0.9533**	0.9362
TEGNN	RSE	0.1824	0.2612	0.3289	0.4733	0.4421	0.4433	0.4508	0.4692	0.0774	0.0862	0.0948	0.0985	0.0178	0.0245	0.0363	0.0449
	RSE(Cor)	**0.1739**	**0.2298**	**0.2943**	**0.3942**	**0.4178**	0.4505	**0.4414**	**0.4495**	**0.0748**	**0.0862**	**0.0938**	**0.0965**	**0.0177**	0.0255	**0.0348**	**0.0507**
	CORR	0.9847	0.9676	0.9379	0.8854	0.8853	0.8820	0.8743	0.8671	0.9418	0.9310	0.9225	0.9182	0.9815	0.9732	0.9566	0.9352
	CORR(Cor)	**0.9856**	**0.9742**	**0.9572**	**0.9183**	**0.8978**	0.8792	**0.8830**	**0.8781**	**0.9460**	**0.9531**	**0.9250**	**0.9201**	**0.9817**	**0.9732**	**0.9588**	**0.9385**

Table 6: Model efficiency comparison on Electricity with input length 96 and output length 96.

Models	Pred Length	GPU Memory (GB)	Training Time (ms/iter)	MSE Rank
MSCGrapher	96	6.48	404 ms	1
MSGNet	96	6.55	281 ms	4
TimesNet	96	5.81	532 ms	3
DLinear	96	1.38	15 ms	5
FourierGNN	96	21.57	434 ms	6
StemGNN	96	7.25	186 ms	2

5.6 TRANSFERABILITY OF CORRELATION LEARNING

In this section, we integrate the correlation learning method into GNN-based models: MTGNN [Wu et al., 2020] and TEGNN [Duan et al., 2022], to validate its transferability. For ease of comparison, experiments are conducted on same MTS datasets: Solar-Energy, Traffic, Electricity, and Exchange-Rate. Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) are used for evaluation.

Table 5 shows the comparative results before and after integrating our correlation learning method into MTGNN and TEGNN. (Cor) indicates the incorporation of our proposed correlation learning method into the respective models. From Table 5, it is evident that both models gain performance improvements in the majority of scenarios, especially for TEGNN on Solar-Energy and Electricity datasets. After replacing the original correlation learning, MTGNN shows slight improvements across all horizons on the smaller datasets, Electricity and Exchange. However, in the other two larger datasets, the performance improvement is more noticeable for long-series predictions. TEGNN exhibits significant improvements in the first three datasets, which is likely because these datasets are collected on an hourly or minute basis, containing more time series information. With the replace of correlation learning, TEGNN has better ability to capture the complex inter-series relationships. This indicates that our correlation learning method can effectively extract inter-series dynamic correlations from MTS.

5.7 COMPUTATIONAL EFFICIENCY

For efficiency evaluation, we select the more complex Electricity dataset to conduct a comprehensive comparison of GPU memory usage, running speed, and MSE ranking across different models under the prediction length of 96. This approach allows us to systematically assess the trade-off between accuracy and computational efficiency. To ensure fairness, all models were tested under the same conditions. The detailed results are presented in Table 6.

In Table 6, although MSCGrapher is not the best among all models in terms of training speed and GPU memory usage, it strikes a good balance between overall efficiency and performance. Specifically, MSCGrapher maintains a controllable level of resource consumption while achieving significantly better prediction accuracy than models such as FourierGNN, demonstrating strong modeling capability. Although TimesNet shows advantages in memory usage, its noticeably slower training speed hampers overall training efficiency. Considering both accuracy and computational cost, MSCGrapher exhibits stable and superior performance.

6 CONCLUSION

In this study, we propose a novel MTSF model, MSCGrapher, which starts from the premise that different relationships exist in MTS at various scales. MSCGrapher effectively captures both inter-series correlations with varying strengths and intra-series temporal correlations at different time scales by combining multi-scale correlation learning block with multi-head self-attention. Extensive experiments on real-world datasets shows that MSCGrapher outperforms existing models. When introduced our adaptive correlation learning method, two GNN-based methods also gain better performance, which proves that modeling the dynamic varying correlations is helpful for MTSF task.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China [grant number 62162062] and the Science and Technology Development Plan Project of Jilin Province [20220203127SF] .

References

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.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arxiv. arXiv preprint arXiv:1803.01271, 10, 2018.

Wanlin Cai, Yuxuan Liang, Xianggen Liu, Jianshuai Feng, and Yuankai Wu. Msgnet: Learning multi-scale inter-series correlations for multivariate time series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 11141–11149, 2024.

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.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In Proceedings of the AAAI conference on artificial intelligence, pages 6989–6997, 2023.

Cathy W. S. Chen, Richard Gerlach, Wcw Lee, and Edward M. H. Lin. Bayesian forecasting for financial risk management, pre and post the global financial crisis. Working Papers, 31(8):661–687, 2011.

Ling Chen, Donghui Chen, Zongjiang Shang, Binqing Wu, Cen Zheng, Bo Wen, and Wei Zhang. Multi-scale adaptive graph neural network for multivariate time series forecasting. IEEE Transactions on Knowledge and Data Engineering, 35(10):10748–10761, 2023.

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

Ziheng Duan, Haoyan Xu, Yida Huang, Jie Feng, and Yueyang Wang. Multivariate time series forecasting with transfer entropy graph. Tsinghua Science and Technology, 28(1):141–149, 2022.

Wei Fan, Shun Zheng, Xiaohan Yi, Wei Cao, Yanjie Fu, Jiang Bian, and Tie-Yan Liu. Depts: Deep expansion learning for periodic time series forecasting. arXiv preprint arXiv:2203.07681, 2022.

Syeda Sitara Wishal Fatima and Afshin Rahimi. A review of time-series forecasting algorithms for industrial manufacturing systems. Machines, 12(6):380, 2024.

Ulla K Heede and Alexey V Fedorov. Towards understanding the robust strengthening of enso and more frequent extreme el niño events in cmip6 global warming simulations. Climate Dynamics, 61(5):3047–3060, 2023.

Qihe Huang, Lei Shen, Ruixin Zhang, Shouhong Ding, Binwu Wang, Zhengyang Zhou, and Yang Wang. Crossgnn: Confronting noisy multivariate time series via cross interaction refinement. Advances in Neural Information Processing Systems, 36:46885–46902, 2023.

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

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.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926, 2017.

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, 2022a.

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In International conference on learning representations, 2021.

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, 2022b.

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

Binh Nguyen-Thai, Vuong Le, Ngoc-Dung T Tieu, Truyen Tran, Svetha Venkatesh, and Naeem Ramzan. Learning evolving relations for multivariate time series forecasting. Applied Intelligence, 54(5):3918–3932, 2024.

