ST-Hyper: Learning High-Order Dependencies Across Multiple Spatial-Temporal Scales for Multivariate Time Series Forecasting

Binqing Wu*

College of Computer Science and Technology, Zhejiang University Hangzhou, Zhejiang, China binqingwu@cs.zju.edu.cn

Zongjiang Shang

College of Computer Science and Technology,
Zhejiang University
Hangzhou, Zhejiang, China
zongjiangshang@cs.zju.edu.cn

ABSTRACT

In multivariate time series (MTS) forecasting, many deep learning based methods have been proposed for modeling dependencies at multiple spatial (inter-variate) or temporal (intra-variate) scales. However, existing methods may fail to model dependencies across multiple spatial-temporal scales (ST-scales, i.e., scales that jointly consider spatial and temporal scopes). In this work, we propose ST-Hyper to model the high-order dependencies across multiple ST-scales through adaptive hypergraph modeling. Specifically, we introduce a Spatial-Temporal Pyramid Modeling (STPM) module to extract features at multiple ST-scales. Furthermore, we introduce an Adaptive Hypergraph Modeling (AHM) module that learns a sparse hypergraph to capture robust high-order dependencies among features. In addition, we interact with these features through tri-phase hypergraph propagation, which can comprehensively capture multi-scale spatial-temporal dynamics. Experimental results on six real-world MTS datasets demonstrate that ST-Hyper achieves the state-of-the-art performance, outperforming the best baselines with an average MAE reduction of 3.8% and 6.8% for long-term and short-term forecasting, respectively.

KEYWORDS

Multivariate Time Series Forecasting, Hypergraph Modeling, Spatial Temporal Graph Neural Network

1 INTRODUCTION

Multivariate time series (MTS) forecasting plays a critical role in various real-world applications, e.g., transportation management [38], environment monitoring[16], and weather prediction[37]. The task is inherently challenging, as the complex spatial (inter-variable) and temporal (intra-variable) dependencies in MTS should be effectively modeled.

Recent deep learning methods have demonstrated strong potential in modeling spatial and temporal dependencies. To model spatial dependencies, some works have leveraged graph neural networks (GNNs) [33, 39], hypergraph neural networks (HGNNs) [30, 42], and Transformer-based architectures [20, 41]. To model

Jianlong Huang*
College of Computer Science and Technology,
Zhejiang University
Hangzhou, Zhejiang, China
22251226@cs.zju.edu.cn

Ling Chen[†]

College of Computer Science and Technology,
Zhejiang University
Hangzhou, Zhejiang, China
lingchen@cs.zju.edu.cn

temporal dependencies, some works have employed a variety of architectures, including multi-layer perceptrons (MLPs) [31, 40], recurrent neural networks (RNNs) [10, 13], temporal convolutional networks (TCNs) [3, 35], and self-attention mechanisms [23, 43]. Despite these advances, these methods only model dependencies at a single spatial or temporal scale, which restricts their capacity to capture the multi-scale dynamics that are often present in real-world time series data.

In reality, dependencies at multiple spatial or temporal scales are ubiquitous. For example, weather data often exhibits dependencies at different spatial scales (e.g., city-scale and country-scale) and dependencies at different temporal scales (e.g., day-scale and season-scale). To capture these dependencies, many methods have been proposed. Some methods focus on modeling dependencies at multiple spatial scales [6, 8, 29, 34]. They cluster variables into groups of varying sizes and model variable-level and group-level spatial patterns. Some methods focus on modeling dependencies at different temporal scales [18, 25, 28]. They segment time series with different temporal durations or resolutions and model short-term and long-term temporal patterns. More recently, several methods have attempted to model dependencies across multiple spatial scales and multiple temporal scales independently, either in parallel or sequentially [4, 5, 16]. However, these works treat multiple spatial and temporal scales in isolation, potentially overlooking dependencies that arise across multiple spatial-temporal scales (ST-scales)-scales that jointly consider spatial and temporal scopes (e.g., city-day, city-season, and country-season scales in weather data).

ST-scales are essential, as some important patterns emerge only at specific ST-scales. For example, the sea-land breeze is a daily phenomenon that occurs in coastal cities, corresponding to a small-spatial-short-temporal scale. In addition, the Meiyu season, characterized by persistent rainfall, is a seasonal phenomenon observed in cities within the Yangtze River basin, corresponding to a small-spatial-long-temporal scale. Moreover, climate change (e.g., multi-decade trends across countries) becomes apparent only when analyzed on a large-spatial-long-temporal scale. These examples highlight the necessity of explicitly modeling dependencies across multiple ST-scales to capture the full range of spatial-temporal dynamics of MTS.

^{*}Equal contribution.

[†]Corresponding author.

Despite their importance, modeling such dependencies remains non-trivial. These dependencies are often highly heterogeneous and context-dependent, lacking consistent or pre-defined structural patterns. This variability severely limits the ability to impose unified inductive biases or manually design generalizable structures. In addition, dependencies across multiple spatial-temporal scales give rise to intricate cross-scale interactions. Such interactions are not limited to simple pairwise patterns; instead, they often involve high-order patterns that conventional modeling frameworks are fundamentally ill-equipped to capture.

To this end, we propose ST-Hyper, which is the first work to incorporate adaptive hypergraph modeling to learn the high-order dependencies across multiple ST-scales for MTS forecasting. The main contributions of this work are as follows:

- We introduce a Spatial-Temporal Pyramid Modeling (STPM) module that first learns a spatial pyramidal graph to obtain series at multiple spatial scales, and then employs temporal multi-scale networks and ST-Encoders to extract features across multiple ST-scales. In particular, we introduce a graph pooling loss for spatial pyramidal graphs based on Laplacian preservation and entropy regularization to encourage the grouping of correlated variables and non-overlapping variable assignments, which can reduce information redundancy.
- We introduce an Adaptive Hypergraph Modeling (AHM) module that learns a sparse hypergraph structure to focus on the most correlated features, which can capture the robust high-order dependencies across multiple ST-scales. In addition, we perform tri-phase hypergraph propagation to interact features across multiple ST-scales, which can allow them to reinforce each other and mitigate the impact of anomalies or perturbations. Thereby, AHM can comprehensively and reliably capture multi-scale spatial-temporal dynamics.
- We evaluate ST-Hyper on six real-world MTS datasets. The experimental results demonstrate that ST-Hyper achieves state-of-the-art (SOTA) performance, outperforming the best baseline with an average MAE reduction of 3.8% and 6.8% for long-term and short-term forecasting, respectively.