Zongjiang Shang and Ling Chen. Mshyper: Multi-scale hypergraph transformer for long-range time series forecasting. arXiv preprint arXiv:2401.09261, 2024.

Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 7912–7921, 2019.

E Vijay, Arindam Jati, Nam Nguyen, Gift Sinthong, and Jayant Kalagnanam. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2023.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In The eleventh international conference on learning representations, 2023.

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.

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

Yuankai Wu, Hongyu Yang, Yi Lin, and Hong Liu. Spatiotemporal propagation learning for network-wide flight delay prediction. IEEE Transactions on Knowledge and Data Engineering, 36(1):386–400, 2023.

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.

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

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875, 2017.

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, pages 8980–8987, 2022.

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, pages 11121–11128, 2023a.

Zhen Zeng, Rachneet Kaur, Suchetha Siddagangappa, Saba Rahimi, Tucker Balch, and Manuela Veloso. Financial time series forecasting using cnn and transformer. arXiv preprint arXiv:2304.04912, 2023b.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In The eleventh international conference on learning representations, 2023.

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.

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, pages 11106–11115, 2021.

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.

MSCGrapher: Learning Multi-Scale Dynamic Correlations for Multivariate Time Series Forecasting (Supplementary Material)

Xian Yang[1] **Zhenguo Zhang**[*1] **Shihao Lu**[1]

[1]Department of Computer Science and Technology, Yanbian University, 977 Gongyuan Road, Yanji, 133002, China

A DESCRIPTIONS OF NOTATIONS

To facilitate understanding of the symbols used in this paper, a detailed summary of the key notations is provided in Table 7.

B DETAILS OF IMPLEMENTING MULTIVARIATE TIME SERIES FORECASTING

In this section, we summarize the detailed information on datasets, baselines, evaluation metrics, and hyperparameter settings. The code is available: https://github.com/sopphia2001/MSCGrapher.

B.1 DATASETS

We use a total of 12 real-world datasets to evaluate MSCGrapher, covering various domains such as weather, electricity, and traffic. The specific information of the datasets is as follows:

- Flight[Cai et al., 2024]: Contains flight data variations for seven major airports in Europe from January 2019 to December 2021, including flight data particularly related to COVID-19 (post-2020), with important information such as flight numbers, departure and destination airports, departure times, and landing times.
- ETT[Zhou et al., 2021]: Includes seven factors of electric transformers from July 2016 to July 2018. There are four subsets: ETTh1 and ETTh2 are recorded hourly, while ETTm1 and ETTm2 are recorded every 15 minutes.
- Weather[Wu et al., 2021]: Includes 21 meteorological factors collected every 10 minutes from 1990 to 2016.
- Electricity[Wu et al., 2021]: This dataset contains electricity consumption data from the UCI Machine Learning Repository, which summarizes hourly electricity consumption of 321 customers from 2012 to 2014.
- Exchange-Rate[Wu et al., 2021]: Collects panel data of daily exchange rates for eight countries from 1990 to 2016.
- PEMS: Contains data from the California public transportation network collected in 5-minute windows. We use the same two public subsets (PEMS03, PEMS04, PEMS07 and PEMS08) adopted in SCINet[Liu et al., 2022a].

For the multivariate time series forecasting, we set the input length to 96. The output length for the PEMS datasets is set to {12, 24, 36, 96}, while for other datasets, the output length is set to {96, 192, 336, 720}. Table 8 lists the detailed information of the datasets, which is crucial for understanding the characteristics of the datasets.

*Corresponding author.
*Corresponding author.

Table 7: Description of the key notations.

Notation	Descriptions
$\mathbf{X}_t$	Original input series
$\mathbf{Y}_t$	Target output series
N	Number of variables in the series
L	Length of the historical window
T	Length of the prediction window
$\mathbf{X}_{t,:}$	Time series collected at time step t
$\mathbf{X}_{:,n}$	Entire time series of of each variable indexed by n
$G = (\mathbf{V}, \mathbf{E})$	Graph with node set V and edge set E
G_{strong}	Strong correlation graph
G_{weak}	Weak correlation graph
$f = \{f_1, \ldots, f_k\}$	Multi-scale representations in MTS
$\mathbf{A}^k_{\text{strong}}$	Adjacency matrix of the strong correlation graph at scale k
$\mathbf{A}^k_{\text{weak}}$	Adjacency matrix of the weak correlation graph at scale k
$\mathbf{E}_{i\in\{m,d,w,h,t\}}$	Embeddings for month, day, week, hour, and minute
$\mathbf{H}_{\text{emb}}$	Embedding of the original input series
$\mathbf{X}_f$	Fast Fourier Transform of $\mathbf{H}_{\text{emb}}$
$\mathbf{F}$	Overall amplitude measure
p_i	Period corresponding to different scales
$\mathbf{X}^i$	The i-th reshaped time series for time scale i
$\mathbf{E}^l_1, \mathbf{E}^l_2$	Learnable parameters at layer l for source and target node embeddings
$\mathbf{A}^l$	Adjacency matrices at layer l
$\mathbf{A}^l_{\text{strong}}, \mathbf{A}^l_{\text{weak}}$	Strong and Weak correlation matrices at layer l
$\mathbf{E}^l_{\text{index}}$	Edge index matrix at layer l indicating connections
$\mathbf{A}^l_{\text{edge}}$	Edge attribute matrix at layer l describing edge features
$\mathbf{G}_{\text{data}}$	Contains node and edge features
$\mathbf{B}^l$	All-zero batch tensor
r	Source node of an edge
c	Target node of an edge
$\mathbf{E}_{\text{rep}}$	Node features from source and target node indices
$\mathbf{S}_{\text{edge}}$	Computed edge scores
$\mathbf{A}^l_f$	Fused adjacency matrix at layer l
$\mathbf{X}^i_{\text{out}}$	The output after fusion at scale i

B.2 BASELINES

We compare MSCGrapher with 13 baselines to validate its forecasting performance. We select outstanding time series forecasting models from 2021 to 2024. The specific model codes are as follows:

- Informer: `https://github.com/zhouhaoyi/Informer2020`
- Autoformer: `https://github.com/thuml/Autoformer`
- Pyraformer: `https://github.com/ant-research/Pyraformer`
- FEDformer: `https://github.com/MAZiqing/FEDformer`
- Stationary: `https://github.com/thuml/Nonstationary_Transformers`
- TiDE: `https://github.com/google-research/google-research/tree/master/tide`
- TimesNet: `https://github.com/thuml/Time-Series-Library`
- DLinear: `https://github.com/honeywell21/DLinear`
- MSGNet: `https://github.com/YoZhibo/MSGNet`
- MSHyper: `https://github.com/shangzongjiang/Ada-MSHyper`

Table 8: Detailed Information of Datasets. The frequency indicates the sampling interval of the time points.

Datasets	Nodes	Prediction Length	Train/Valid/Test Size	Split Ratio	Frequency
Flight	7	{96, 192, 336, 720}	(18221, 2537, 5165)	7:1:2	Hourly
ETTh1	7	{96, 192, 336, 720}	(8545, 2881, 2881)	6:2:2	Hourly
ETTh2	7	{96, 192, 336, 720}	(8545, 2881, 2881)	6:2:2	Hourly
ETTm1	7	{96, 192, 336, 720}	(34465, 11521, 11521)	6:2:2	15 min
ETTm2	7	{96, 192, 336, 720}	(34465, 11521, 11521)	6:2:2	15 min
Weather	21	{96, 192, 336, 720}	(36792, 5271, 10540)	7:1:2	10 min
Electricity	321	{96, 192, 336, 720}	(18317, 2633, 5261)	7:1:2	Hourly
Exchange-Rate	8	{96, 192, 336, 720}	(5120, 665, 1422)	7:1:2	Daily
PEMS03	358	{12, 24, 48, 96}	(15629, 5147, 5147)	6:2:2	5 min
PEMS04	307	{12, 24, 48, 96}	(10100, 3303, 3304)	6:2:2	5 min
PEMS07	883	{12, 24, 48, 96}	(16839, 5550, 5550)	6:2:2	5 min
PEMS08	170	{12, 24, 48, 96}	(10618, 3476, 3477)	6:2:2	5 min

- CrossGNN: `https://github.com/hqh0728/CrossGNN`
- StemGNN: `https://github.com/microsoft/StemGNN`
- FourierGNN: `https://github.com/aikunyi/FourierGNN`

B.3 EVALUATION METRICS

In the experiments, we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics. For multivariate time series, given the true value $\mathbf{Y}_t = \{x_t, \ldots, x_{t+T-1}\} \in \mathbb{R}^{N \times T}$ at time step t and the predicted values $\widehat{\mathbf{Y}}_t = \{x_t, \ldots, x_{t+T-1}\} \in \mathbb{R}^{N \times T}$ for N variables over the next T time steps, the definitions of the metrics are as follows, where $x_{ij} \in \mathbf{Y}_t$, $\widehat{x}_{ij} \in \widehat{\mathbf{Y}}_t$:

$$\mathrm{MSE} = \frac{1}{N \times T} \sum_{i=1}^{N} \sum_{j=1}^{T} (x_{ij} - \widehat{x}_{ij})^2 \tag{14}$$

$$\mathrm{MAE} = \frac{1}{N \times T} \sum_{i=1}^{N} \sum_{j=1}^{T} |x_{ij} - \widehat{x}_{ij}| \tag{15}$$

B.4 SETUP AND HYPERPARAMETERS

All experiments are conducted on an RTX 4090 24GB GPU using the PyTorch framework. We use the Adam optimizer with a learning rate of 10^{-4} and a batch size of 32. The default loss function is MSE, with the number of training epochs set to 10 and early stopping applied where appropriate. The embedding dimension c_{dim} is set within the range {16, 32, 64, 128, 512, 1024}, k is set within the range {3, 5}, and the number of graph convolution layers is set to 2. All comparison baseline models are implemented based on the benchmarks from the TimesNet[Wu et al., 2022] repository, which builds upon the configurations provided in the original papers or official code of each model. Specific hyperparameters for different datasets are provided in Table 9.

C IMPLEMENTATION DETAILS FOR VALIDATING THE TRANSFERABILITY OF CORREALTION LEARNING METHODS

In this section, we summarize the detailed information on datasets, baselines, evaluation metrics, and hyperparameter settings.

Table 9: Hyperparameter settings for different datasets.

Datasets	Flight	ETTh1	ETTh2	ETTm1	ETTm2	Weather	Electricity	Exchange	PEMS03	PEMS08
Epochs	10									
Batch size	32									
Loss	MSE									
Optimizer	Adam									
Learning rate	1e-4									
k	5	3	5	3		5				
C_{dim}	32					64	512	64	512	
Ratio	0.25									
Dropout	0.05	0.1	0.05		0.3	0.05		0.2	0.05	
Dim of E	100	10					100	10		
Heads	8									

C.1 DATASETS

We use four datasets for validation. In addition to Electricity and Exchange-Rate, we also use the Solar-Energy and Traffic datasets. The specific information is as follows:

- Solar-Energy: This dataset contains solar energy data collected by the National Renewable Energy Laboratory in 2007, sampled every 10 minutes from 137 photovoltaic stations in Alabama.
- Traffic: This dataset includes road occupancy data (ranging between 0 and 1) from the California Department of Transportation. The data is aggregated hourly from 862 sensors in the San Francisco Bay Area from 2015 to 2016.

According to the original paper, the four datasets are split chronologically into training (60%), validation (20%), and test sets (20%). For validating the transferability of correaltion learning methods, we set the input window to 168, and the output horizons to {3, 6, 12, 24}. Specifically, the prediction horizons for the Solar-Energy dataset range from 30 to 240 minutes, for Traffic and Electricity datasets range from 3 to 24 hours, and for the Exchange-Rate dataset range from 3 to 24 days. Table 10 lists the detailed information of the datasets, which is crucial for understanding their characteristics.

Table 10: Detailed Information of Datasets. The frequency indicates the sampling interval of the time points.