2 RELATED WORKS

Due to the superiority in modeling spatial and temporal dependencies, deep-learning-based methods have become the mainstream choices for MTS forecasting. For modeling temporal dependencies, existing methods use MLPs, CNNs, RNNs, and self-attention networks. For example, DLinear [40] employs two MLPs to model seasonal and trend-cycle components, thereby enabling MLPs to capture trends in time series. LSTNet [13] employs the long short-term memory (LSTM) to model temporal dependencies, but it suffers the problem of gradient vanishing/exploding in RNNs. PatchTST [22] segments time series into multiple subseries, thereby reducing the computational complexity by subseries-level self-attention modeling. For modeling spatial dependencies, existing methods use GNNs, HGNNs, and Transformers. For example, MTGNN [38] employs a graph structure learning method based on learnable node embeddings to capture dependencies among variables. DyHSL [42]

employs a hypergraph structure learning module to model the high-order dependencies among variables for traffic forecasting. MixRNN+ [14] captures pairwise and high-order spatial dependencies via hypergraph structure learning for spatial-correlated time series forecasting. Crossformer [41] integrates a multi-head self-attention network with learnable vectors as routers to distribute messages across all variables. Nethertheless, these methods only model dependencies at a single spatial or temporal scale, which restricts their capacity to capture the multi-scale dynamics.

To solve this problem, many methods have been proposed to model dependencies at multiple spatial scales or multiple temporal scales. Some methods aim to model dependencies at multiple spatial scales. For example, HGCN [8] generates hierarchical spatial graphs via spectral clustering on the distance adjacency matrix and uses spatial-temporal graph neural networks (STGNNs) to model spatial and temporal dependencies at each spatial scale. GAGNN [6] learns a probabilistic assignment matrix between cities and city groups to model spatial dependencies at each spatial scale. HSTGL [17] learns micro and macro graphs that are constructed based on transportation connectivity and the geographical correlation, respectively, to model dependencies at two spatial scales. Some methods aim to model dependencies at multiple temporal scales. For example, MSHyper [24] extracts features at multiple temporal scales and pre-defines high-order dependencies between these features. TimeMixer [28] decomposes time series at each temporal scale into seasonal and trend components, then separately mixes multi-scale seasonal and trend representations. MSGNet [4] extracts series at different temporal scales based on periods of time series, and employs GNNs and self-attention networks to model spatial and temporal dependencies at each temporal scale, respectively. In addition, some methods model dependencies across multiple spatial scales and dependencies across multiple temporal scales. For example, AirFormer [16] leverages two types of self-attention networks to model dependencies across multiple spatial scales and dependencies at each temporal scale. Corrformer [37] employs a cross-correlation mechanism to model dependencies across multiple spatial scales and an auto-correlation mechanism to model dependencies across multiple temporal scales. However, these methods treat spatial and temporal scales in isolation, potentially overlooking dependencies that arise from their joint interaction.

To address this issue, ST-Hyper first extracts features at multiple ST-scales and learn a sparse hypergraph between them adaptively, which can capture robust high-order dependencies across multiple ST-scales. Moreover, we conduct tri-phase hypergraph propagation to interact with these features, which can capture multi-scale spatial-temporal dynamics comprehensively and reliably.

3 PRELIMINARIES

MTS Forecasting Formulation. MTS forecasting aims to predict future values based on historical values. Formally, we represent historical values as $X = \{X^1, X^2, \dots, X^T\} \in \mathbb{R}^{N \times T}$, where N is the number of variables and T is the input length of MTS. MTS forecasting is formulated as learning a function \mathcal{F} that maps historical T steps of values to future τ steps of values:

$$\hat{X}^{T+1:T+\tau} = \mathcal{F}(X;\Theta) \in \mathbb{R}^{N \times \tau},\tag{1}$$

where Θ denotes all learnable parameters of \mathcal{F} .

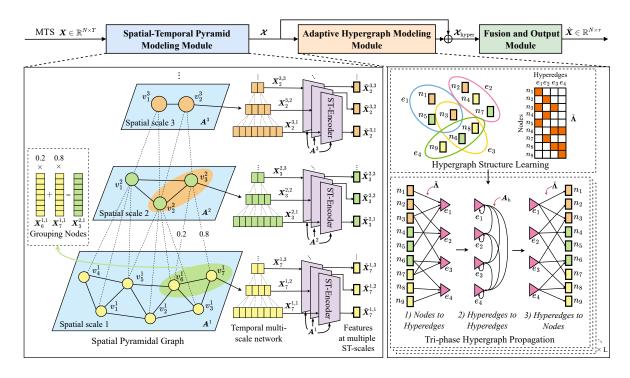


Figure 1: The Framework of ST-Hyper.

Definition of a Graph for MTS. A graph for MTS is defined as G = (V, E, A), where V denotes a set of N nodes. Each node represents a variable. v_i denotes node i in V. E denotes a set of edges. Each edge represents a correlation between variables. $A \in \mathbb{R}^{N \times N}$ is a weighted adjacency matrix to represent the structure of graph G.

Definition of a Hypergraph for MTS. A hypergraph for MTS is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} denotes a set of nodes, and \mathcal{E} denotes a set of hyperedges. Unlike an edge that only connects two nodes in a graph, a hyperedge in a hypergraph can connect one or more nodes. The hypergraph structure can be represented by an incidence matrix $\Lambda \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$, where each entry $\Lambda(v, e)$ indicates whether a node is in a hyperedge or not. A weighted incidence matrix can be formulated as:

$$\Lambda(v,e) = \begin{cases} w(v,e), & \text{if } v \in e, \\ 0, & \text{otherwise,} \end{cases}$$
(2)

where w(v, e) ranges from 0 to 1, representing the assignment possibility from node v to hyperedge e.

Memory-Network-Based Graph Structure Learning. A memory network contains an array of memory items (i.e., learnable vectors) to store feature information for pattern matching [32]. The memory-network-based graph structure learning method [12] augments the generation of node embeddings representing node information in a graph, which can be formulated as:

$$A = \operatorname{SoftMax} \left(\operatorname{ReLU} \left(E_1 \left(E_2 \right)^{\mathrm{T}} \right) \right),$$

$$E_1 = W_{E_1} M, \quad E_2 = W_{E_2} M,$$
(3)

where $A \in \mathbb{R}^{N \times N}$ denotes the learned adjacency matrix, $E_1, E_2 \in \mathbb{R}^{N \times d}$ denote parameterized node embeddings with d dimensions. $M \in \mathbb{R}^{m \times d}$ denotes m memory items with d dimensions. $W_{E_1}, W_{E_2} \in \mathbb{R}^{N \times m}$ denote projection matrices that map memory items to node embeddings.

4 METHODOLOGY

4.1 Model Overview

Empowered by adaptive hypergraph modeling, ST-Hyper aims to model the high-order dependencies across multiple ST-scales. As shown in Figure 1, Spatial-Temporal Pyramid Modeling (STPM) module is introduced to first transform input MTS into series at multiple spatial scales and then extract features at each ST-scale. Next, Adaptive Hypergraph Modeling (AHM) module is introduced to model the high-order dependencies across multiple ST-scales. Finally, the features from STPM and AHM modules are aggregated and fed into Fusion and Output module for final predictions.

4.2 Spatial-Temporal Pyramid Modeling Module

STPM module is designed to extract features at multiple ST-scales from fine to coarse. Specifically, STPM incorporates *Spatial Pyramidal Graph (SPG) Learning* to construct graphs at multiple spatial scales and learn mappings between spatial scales, as well as *Multi-Scale Feature Extraction* to transform input MTS into features at multiple ST-scales.