Datasets	Nodes	Horizon	Train/Valid/Test Size	Split Size	Frequency
Solar-Energy	137	{3,6,12,24}	(31536,10512,10512)	6:2:2	10 minutes
Traffic	862	{3,6,12,24}	(10526,3509,3509)	6:2:2	Hourly
Electricity	321	{3,6,12,24}	(15782,5261,5261)	6:2:2	Hourly
Exchange-Rate	8	{3,6,12,24}	(4553,1518,1517)	6:2:2	Daily

C.2 BASELINES

We integrate the correlation method into two different baseline models to validate the effectiveness and transferability of the proposed correlation learning methods through comparisons of accuracy before and after integration. The specific baseline model codes are as follows:

- MTGNN: `https://github.com/nnzhan/MTGNN`
- TEGNN: `https://github.com/RRRussell/MTHetGNN`

C.3 EVALUATION METRICS

In the experiments, we use Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) as evaluation metrics. For multivariate time series, given the true value $\mathbf{Y}_t = \{x_t, \ldots, x_{t+T-1}\} \in \mathbb{R}^{N \times T}$ at time step t and the predicted values $\widehat{\mathbf{Y}}_t = \{x_t, \ldots, x_{t+T-1}\} \in \mathbb{R}^{N \times T}$ for N variables over the next T time steps, the definitions of the metrics are as follows, where $x_{ij} \in \mathbf{Y}_t$, $\widehat{x}_{ij} \in \widehat{\mathbf{Y}}_t$:

$$\mathrm{RSE} = \frac{\sqrt{\sum_{i=1}^{T} \sum_{j=1}^{N} \left(x_{ij} - \widehat{x}_{ij}\right)^2}}{\sqrt{\sum_{i=1}^{T} \sum_{j=1}^{N} \left(x_{ij} - \mathrm{mean}(x)\right)^2}} \tag{16}$$

$$\mathrm{CORR} = \frac{1}{T} \sum_{j=1}^{N} \frac{\sum_{i=1}^{T} \left(x_{ij} - \mathrm{mean}(x_{*j})\right) \left(\widehat{x}_{ij} - \mathrm{mean}(\widehat{x}_{*j})\right)}{\sqrt{\sum_{i=1}^{T} \left(x_{ij} - \mathrm{mean}(x_{*j})\right)^2 \sum_{i=1}^{T} \left(\widehat{x}_{ij} - \mathrm{mean}(\widehat{x}_{*j})\right)^2}} \tag{17}$$

C.4 SETUP AND HYPERPARAMETERS

All experiments are conducted on an RTX 4090 24GB GPU using the PyTorch framework. We use the Adam optimizer for fine-tuning and optimize all trainable parameters through backpropagation. The learning rate is set to 10^{-3}, and we choose L1Loss as the loss function with 30 training epochs. We integrate correlation learning methods into the baseline models with a ratio of 0.25, using the same hyperparameters as in the original papers. For detailed parameters of each baseline, please refer to [Wu et al., 2020, Duan et al., 2022].

D REVIEW WINDOW EXPERIMENTS

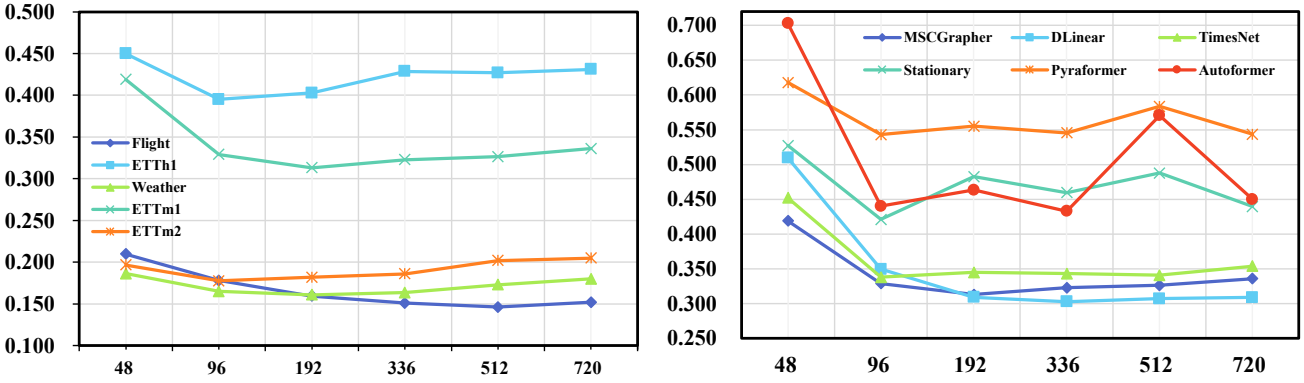
To better highlight our model's performance on long series, we extend the input length to increase the historical information available to the model, evaluating its performance in handling longer temporal dependencies. We conducte experiments on five datasets with input lengths of {48, 96, 192, 226, 512, 720} and an output length of 96, using MSE as the evaluation metric. The results are shown in Figure 7a.

The figure shows that as input length increases, MSCGrapher's overall predictive performance declines, highlighting its strength in capturing long-term trends and complex dependencies. We believe that multi-scale operations in MSCGrapher are crucial to this process. These operations divide long time series into sub-series of different scales, shortening series length, improving processing efficiency, and capturing dependencies across various time scales. This method effectively overcomes the performance fluctuations and instability issues that traditional Transformer models face with long series. Additionally, multi-scale operations enable MSCGrapher to flexibly model across different time scales, providing a comprehensive understanding of complex patterns and trends. By applying the Transformer mechanism to each sub-series, MSCGrapher can make fine-grained and coarse-grained predictions, improving overall accuracy.

In general, the size of the historical review window influences the dependencies that the model learns from historical information. To assess MSCGrapher's effectiveness with extended historical windows, we compare it with well-performing models on the ETTm1 dataset. The results are presented in Figure 7b. We observe that as the historical review window increases, MSCGrapher decreases and eventually stabilizes, which indicates that it can effectively handle large amounts of historical data and extract helpful information. This is due to MSCGrapher's capability to identify different time scales and learn corresponding relationship graphs allows it to capture long-series time dependencies effectively. TimesNet shows a similar overall trend to MSCGrapher but performs poorly; DLinear performs relatively well in long-term forecasting but is unsatisfactory for short-term predictions; Other methods exhibit significant fluctuations as the historical review window lengthens.

E MORE LEARNED GRAPH VISUALIZATION

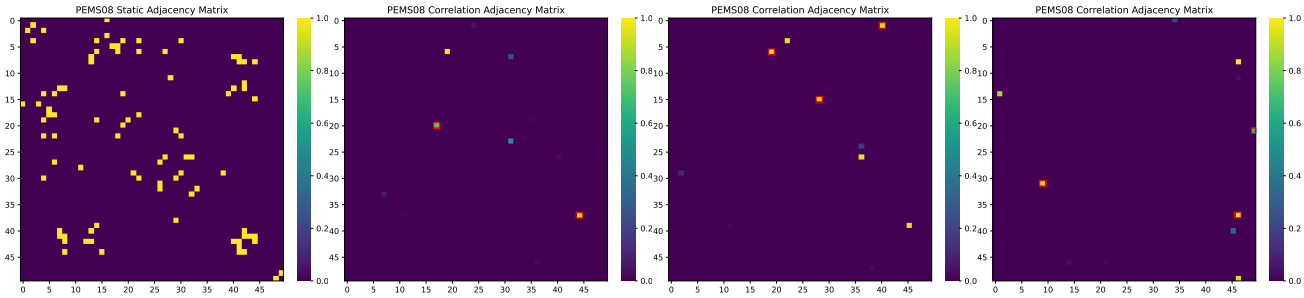
To illustrate the specific role of the information obtained from the correlation graph, we provide additional visualization examples to demonstrate its advantages, as shown in Figures 8 and 9. Taking the PEMS dataset as an example, which



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

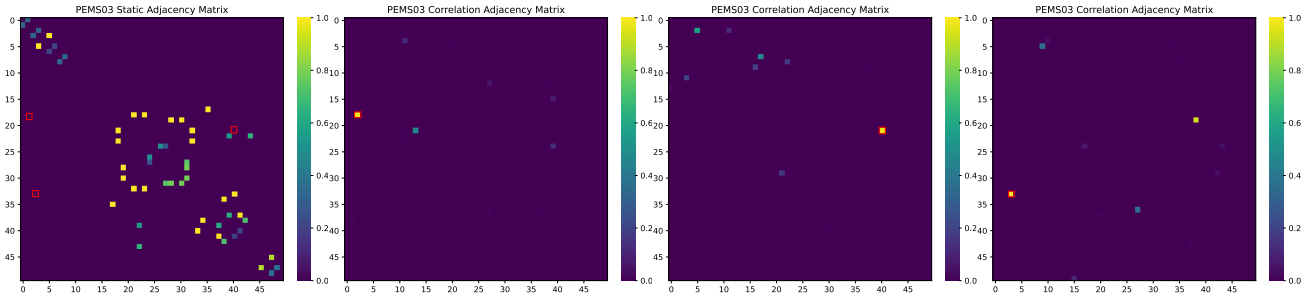
(b) Forecasting results for different models on ETTm1 with output length 96 and input lengths in {48, 96, 192, 336, 512}.

Figure 7: Review window experiments with diffident datasets(a) and models(b).



(a) Static (b) 6h (c) 12h (d) 24h

Figure 8: Visualization of the adjacency matrix for the top 50 nodes in the PEMS08 dataset, showcasing the learnable adjacency matrices at different scales in the first layer and the preset static adjacency matrix. The preset static adjacency matrix fails to capture the correlations between time series with strong similarities.



(a) Static (b) 6h (c) 12h (d) 24h

Figure 9: Visualization of the adjacency matrix for the top 50 nodes in the PEMS03 dataset, showcasing the learnable adjacency matrices at different scales in the first layer and the preset static adjacency matrix. The preset static adjacency matrix fails to capture the correlations between time series with strong similarities.

includes an adjacency matrix based on predefined distances, we compare it with the correlation matrix learned by our model to further validate the effectiveness of the correlation information.

Specifically, as seen in Figures 8 and 9, the learned correlation matrix is much sparser, indicating that MSCGrapher relies on a more concise set of inter-series relationships during prediction. At the same time, the learned adjacency matrix captures connections between nodes that are physically distant but exhibit high similarity in their time series behavior, as highlighted by the red box in the figures. To deepen the understanding of the model's learning mechanism, we also visualize the time series corresponding to node pairs with high weights in the adjacency matrix, as shown in Figure 10.