Spatial Pyramidal Graph Learning. SPG learning is introduced to first construct graphs at multiple spatial scales. We utilize a

memory-network-based graph structure learning method (referred to Equation 3) to learn a graph structure for each spatial scale. For the spatial scale j, the learned spatial graph can be denoted as $A^j \in \mathbb{R}^{N_j \times N_j}$, where $N_j = \lfloor N_{j-1}/q \rfloor$ represents the number of nodes at spatial scale j. q is the graph pooling ratio used to coarsen spatial nodes. As shown in Figure 1, v_i^j denotes node i in A^j . To discover the mapping between two adjacent spatial scales, we learn a probabilistic matrix $S^j \in \mathbb{R}^{N_j \times N_{j+1}}$ to assign the nodes at spatial scale j to group nodes at spatial scale j + 1. The entries of S^j range from 0 to 1.

To learn group-wise dependencies between nodes, ST-Hyper integrates a **graph pooling loss** \mathcal{L}_{GP} as a regularization term:

$$\mathcal{L}_{GP} = \mathcal{L}_{LP} + \mathcal{L}_{E} = ||A_{dtw}^{j}, S^{j}(S^{j})^{T}||_{F} + \frac{1}{n} \sum_{i=1}^{n} \mathcal{P}(S_{i}^{j}), \quad (4)$$

where $\mathcal{L}_{\mathrm{LP}}$ represents the Frobenius norm, which intuitively means that nodes with stronger correlations (i.e., larger weights in A^j_{dtw}) should be grouped together. \mathcal{L}_{E} represents the average entropy loss of all n rows of S^j , which makes each row of S^j close to a one-hot vector, so that the membership of each group node can be clearly defined. \mathcal{P} denotes the entropy function. Dynamic Time Warping (DTW) [21] can measure the similarity between time series. In equation 4, $A^1_{\mathrm{dtw}} \in \mathbb{R}^{N \times N}$ is obtained by computing pairwise DTW distances of N time series over the entire training time range. Then, we can get DTW distance-based adjacency matrix at larger spatial scales based on assignment matrices:

$$A_{\rm dtw}^{j} = (S^{j-1})^{\rm T} A_{\rm dtw}^{j-1} S^{j-1}, \tag{5}$$

where $A_{\text{dtw}}^{j} \in \mathbb{R}^{N_{j} \times N_{j}}$ represents the DTW distance-based adjacency matrix at spatial scale j.

Multi-Scale Feature Extraction. This sub-module aims to extract features at multiple ST-scales based on the input MTS $X \in \mathbb{R}^{N \times T}$ and the constructed SPG. We define $X_i^{j,k}$ as features of node i at spatial scale j and temporal scale k. The input MTS X is represented as $X^{1,1} \in \mathbb{R}^{N_1 \times T_1}$, where $N_1 = N$ and $T_1 = T$, corresponding to features at the smallest ST-scale. Given $X^{1,1}$, we first utilize multiple assignment matrices $\{S^j\}_{j=1}^{J-1}$ to obtain series at multiple spatial scales:

$$X^{j+1,1} = (S^j)^{\mathrm{T}} X^{j,1}, \tag{6}$$

where $X^{j,1} \in \mathbb{R}^{N_j \times T_1}$ represents the series at spatial scale j. Thus, we can obtain $\{X^{j,1}\}_{j=1}^{J} \in \mathbb{R}^{(N_1+\ldots+N_j+\ldots+N_j)\times T_1}$, representing series at total J spatial scales.

To extract features at multiple temporal scales for each spatial scale, we utilize temporal multi-scale networks incorporating 1D CNNs to coarsen the temporal information. Given series at spatial scale $j \, X^{j,1}$, we employ 1D convolutional layers with kernel size 1×1 and 1×2 average pooling layers to obtain series at larger temporal scales:

$$X^{j,k} = \text{Pooling}\left(\text{Conv}\left(X^{j,k-1}\right)\right),$$
 (7)

where $X^{j,k} \in \mathbb{R}^{N_j \times T_k}$ represents the series at spatial scale j and temporal scale k, with T_k equal to $\frac{T_1}{2^{(k-1)}}$. To aggregate information of adjacent time steps, we patchify $X^{j,k}$ by dividing each time series into a set of non-overlapping patches [22]:

$$X_{\text{patch}}^{j,k} = \text{Linear}\left(\text{Patchify}\left(X^{j,k}\right)\right),$$
 (8)

where $X_{\mathrm{patch}}^{j,k} \in \mathbb{R}^{N_j \times \lfloor \frac{T_k}{r} \rfloor \times D}$ represents features at spatial scale j and temporal scale k after patchifying and a linear projection, r is the patch length, and D is the feature dimension. After this process, we can obtain $X = \{X_{\mathrm{patch}}^{j,k}\}_{j=1,k=1}^{J,K}$, representing features at total J spatial scales and K temporal scales (i.e., $J \times K$ ST-scales).

Then, ST-Encoders based on graph convolutional recurrent units (GCRUs) [1] are introduced to extract spatial-temporal information by coupling GNNs and gated recurrent units (GRUs). For each ST-scale, we employ an ST-Encoder with non-shared learnable parameters to extract scale-specific spatial-temporal information:

$$X_{\text{en}}^{j,k} = \text{GCRUs}(X_{\text{patch}}^{j,k}, A^j), \tag{9}$$

where $X_{\mathrm{en}}^{j,k} \in \mathbb{R}^{N \times D}$ represents the features at spatial scale j and temporal scale k after encoding, which is the final hidden state outputted by GCRUs and D is the feature dimension. Recall that the spatial graph at each spatial scale is learned based on a memory network, which can store typical features specific to a spatial scale. Thus, we augment $X_{\mathrm{en}}^{j,k}$ by matching it with the memory network at a specific spatial scale:

$$Q^{j,k} = X_{\text{en}}^{j,k} W_j + b_j,$$

$$X_{\text{att}}^{j,k} = \text{SoftMax}(Q^{j,k} (M^j)^T) M^j,$$

$$\tilde{X}^{j,k} = \text{Concat}(X_{\text{att}}^{j,k}, X_{\text{en}}^{j,k}),$$
(10)

where $Q^{j,k} \in \mathbb{R}^{N_j \times d}$ denotes the query matrix. $M^j \in \mathbb{R}^{m \times d}$ represents the m memory items with d dimensions stored in the memory network at spatial scale j. $X_{\mathrm{att}}^{j,k} \in \mathbb{R}^{N_j \times d}$ denotes the features after an attention-based pattern matching operation. After that, we concatenate $X_{\mathrm{att}}^{j,k}$ and $X_{\mathrm{en}}^{j,k}$ to obtain the features augmented by a memory network $\tilde{X}^{j,k} \in \mathbb{R}^{N_j \times (D+d)}$. Through STPM module, we can obtain $X = \{\tilde{X}^{j,k}\}_{j=1,k=1}^{J,K} \in \mathbb{R}^{(N_1+\ldots+N_J)\times K\times (D+d)}$, representing features at $J \times K$ ST-scales.