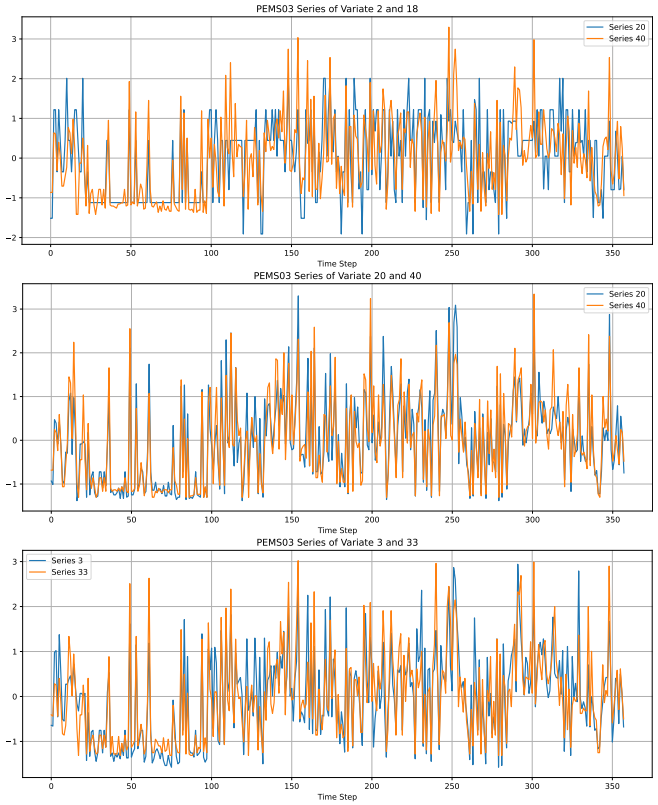


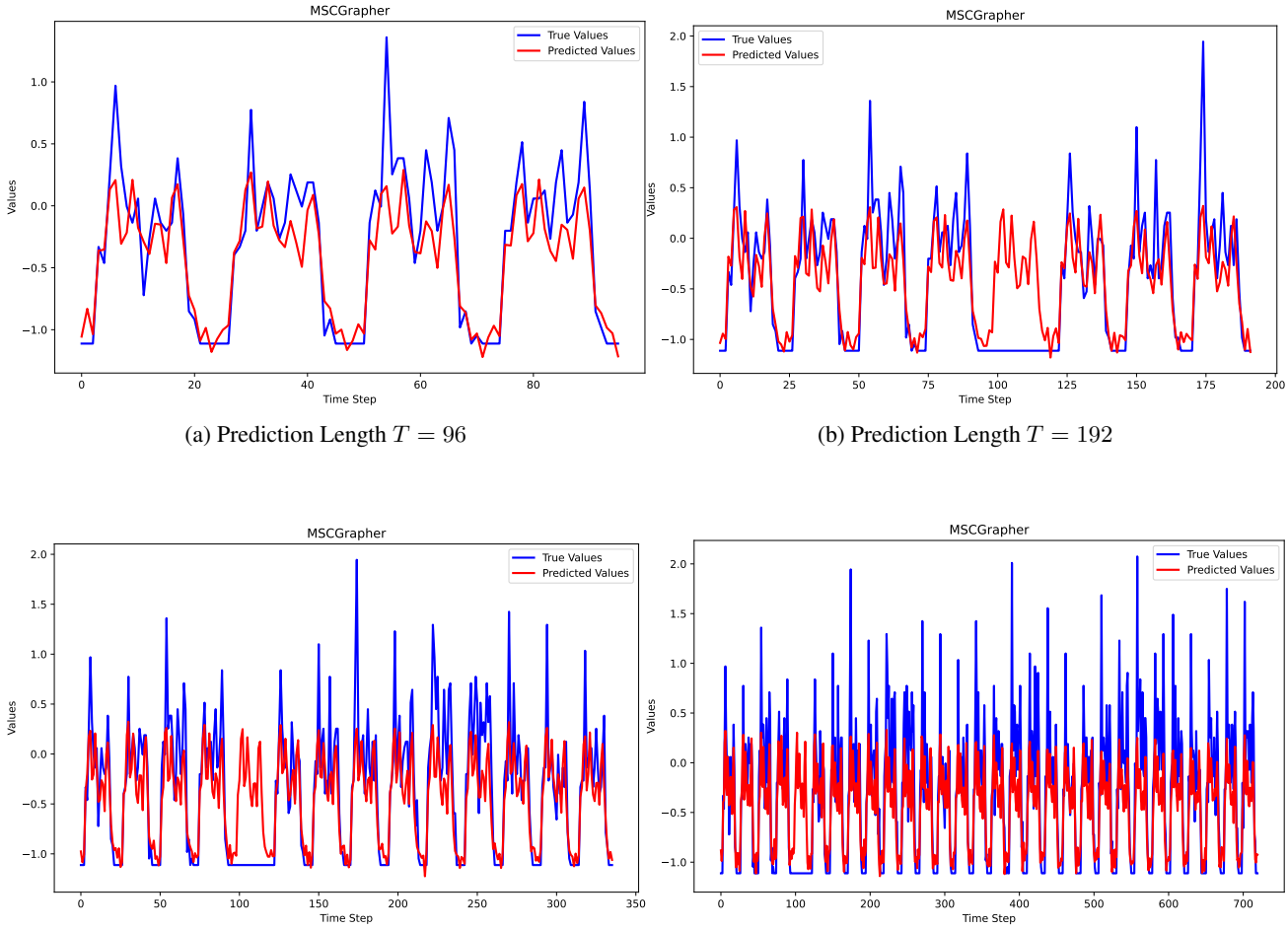
Figure 10: Visualization of time series for node pairs with higher values in the learnable adjacency matrix of the PEMS03.

F MORE FORECASTING RESULTS VISUALIZATION

To more intuitively demonstrate MSCGrapher's outstanding performance, we present additional visual results in the figures. Figure 11 shows the forecasting results for the Flight dataset with an input length of 96 and output lengths of {96, 192, 336, 720}. Figures 12 and 13 display the forecasting results for different models on the Exchange and Weather datasets, with an input length of 96 and output length of 96. It can be observed that, across these two datasets, the overall prediction accuracy of all models does not reach an ideal level. However, MSCGrapher maintains relatively high accuracy within the 0–100 prediction interval and demonstrates a more stable forecasting trend than other models in subsequent time periods, indicating its strong capability to model sequence dependencies even when faced with highly volatile data.

G ALL FORECASTING RESULTS

In this section, we present the complete results of multivariate time series forecasting. We use 10 datasets and compare them with 12 deep learning models. The best results are highlighted in red **bold** and the second best results are underlined in blue. From Table 12, MSCGrapher demonstrates outstanding performance in MTSF task. Specifically, MSCGrapher achieves the best performance 46 times throughout the forecasting results, far surpassing other models. Additionally, compared to GNN models in Table 11, MSCGrapher also achieves more advanced performance.



(a) Prediction Length $T = 96$

(b) Prediction Length $T = 192$

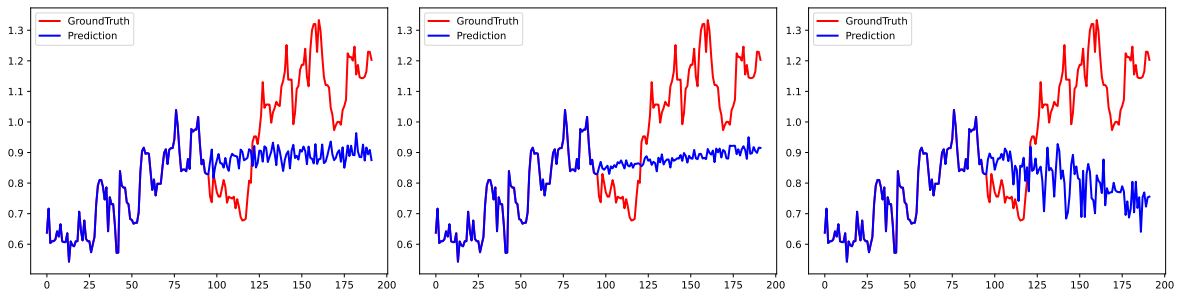
(c) Prediction Length $T = 336$

(d) Prediction Length $T = 720$

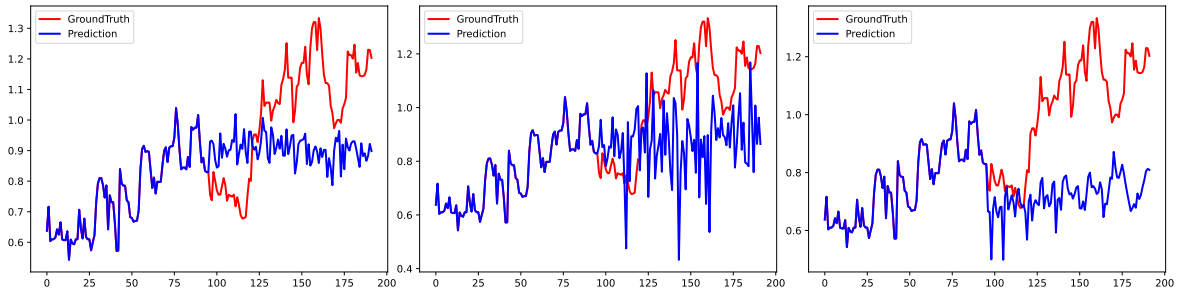
Figure 11: Visualization of the prediction results for the Flight dataset. The input length is 96.

Table 11: The best results for MSCGrapher compared to GNN models are highlighted in red bold.

Dataset		Electricity		Weather		PEMS03		PEMS04		PEMS08	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ours	96	**0.165**	0.274	**0.165**	**0.214**	**0.078**	**0.184**	**0.092**	**0.207**	**0.116**	**0.223**
	192	0.187	0.294	**0.215**	**0.257**	**0.103**	**0.214**	**0.109**	**0.228**	**0.149**	**0.255**
	336	**0.198**	0.307	0.275	**0.301**	**0.151**	**0.257**	**0.144**	**0.265**	**0.189**	**0.275**
	720	**0.232**	**0.333**	0.363	**0.359**	**0.213**	**0.309**	**0.201**	**0.319**	**0.313**	**0.350**
FourierGNN	96	0.211	0.307	0.177	0.240	0.087	0.202	0.112	0.231	0.143	0.263
	192	0.214	0.312	0.218	0.279	0.120	0.240	0.153	0.272	0.210	0.320
	336	0.227	0.325	**0.265**	0.318	0.177	0.294	0.209	0.321	0.216	0.311
	720	0.260	0.354	**0.336**	0.370	0.218	0.333	0.247	0.354	0.294	0.356
StemGNN	96	0.165	**0.267**	0.181	0.250	0.119	0.244	0.144	0.276	0.246	0.319
	192	**0.180**	**0.283**	0.226	0.289	0.179	0.305	0.188	0.317	0.281	0.337
	336	0.200	**0.306**	0.287	0.338	0.191	0.303	0.234	0.342	0.305	0.356
	720	0.243	0.345	0.379	0.406	0.258	0.355	0.303	0.396	0.380	0.393

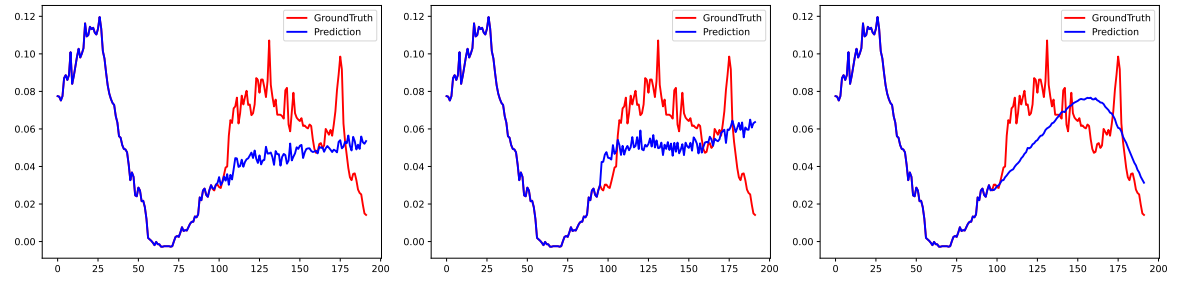


(a) MSCGrapher (b) MSGNet (c) DLinear

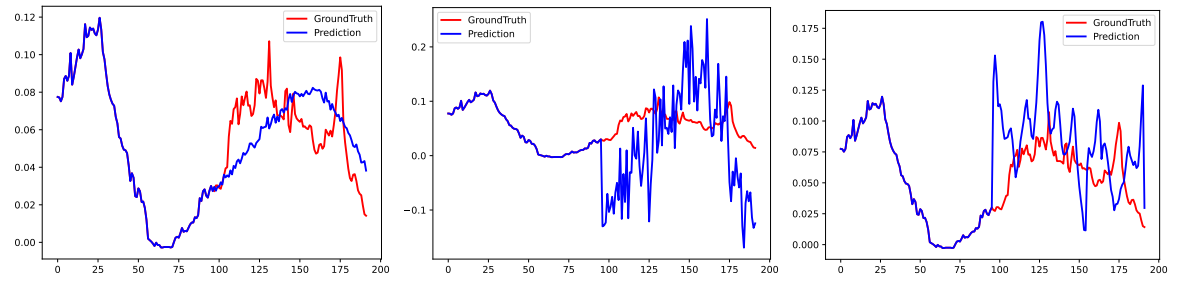


(d) TiDE (e) Pyraformer (f) Autoformer

Figure 12: Visualization of forecasting results for exchange dataset with an input length of 96 and output length of 96.



(a) MSCGrapher (b) MSGNet (c) DLinear



(d) TiDE (e) Pyraformer (f) Autoformer

Figure 13: Visualization of forecasting results for weather dataset with an input length of 96 and output length of 96.

Table 12: The complete results of multivariate time series forecasting. For the PEMS dataset, the output lengths are {12, 24, 36, 48}, and for other datasets, the output lengths are {96, 192, 336, 720}.