4.3 Adaptive Hypergraph Modeling Module

AHM module is introduced to model the high-order dependencies across multiple ST-scales through adaptive hypergraph modeling. Specifically, AHM incorporates *hypergraph structure learning* to discover group-wise dependencies among features from multiple ST-scales and *tri-phase hypergraph propagation* to further capture the high-order dependencies across multiple ST-scales.

Hypergraph Structure Learning. We treat each feature of X as a node in a hypergraph. Thus, the total number of nodes in a hypergraph α is $(N_1 + \ldots + N_J) \times K$. The information of each node in a hypergraph comes from the corresponding feature of X. We introduce a randomly initialized weighted incidence matrix $\Lambda \in \mathbb{R}^{\alpha \times \beta}$ to reflect the hypergraph structure, where β denotes the number of hyperedges. Each entry of Λ (ranging from 0 to 1) represents the probability of assigning a node to a hyperedge. However, learning a dense incidence matrix will result in too many nodes being associated with each hyperedge. To this end, each column of Λ is sparsified by selecting top K' weights, indicating that the most correlated K' nodes are associated with a hyperedge. Thus, we can get the sparsified incidence matrix $\tilde{\Lambda} \in \mathbb{R}^{\alpha \times \beta}$, which can discover the group-wise dependencies among nodes.

Tri-phase Hypergraph Propagation. Tri-phase hypergraph propagation is introduced to model the interaction between nodes and hyperedges based on the learned hypergraph structure, consisting of nodes to hyperedges, hyperedges to hyperedges, and hyperedges to nodes propagation phases. Compared to the traditional hypergraph convolution [2], there are two advantages: 1) Hyperedges to hyperedges phase can enhance the dependency modeling among hyperedges by learning a hyperedge graph and employing graph attention networks (GATs) to adaptively interact hyperedges. 2) Hyperedges to nodes phase can update node information by adaptively interacting nodes with their associated hyperedges based on an attention mechanism.

1) Nodes to Hyperedges. This phase aims to make each hyperedge aggregate the information from nodes associated with it, so that each hyperedge can represent a group of features across multiple ST-scales. This process can be formulated as:

$$\mathcal{E}_1 = \phi(U\tilde{\Lambda}^T X) + \tilde{\Lambda}^T X, \tag{11}$$

where $\mathcal{E}_1 \in \mathbb{R}^{\beta \times D_e}$ represents the obtained hyperedges and $D_e = D + d$ is the hyperedge dimension. $U \in \mathbb{R}^{\beta \times \beta}$ represents the normalized weights of hyperedges. ϕ denotes an activation function.

2) Hyperedges to Hyperedges. Since each hyperedge is aggregated from a group of nodes, modeling the interaction among hyperedges is significant for capturing group-wise dependencies. To discover the relationships between hyperedges, we utilize the memorynetwork-based graph learning method (referred to Equation 3) to learn a hyperedge graph $A_{\rm h} \in \mathbb{R}^{\beta \times \beta}$. Moreover, we leverage a GAT [27] to adaptively determine the importance of neighboring hyperedges to each hyperedge:

$$\mathcal{E}_2 = GAT(\mathcal{E}_1, A_h), \tag{12}$$

where $\mathcal{E}_2 \in \mathbb{R}^{\beta \times D_e}$ denotes the updated hyperedges. The memory network can store typical features shared on the hyperedge graph. Similar to Equation 10, we augment \mathcal{E}_2 by an attention-based pattern matching operation:

$$Q_{h} = \mathcal{E}_{2}W_{Q} + b_{Q},$$

$$\mathcal{E}'_{2} = \text{SoftMax}(Q_{h}(M_{h})^{T})M_{h},$$

$$\tilde{\mathcal{E}}_{2} = \text{Concat}(\mathcal{E}_{2}, \mathcal{E}'_{2}),$$
(13)

where $Q_{\rm h} \in \mathbb{R}^{\beta \times d}$ denotes the query matrix. $\mathcal{E}_2' \in \mathbb{R}^{\beta \times d}$ denotes the hyperedges after interacting with the memory items $M_{\rm h} \in \mathbb{R}^{m \times d}$. We can get memory-augmented hyperedges $\tilde{\mathcal{E}}_2 \in \mathbb{R}^{\beta \times (D_{\rm e}+d)}$ by concatenating \mathcal{E}_2 and \mathcal{E}_2' . Furthermore, we aggregate \mathcal{E}_1 and $\tilde{\mathcal{E}}_2$ by an MLP:

$$\hat{\mathcal{E}} = MLP(\mathcal{E}_1 + (W\tilde{\mathcal{E}}_2 + b)), \tag{14}$$

where $\hat{\mathcal{E}} \in \mathbb{R}^{\beta \times D_e}$ represents the updated hyperedges.

3) Hyperedges to Nodes. After the propagation between hyperedges, we update node information by interacting nodes with their associated hyperedges based on an attention mechanism. According to the incidence matrix $\tilde{\Lambda}$, we can get an attention mask to identify which hyperedges are associated with each node:

$$\Gamma_{i,j} = \begin{cases} 0, & \tilde{\Lambda}_{i,j} \neq 0, \\ -\infty, & \tilde{\Lambda}_{i,j} = 0. \end{cases}$$
 (15)

Based on the attention mask Γ , we employ an attention mechanism to interact nodes with their associated hyperedges:

$$Q = XW_{q} + b_{q}, \quad K = \hat{\mathcal{E}}W_{k} + b_{k}, \quad V = \hat{\mathcal{E}}W_{v} + b_{v},$$

$$X_{\text{att}} = \text{SoftMax}(QK^{T} + \Gamma)V,$$

$$X_{\text{hyper}} = \text{LayerNorm}(\text{MLP}(X_{\text{att}}) + X),$$
(16)

where $Q \in \mathbb{R}^{\alpha \times (D+d)}$ represents the query matrix projected from nodes, $K, V \in \mathbb{R}^{\beta \times D_e}$ represent the key matrix and the value matrix, respectively, projected from hyperedges, and $D_e = D + d$. $X_{\mathrm{att}} \in \mathbb{R}^{\alpha \times D_e}$ represents the attention output after the interaction between nodes and hyperedges. $X_{\mathrm{hyper}} = \{X_{\mathrm{hyper}}^{j,k}\}_{j=1,k=1}^{J,K} \in \mathbb{R}^{(N_1+\ldots+N_J)\times K\times D}$ represents the updated nodes (i.e., the output of AHM module).

4.4 Fusion and Output Module

Fusion. We fuse features X_{hyper} to get the fused representations for N variables. We first fuse features at all temporal scales for each spatial scale. We use normalized weights $\{\omega^{j,k}\}_{k=1}^K$ to decide the contribution of features at each temporal scale and obtain the fused features at spatial scale j $\mathbf{O}^j \in \mathbb{R}^{N_j \times D}$. After that, we fuse features at all spatial scales by passing features at larger spatial scales to features at spatial scale 1. These two processes can be formulated as:

$$O^{j} = \sum_{k=1}^{K} (\omega^{j,k} X_{\text{hyper}}^{j,k}), \quad X_{\text{fused}} = O^{1} + \sum_{j=2}^{J} \prod_{l=0}^{j-1} S^{l} O^{j}, \quad (17)$$

where $\prod_{l=0}^{j-1} S^l \in \mathbb{R}^{N_1 \times N_j}$ represents probabilities to pass features at spatial scale j to features at spatial scale 1. $X_{\mathrm{fused}} \in \mathbb{R}^{N_1 \times D_{\mathrm{e}}}$ represents the fused representations for N variables, with $N_1 = N$. **Output.** MTS forecasting tasks can be divided into short-term and long-term forecasting, which depends on whether output length is longer than input length [35]. For short-term forecasting, to enhance the dependency modeling of adjacent time steps, we use GCRUs to decode $X_{\mathrm{fused}} \in \mathbb{R}^{N \times D_{\mathrm{e}}}$ and output the next τ hidden states $\hat{H} \in \mathbb{R}^{N \times \tau \times D_{\mathrm{e}}}$ recurrently. \hat{H} is then projected to get τ predictions $\hat{X} \in \mathbb{R}^{N \times \tau}$. For long-term forecasting, to avoid the cumulative error caused by sequential modeling, we use MLPs to transform $X_{\mathrm{fused}} \in \mathbb{R}^{N \times D_{\mathrm{e}}}$ to $\hat{X} \in \mathbb{R}^{N \times \tau}$. We integrate \mathcal{L}_1 loss and graph pooling loss $\mathcal{L}_{\mathrm{GP}}$ (Eq.4) as the training loss:

$$\mathcal{L}_{1} = \sum_{i=1}^{N} |\hat{X}^{T+1:T+\tau} - X^{T+1:T+\tau}|, \quad \mathcal{L}_{train} = \mathcal{L}_{1} + \lambda \mathcal{L}_{GP}, \quad (18)$$

where $\hat{X}^{T+1:T+\tau}$ is predicted values and $X^{T+1:T+\tau}$ is ground truth. λ is a balancing factor that controls the importance of \mathcal{L}_{GP} to the training loss $\mathcal{L}_{\text{train}}$, which ranges from 0 to 1.

5 EXPERIMENTS

5.1 Experimental Settings

Datasets. In light of the number of variables and their inter-variate correlations, we select six public datasets comprising over 100 variables to ensure sufficient complexity and analytical depth. The statistics of these datasets are summarized in Table 1.

Table 1: Dataset statistics.

Datasets	# Samples	# Variables	Sample rate	Input length	Output length	Feature
METR-LA	34,272	207	5 minutes	12	12	Traffic speed
PEMS-BAY	52,116	325	5 minutes	12	12	Traffic speed
China-AQI	20,400	209	1 hour	96	24	Air Quality Index
Electricity	26,304	321	1 hour	96	96, 192, 336, 720	Electricity consumption
Solar-Energy	52,560	137	10 minutes	96	96, 192, 336, 720	Solar power generation
Temperature	17,544	3,850	1 hour	96	192	Temperature

Table 2: Hyperparameter search space and choices on different datasets.

Hyperparameters	Search Space	METR-LA	PEMS-BAY	China-AQI	Electricity	Solar-Energy	Temperature
Spatial graph pooling ratio q	{10, 20, 30, 40, 50}	10	20	20	20	20	30
Number of spatial scales J	{1, 2, 3, 4}	2	2	2	2	2	3
Patch length r	{2, 4, 8, 16, 24}	2	2	16	8	16	16
Number of temporal scales K	{1, 2, 3, 4}	3	3	3	3	3	3
Number of K'	{10, 20, 30, 40, 50}	20	20	20	10	10	20
Balancing factor of $\mathcal{L}_{GP} \lambda$	{1e-3, 5e-2, 1e-2, 5e-1, 1e-1}	1e-1	1e-1	1e-2	1e-2	1e-2	1e-2

Table 3: Results for long-term forecasting.

Method	Dataset		Elect	tricity		Solar-Energy				Temperature
	Horizon	96	192	336	720	96	192	336	720	192
ST-Hyper	MSE	0.143	$\frac{0.166}{0.267}$	0.181	0.225	0.185	0.212	0.229	0.236	0.267
(Ours)	MAE	0.237		0.270	0.316	0.210	0.237	0.249	0.261	0.380
MSHyer	MSE	0.152*	0.171*	0.187*	0.224*	0.215	$\frac{0.223}{0.273}$	0.249	0.264	0.301
(2024)	MAE	0.252*	0.271*	0.284*	0.316*	0.264		0.290	0.308	0.422
iTransformer	MSE	0.148*	0.162*	0.178*	0.225 * 0.317*	0.203*	0.233*	0.248*	0.249*	0.288
(2024)	MAE	0.240*	0.253*	0.269*		0.237*	0.261*	0.273*	0.275*	0.391
TimeMixer (2024)	MSE MAE	0.153* 0.247*	0.166* 0.256*	0.185* 0.277*	0.225* 0.310*	$\frac{0.195}{0.266}$	0.224 0.287	$\frac{0.237}{0.295}$	0.242	0.292 0.396
MSGNet	MSE	0.166*	0.184*	0.195*	0.231*	0.232	0.256	0.279	0.287	0.298
(2024)	MAE	0.274*	0.292*	0.302*	0.332*	0.277	0.295	0.314	0.32	0.411
Corrformer	MSE	0.179	0.196	0.218	0.251	0.211	0.237	0.252	0.264	0.286
(2023)	MAE	0.285	0.293	0.320	0.339	0.289	0.301	0.314	0.328	0.389
AirFormer	MSE	0.178	0.192	0.205	0.242	0.204	0.225	0.250	0.256	0.321
(2023)	MAE	0.281	0.290	0.309	0.334	0.276	0.293	0.303	0.306	0.434
PatchTST	MSE	0.181*	0.188*	0.204*	0.246*	0.234*	0.267*	0.290*	0.289*	0.295
(2023)	MAE	0.270*	0.274*	0.293*	0.324*	0.286*	0.310*	0.315*	0.317*	0.399
Crossformer	MSE	0.219*	0.231*	0.246*	0.280*	0.310*	0.734*	0.750*	0.769*	0.335
(2023)	MAE	0.314*	0.322*	0.337*	0.363*	0.331*	0.725*	0.735*	0.765*	0.444
CrossGNN	MSE	0.173*	0.195*	0.206*	0.231*	0.297	0.357	0.374	0.376	0.307
(2023)	MAE	0.275*	0.288*	0.300*	0.335*	0.339	0.372	0.389	0.387	0.414
GAGNN	MSE	0.227	0.248	0.275	0.295	0.293	0.342	0.395	0.411	0.322
(2023)	MAE	0.329	0.369	0.383	0.374	0.326	0.354	0.402	0.429	0.423
MTGNN	MSE	0.217*	0.238*	0.260*	0.290*	0.337	0.360	0.377	0.399	0.325
(2020)	MAE	0.318*	0.352*	0.348*	0.369*	0.353	0.380	0.398	0.422	0.431

METR-LA and PEMS-BAY datasets are provided by MTGNN [38]. China-AQI dataset is provided by GAGNN [6]. Electricity and Solar-Energy datasets are provided by Autoformer [36]. Temperature dataset is provided by Corrformer [37]. We conduct the pre-processing strategies on datasets according to the original established protocols in the corresponding papers. We split datasets chronologically for training, validation, and testing with the ratio 7:1:2 for all datasets.