Models		OURS		MSGNet(2024)		Dlinear(2023)		TimesNet(2023)		TiDE(2023)		Stationary(2022)		FEDformer(2022)		Pyraformer(2022)		Autoformer(2021)		Informer(2021)	
Datasets	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Flight	96	**0.178**	**0.294**	0.183	0.300	0.221	0.337	0.237	0.350	0.224	0.340	0.386	0.461	0.360	0.452	0.436	0.496	0.204	0.319	0.333	0.405
	192	**0.183**	**0.301**	0.189	0.306	0.220	0.336	0.224	0.337	0.227	0.342	0.422	0.478	0.397	0.474	0.437	0.492	0.200	0.314	0.358	0.421
	336	**0.201**	**0.315**	0.207	0.320	0.229	0.342	0.289	0.394	0.234	0.346	0.544	0.533	0.492	0.527	0.444	0.494	0.201	0.318	0.398	0.446
	720	**0.243**	**0.351**	0.252	0.358	0.263	0.366	0.310	0.408	0.270	0.371	0.890	0.697	0.424	0.488	0.476	0.504	0.345	0.426	0.476	0.484
ETTh1	96	0.393	0.410	0.423	0.440	0.386	**0.400**	**0.384**	0.402	0.427	0.450	0.513	0.491	0.395	0.424	0.664	0.612	0.449	0.459	0.865	0.713
	192	0.439	0.443	0.445	0.469	0.437	0.432	**0.436**	**0.429**	0.472	0.486	0.534	0.504	0.469	0.470	0.790	0.681	0.500	0.482	1.008	0.792
	336	**0.480**	0.468	0.481	0.473	0.481	**0.459**	0.491	0.475	0.527	0.527	0.588	0.535	0.530	0.499	0.891	0.738	0.521	0.496	1.107	0.809
	720	**0.488**	**0.485**	0.540	0.524	0.519	0.516	0.521	0.500	0.644	0.605	0.643	0.616	0.598	0.544	0.963	0.782	0.514	0.512	1.181	0.965
ETTh2	96	0.326	0.370	0.348	0.399	0.333	0.387	0.340	0.374	**0.304**	**0.359**	0.476	0.458	0.358	0.397	0.645	0.597	0.346	0.388	3.755	1.525
	192	0.407	**0.413**	0.404	0.431	0.477	0.476	0.402	0.414	**0.394**	0.422	0.512	0.493	0.429	0.439	0.788	0.683	0.456	0.452	5.602	1.931
	336	0.422	0.434	0.435	0.443	0.594	0.541	0.452	0.452	**0.385**	**0.421**	0.552	0.551	0.496	0.487	0.907	0.747	0.482	0.486	4.721	1.835
	720	**0.416**	**0.441**	0.421	0.451	0.831	0.657	0.462	0.468	0.463	0.475	0.562	0.560	0.463	0.474	0.963	0.783	0.515	0.511	3.647	1.625
ETTm1	96	**0.323**	**0.368**	0.326	0.371	0.345	0.372	0.338	0.375	0.356	0.381	0.386	0.398	0.379	0.419	0.543	0.510	0.505	0.475	0.672	0.571
	192	**0.374**	0.396	0.376	0.397	0.380	0.389	0.374	**0.387**	0.391	0.399	0.459	0.444	0.426	0.441	0.557	0.537	0.553	0.496	0.795	0.669
	336	0.421	0.426	0.417	0.421	0.413	0.413	**0.410**	**0.411**	0.424	0.423	0.495	0.464	0.445	0.459	0.745	0.655	0.621	0.537	1.212	0.871
	720	0.483	0.461	0.482	0.459	**0.474**	0.453	0.478	**0.450**	0.480	0.456	0.543	0.516	0.543	0.490	0.908	0.724	0.671	0.561	1.166	0.823
ETTm2	96	**0.178**	**0.260**	0.184	0.267	0.193	0.292	0.187	0.267	0.182	0.264	0.192	0.274	0.203	0.287	0.435	0.507	0.255	0.339	0.365	0.453
	192	**0.248**	**0.307**	0.248	0.307	0.284	0.362	0.249	0.309	0.256	0.323	0.280	0.339	0.269	0.328	0.730	0.673	0.281	0.340	0.533	0.563
	336	**0.311**	**0.345**	0.312	0.346	0.369	0.427	0.321	0.351	0.313	0.354	0.334	0.361	0.325	0.366	1.201	0.845	0.339	0.372	1.363	0.887
	720	0.410	**0.402**	0.414	0.404	0.554	0.522	**0.408**	0.403	0.419	0.410	0.417	0.413	0.421	0.415	3.625	1.451	0.433	0.432	3.379	1.338
weather	96	**0.165**	**0.214**	0.165	0.214	0.196	0.255	0.172	0.220	0.202	0.261	0.173	0.223	0.217	0.296	0.896	0.556	0.266	0.336	0.300	0.384
	192	**0.215**	**0.257**	0.215	0.258	0.237	0.296	0.219	0.261	0.242	0.298	0.245	0.285	0.276	0.336	0.622	0.624	0.307	0.367	0.598	0.544
	336	**0.275**	**0.301**	0.276	0.301	0.283	0.335	0.280	0.306	0.287	0.335	0.321	0.338	0.339	0.380	0.739	0.753	0.359	0.395	0.578	0.523
	720	0.363	**0.359**	0.371	0.362	**0.345**	0.381	0.365	0.359	0.351	0.386	0.414	0.410	0.403	0.428	1.004	0.934	0.419	0.428	1.059	0.741
electricity	96	**0.165**	0.274	0.169	0.279	0.197	0.282	0.168	**0.272**	0.194	0.277	0.169	0.274	0.193	0.308	0.386	0.449	0.201	0.317	0.274	0.368
	192	0.187	0.294	0.188	0.296	0.196	0.285	0.184	0.289	0.193	**0.280**	**0.182**	0.286	0.201	0.315	0.386	0.443	0.222	0.334	0.396	0.386
	336	**0.198**	0.307	0.199	0.307	0.209	0.301	0.198	0.300	0.206	**0.296**	0.200	0.304	0.214	0.329	0.378	0.443	0.231	0.338	0.300	0.394
	720	0.232	0.333	0.227	0.330	0.245	0.333	**0.220**	**0.320**	0.242	0.328	0.222	0.321	0.246	0.355	0.376	0.445	0.254	0.361	0.373	0.439
exchange	96	0.099	0.227	0.105	0.231	**0.088**	**0.218**	0.107	0.234	0.107	0.233	0.111	0.237	0.148	0.278	1.093	0.884	0.197	0.323	0.847	0.752
	192	0.193	**0.315**	0.196	0.318	**0.176**	0.315	0.226	0.344	0.201	0.323	0.219	0.335	0.271	0.380	1.085	0.976	0.300	0.369	1.204	0.895
	336	0.369	0.442	0.370	0.442	**0.313**	**0.427**	0.367	0.448	0.351	0.432	0.421	0.476	0.460	0.500	1.597	1.090	0.509	0.524	1.672	1.036
	720	0.923	0.730	0.940	0.738	**0.839**	**0.695**	0.964	0.746	0.940	0.735	1.092	0.769	1.195	0.841	1.735	1.124	1.447	0.941	2.478	1.310
PEMS03	12	**0.078**	**0.184**	0.079	0.186	0.122	0.243	0.085	0.192	0.178	0.305	0.081	0.188	0.126	0.251	0.152	0.253	0.272	0.385	0.183	0.284
	24	**0.103**	**0.214**	0.104	0.215	0.201	0.317	0.118	0.223	0.257	0.371	0.105	0.214	0.149	0.275	0.186	0.290	0.334	0.440	0.193	0.293
	36	**0.151**	**0.257**	0.151	0.257	0.333	0.425	0.155	0.260	0.379	0.463	0.154	0.257	0.227	0.348	0.520	0.526	1.032	0.782	0.202	0.304
	48	**0.213**	**0.309**	0.218	0.313	0.457	0.515	0.228	0.317	0.490	0.539	0.247	0.336	0.348	0.434	0.584	0.590	1.031	0.796	0.225	0.319
PEMS08	12	0.116	0.223	0.116	0.224	0.154	0.276	0.112	0.212	0.227	0.343	**0.109**	**0.207**	0.173	0.273	0.216	0.246	0.436	0.485	0.297	0.313
	24	0.149	0.255	0.149	0.255	0.248	0.353	0.141	0.238	0.318	0.409	**0.140**	**0.236**	0.210	0.301	0.249	0.267	0.467	0.502	0.321	0.317
	36	**0.189**	**0.275**	0.196	0.285	0.440	0.470	0.198	0.283	0.497	0.510	0.211	0.294	0.320	0.394	0.288	0.297	0.966	0.733	0.308	0.311
	48	**0.313**	**0.350**	0.361	0.527	0.674	0.565	0.320	0.351	0.721	0.592	0.345	0.367	0.442	0.465	0.324	0.359	1.385	0.915	0.327	0.361
1st Count		**46**		0		11		11		7		5		0		0		0		0	