Baselines. We compare ST-Hyper with SOTA MTS methods for long- and short-term forecasting, including: MSHyper [24] builds multi-scale temporal features via pooling and hypergraphs; iTransformer [20] embeds series as variable tokens and models attention among them; TimeMixer [28] samples series at multiple scales, decomposes them into seasonal and trend parts, and mixes them separately; MSGNet [4] extracts series by periods and models spatial/temporal dependencies with GNNs and self-attention; Corrformer [37] uses cross- and auto-correlation for spatial and temporal dependencies; AirFormer [16] employs dual self-attention

for multi-scale spatial and temporal modeling; PatchTST [22] segments series into subseries-level patches with channel-independent Transformers; Crossformer [41] merges adjacent steps for multi-scale features and uses a learnable router with self-attention; Cross-GNN [9] extracts series by periods and models dependencies via cross-scale and cross-variable GNNs; GAGNN [6] constructs geometric graphs and pools them across scales; MTGNN [38] uses 1D CNNs for multi-scale features, stacking GCNs for spatial and TCNs for temporal dependencies.

Moreover, we compare ST-Hyper with SOTA STGNNs focusing on various graph structure learning (adaptive, hierarchical, high-order), including: MegaCRN [12] uses learnable embeddings for spatial and temporal heterogeneity; PDFormer [11] employs learnable embeddings emphasizing delay effects; STAFormer [19] adapts Vanilla Transformer with adaptive embeddings for spatiotemporal dependencies; HGCN [8] builds geometric graphs and applies spectral clustering for hierarchical structures; HiGP [7] models hierarchical dependencies with trainable pooling and differentiable reconciliation; MixRNN [15] introduces relation-aware mixers and physics-informed residuals for higher-order spatial dependencies; ASTHGCN [44] uses time-varying learnable hypergraphs for dynamic higher-order spatial modeling; MvHSTM [26] adopts temporal transformers and dual learnable hypergraphs for local and cross-regional dependencies.

Training Details. We conduct the experiment five times with different random seeds and report the average value for evaluation metrics. Following existing works, we use Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) for short-term forecasting [38], and use MAE and Mean Squared Error (MSE) for long-term forecasting [20]. Our implementation of ST-Hyper is in Python 3.8, utilizing PyTorch 1.9.0, and trained on NVIDIA GeForce RTX 3090 Ti GPU. The code is available at the link ¹. We employ the Adam optimizer with an initial learning rate of 0.001. The model is trained for 150 epochs, and the training is early stopped if the validation loss fails to decrease with 15 epochs. The batch size is set to 8 for Temperature datasets, and 32 for others. The hidden dimension of GCRUs is set to 64. The memory number is set to 20, with dimension 32. The number of layers of AHM module is set to 1. The number of hyperedges is set to 40. The details of hyperparameter tuning for different datasets are given in Table 2.

5.2 Main Results

We evaluate ST-Hyper and baselines on six datasets for long-term and short-term forecasting. **Bold** and <u>underlined</u> indicate the best and second-best performance, respectively. The results are summarized in Table 3, Table 4, and Table 5. Notably, the results for baselines marked with * in Table 3 are sourced from iTransformer [20] or their original publications. Those with * in Table 5 are sourced from STAFormer [19] or their original publications. Other results without marks are obtained by rerunning the official codes using default settings. From these results, we can observe that:

(1) As shown in Table 3, for long-term forecasting, ST-Hyper outperforms the best baselines with an average MAE reduction of 3.8%,

¹https://anonymous.4open.science/ST-Hyper-83E7

Table 4: Results for short-term forecasting.

Method	Dataset	1	METR-L	A	P:	EMS-BA	ΛY	China-AQI			
Method	Horizon	3	6	12	3	6	12	3	6	12	24
ST-Hyper (Ours)	MAE RMSE	2.49	2.86 5.95	3.27 7.08	1.28	1.60 3.68	1.87 4.42	10.88	15.31 23.07	19.86 30.50	24.24 38.39
(Ours)	MAPE	6.34	7.77	9.37	2.68	3.59	4.42	17.47 16.92	24.82	33.48	39.8
MSHyer	MAE	3.54	4.47	6.22	1.62	2.05	2.84	12.83	18.06	23.02	26.40
(2024)	RMSE MAPE	6.73 8.60	8.92 11.64	10.5 12.7	3.46 3.52	4.25 4.41	6.24	22.67 21.75	30.87 29.44	36.74 36.36	43.00 44.26
iTransformer	MAE	4.01	5.29	7.24	1.54	2.08	2.93	12.41	17.36	22.07	26.09
(2024)	RMSE MAPE	9.55 9.00	12.18 11.71	15.42 15.96	3.23 3.19	4.62 4.48	6.4 6.63	21.91 19.22	29.19 27.21	35.83 35.48	41.19 41.5
TimeMixer	MAE	2.88	3.22	3.87	1.61	2.14	2.97	13.96	18.37	23.84	27.57
(2024)	RMSE MAPE	5.88 <u>6.72</u>	6.27 8.33	7.91 10.72	3.33 3.30	4.74 4.66	6.60 6.82	24.44 21.28	30.37 29.89	37.19 36.27	42.25 42.46
MSGNet	MAE	3.72	4.35	6.2	1.52	1.86	2.42	18.45	20.77	23.58	26.24
(2024)	RMSE MAPE	7.10 9.24	8.71 11.33	10.91 13.48	3.40 3.19	4.27 4.26	5.51 5.37	29.46 30.09	32.23 33.15	36.02 38.04	40.84 43.93
Corrformer	MAE	3.26	3.98	4.03	1.55	1.98	2.59	11.74	16.95	22.46	26.54
(2023)	RMSE MAPE	6.16 7.65	7.25 9.47	8.84 11.15	3.24 3.22	4.50 4.36	5.87 6.23	21.63 20.41	29.46 27.38	36.78 36.13	42.66 44.06
AirFormer	MAE	3.6	4.24	4.93	1.46	1.93	2.53	11.87	17.1	22.12	26.18
(2023)	RMSE MAPE	6.84 9.15	8.38 10.57	9.05 12.61	3.19 3.11	4.43 4.31	5.81 6.03	21.37 20.11	29.13 26.85	36.43 35.72	42.04 43.74
PatchTST	MAE	4.27	4.85	5.68	3.57	4.15	4.49	13.49	18.88	23.98	27.20
(2023)	RMSE MAPE	7.21 9.45	8.36 10.61	9.74 12.58	5.91 7.52	6.70 9.10	7.41 10.25	24.01 19.73	30.88 28.67	38.46 38.70	42.81 45.22
Crossformer	MAE	4.65	5.35	6.49	4.76	5.47	5.96	13.08	17.82	22.85	25.99
(2023)	RMSE MAPE	7.95 9.85	9.11 11.29	10.65 13.64	7.93 11.08	8.55 13.23	9.12 14.79	23.67 19.72	28.28 28.35	36.42 39.24	40.67 45.47
CrossGNN	MAE	2.68	3.03	3.40	1.46	1.85	2.37	12.49	17.82	23.69	29.11
(2023)	RMSE MAPE	5.18 6.81	5.79 7.98	$\frac{7.20}{9.91}$	3.05	4.21 4.10	5.40 5.67	22.87 20.91	30.83 28.94	38.62 38.84	46.00 50.45
GAGNN	MAE	4.35	4.78	5.59	1.93	2.43	2.81	12.45	17.22	22.23	26.27
(2023)	RMSE MAPE	7.55 9.79	8.71 10.88	10.48 12.78	3.78 3.99	4.63 4.87	5.78 5.64	22.45 20.62	30.16 27.36	37.35 34.49	43.53 41.31
MTGNN	MAE	2.70	3.06	3.51	1.34	1.67	1.97	12.88	18.96	23.32	28.52
(2020)	RMSE MAPE	5.20 6.85	6.19 8.21	7.26 9.88	2.82	$\frac{3.78}{3.74}$	4.51 4.56	23.37 18.87	32.48 27.95	38.35 35.86	45.24 47.52

Table 5: Results compared with SOTA STGNNs on the META-LA dataset.

Method		3			6				12		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE		
S	ST-Hyper		4.90	6.34	2.86	5.95	7.77	3.27	7.08	9.37	
High-order	MvHSTM*(2025)	2.62	5.03	6.72	2.96	6.00	8.11	3.40	7.15	9.95	
	ASTHGCN*(2023)	2.65	5.05	6.80	3.01	6.07	8.26	3.40	7.08	9.81	
	MixRNN*(2023)	2.63	5.06	/	3.00	6.08	/	3.42	7.16	/	
	MegaCRN*(2023)	2.52	4.94	6.44	2.93	6.06	7.96	3.38	7.23	9.72	
Adaptive	PDFormer*(2023)	2.83	5.45	7.77	3.20	6.46	9.19	3.62	7.47	10.91	
	STAFormer*(2023)	2.65	5.11	6.85	2.97	6.00	8.13	3.34	7.02	9.70	
Hierachical	HiGP*(2024)	2.68	5.19	6.82	3.02	6.31	8.34	3.40	7.49	10.025	
Hierachical	HGCN(2021)	2.89	5.47	7.44	3.34	6.60	8.92	3.87	7.82	10.51	

particularly on Solar-Energy and Temperature datasets with complex spatio-temporal dependencies, highlighting the importance of modeling multi-scale dependencies.

(2) As shown in Table 4, for short-term forecasting, ST-Hyper consistently surpasses all baselines, achieving an average MAE reduction of 6.8%. Its tri-phase hypergraph propagation allows adaptation to local variations while preserving global context, effectively extracting meaningful patterns and improving robustness to noisy spatial-temporal data, i.e., traffic speed and air quality.

(3) As shown in Table 5, ST-Hyper outperforms SOTA STGNNs with adaptive, hierarchical, or high-order graph designs, achieving an average MAE reduction of 2.5%. This improvement stems from its constrained hypergraph construction, where nodes (features at multiple spatio-temporal scales) are guided by a graph pooling

Table 6: Results of ablation study on METR-LA and China-AQI datasets.

Variants	w/o	STPM	ST-Hy	per-M	w/o	\mathcal{L}_{GP}	w/o	AHM	ST-H	yper-H	ST-H	lyper
Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
METR-LA	2.96	6.12	2.93	6.07	2.84	5.94	2.92	6.05	2.87	6.01	2.81	5.89
China-AQI	19.68	34.88	19.39	34.12	19.02	33.24	19.33	33.95	19.05	33.14	18.54	31.32

Table 7: MSE Results with different noise intensities on the Solar-Energy dataset.

SNR	100db	80db	60db	40db
ST-Hyper(Ours)	0.185	0.190	0.202	0.241
iTransformer	0.203	0.228	0.240	0.302
TimeMixer	0.195	0.206	0.232	0.320
CrossGNN	0.297	0.308	0.336	0.401

loss and hyperedges are kept sparse, enhancing the robustness of learned dependencies.

5.3 Ablation Study

STPM module. We evaluate three variants: 1) w/o STPM: removing the STPM module; 2) ST-Hyper-M: replacing memory-network-based graph learning with a non-memory method [1], i.e., $A = \text{SoftMax}(\text{ReLU}(E_1E_2^{\text{T}}))$; 3) w/o \mathcal{L}_{GP} : removing the graph pooling loss. Table 6 shows ST-Hyper outperforms all variants, highlighting the effectiveness of multi-ST-scale feature extraction, memory-network graph learning, and SPG regularization.

AHM module. We evaluate two variants: 1) w/o AHM: removing the AHM module; 2) ST-Hyper-H: replacing tri-phase hypergraph propagation with traditional hypergraph convolution [2]. Table 6 shows ST-Hyper outperforms both, indicating that AHM's tri-phase propagation effectively models high-order dependencies across multiple ST-scales.

5.4 Robustness Analysis

To evaluate model robustness under noisy conditions, we follow the protocol of CrossGNN [9] and add Gaussian white noise of varying intensities to the Solar-Energy dataset. We fix input and output lengths to 96 and use MSE as the evaluation metric. As shown in Table 7, we progressively decrease the signal-to-noise ratio (SNR) from 100 dB to 40 dB, which introduces increasing levels of noise. Across all noise levels, ST-Hyper consistently outperforms the baselines, with relative improvements in MSE ranging from 5.12% to 20.20%. These results highlight the strong noise resilience of ST-Hyper. We attribute this robustness to its ability to capture and integrate patterns across multiple spatio-temporal scales, allowing representations at different granularities to reinforce each other and mitigate the impact of local anomalies or perturbations.

5.5 Hyperparameter Sensitivity Study

Number of Spatial Scales (# Spatial Scale) and Spatial Graph Pooling Ratio q. These two hyperparameters determine the SPG structure. q determines the number of nodes in the spatial graph at spatial scale j (i.e., $N_j = \lfloor N_{j-1}/q \rfloor$). As shown in Figure 2(a), the optimal # Spatial Scale is 2 and the optimal q is 20. The reason is that a smaller # Spatial Scale may fail to model dependencies at

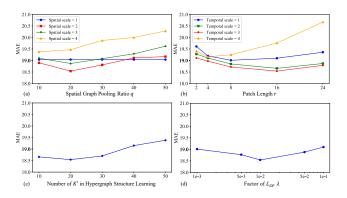


Figure 2: Results of hyperparameter sensitivity study on China-AQI dataset.

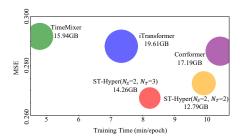


Figure 3: Results of computation cost comparison, where N_S and N_T indicate the number of spatial and temporal scales, respectively, for simplicity.

multiple spatial scales, while a larger # Spatial Scale may introduce too many parameters and make the model difficult to converge. In addition, a smaller q may cause irregular nodes to be grouped into a group node, while a larger q results in too few nodes at larger spatial scales and limits model capacity.

Number of Temporal Scales (# Temporal Scale) and Patch Length r. These two hyperparameters determine how features at multiple temporal scales are extracted for each spatial scale. As shown in Figure 2(b), the optimal # Temporal Scale is 3 and the optimal r is 16. The reason is that a smaller # Temporal Scale may fail to model long-term dependencies, while a larger # Temporal Scale may introduce too many parameters. In addition, a smaller r may overly focus on fine-grained noise, while a larger r may overlook important short-term patterns.

Number of K'. K' determines the number of nodes connected by each hyperedge in the hypergraph. As shown in Figure 2(c), the optimal K' is 20. The reason is that a smaller K' may limit the ability of ST-Hyper to model group-wise dependencies among nodes, while a larger K' may introduce noise into hyperedge features.

Factor of \mathcal{L}_{GP} λ . λ determines the contribution of graph pooling loss \mathcal{L}_{GP} to training loss \mathcal{L}_{train} . As shown in Figure 2(d), the optimal λ is 1e-2. The reason is that a smaller λ may lead to insufficient regularization to SPG learning, while a larger λ may overly constrain the model.

5.6 Computational Cost Comparison

We evaluate ST-Hyper against competitive baselines in training time, GPU memory, and forecasting accuracy, following iTransformer [20], on the Temperature dataset with 3,850 variables. As shown in Fig. 3, ST-Hyper achieves the highest accuracy while using the least GPU memory, thanks to its ability to capture high-order dependencies across multiple spatiotemporal scales using sparse hypergraphs. ST-Hyper requires longer training than TimeMixer, which ignores spatial dependencies, and iTransformer, which ignores multi-scale dynamics. Compared with Corrformer, which models spatial and temporal dependencies independently, ST-Hyper demonstrates superior computational efficiency and predictive accuracy. Overall, ST-Hyper delivers SOTA performance with favorable computational cost, making it practical for real-world forecasting.

5.7 Visualization

Sparsified Hypergraph Incidence Matrix. Figure 4 portrays the subpart of the sparsified hypergraph incidence matrix, i.e., $\tilde{\Lambda}$, on China-AQI dataset. Recall that the entries in the sparsified hypergraph incidence matrix represent assigning possibilities, ranging from 0 to 1, from nodes to hyperedges. The nodes in the hypergraph represent features from multiple ST-scales. On China-AQI dataset, the number of spatial scales is set to 2 and the number of temporal scales is set to 3. As shown in Figure 4, features at different ST-scales are associated with a certain hyperedge. In addition, the member nodes for different hyperedges are not identical, reflecting the diversity of hyperedge information. AHM module incorporating tri-phase hypergraph propagation models the interaction between hyperedges, which can model the high-order dependencies across multiple ST-scales.

Spatial Pyramidal Graph Structure. The visualization in Figure 5 depicts the grouping result for variables on China-AQI dataset, with 2 spatial scales. For each group node at spatial scale 2, we present the nodes at spatial scale 1, with top 10 possibilities, that are assigned to the group node. As shown in Figure 5, we note a significant adjacency among nodes within one group node, primarily owing to their geographical proximity. In addition, there are instances where nodes, despite being geographically distant, are grouped together. This illustrates that our method ST-Hyper may consider a broader range of spatial features when modeling spatial dependencies.

Predicted AQI. We select a case that predicts the future AQI values for Ningbo city on China-AQI dataset to illustrate the superiority of ST-Hyper. As shown in Figure 6, the left part is the historical values (96 hours) of Ningbo city and its neighboring cities and the predicted values (24 hours) of different methods. The right part portrays the geometric distribution of these cities. During this period, a sharp increase occurs in Ningbo city, making forecasting challenging. While the SOTA STGNN-based (MegaCRN) and Transformer-based methods (AirFormer) fail to capture this increase, ST-Hyper successfully does so. This is because ST-Hyper can detect sudden changes in nearby cities in the past by extracting features at multiple ST-scales (e.g., city-hour and city group-day scales), and pass the information about sudden changes to the target city (Ningbo) through interacting features from multiple ST-scales.

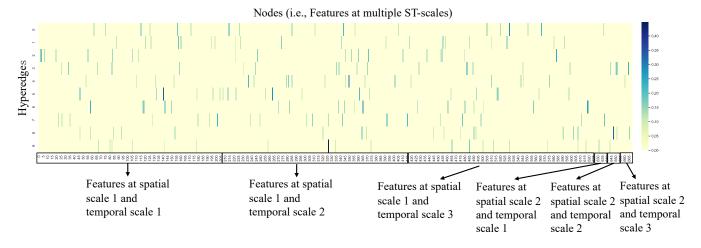


Figure 4: Visualization of sparsified hypergraph incidence matrix on China-AQI dataset.



Figure 5: Visualization of the grouping result of Spatial Pyramidal Graph Learning with 2 spatial scales. Different colors represent different group nodes at spatial scale 2.

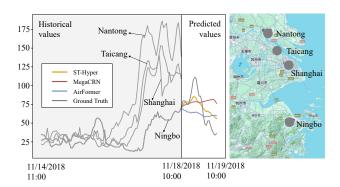


Figure 6: Visualization of predicting AQI for Ningbo city.

6 CONCLUSIONS

In this work, we propose ST-Hyper for MTS forecasting, which can model high-order dependencies across multiple ST-scales through adaptive hypergraph modeling. Extensive experiments on six real-world datasets demonstrate that ST-Hyper achieves SOTA performance, outperforming the best baseline with an average MAE reduction of 3.8% and 6.8% for long-term and short-term forecasting, respectively. Furthermore, additional analyses underscore the robustness of ST-Hyper, as well as the practical importance of modeling dependencies across multiple ST-scales in real-world scenarios. In future work, we will extend ST-Hyper in several directions, including: (1) discovering dynamic hypergraph evolution to capture time-varying high-order dependencies, (2) uncovering directed hypergraphs to enhance interpretability, and (3) improving efficiency to scale ST-Hyper to ultra-large datasets with high-dimensional inputs.

REFERENCES

- Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. In Proceedings of the International Conference on Neural Information Processing Systems. 17804–17815.
- [2] Song Bai, Feihu Zhang, and Philip HS Torr. 2021. Hypergraph convolution and hypergraph attention. Pattern Recognition 110 (2021), 107637.
- [3] Jing Bi, Xiang Zhang, Haitao Yuan, Jia Zhang, and MengChu Zhou. 2021. A hybrid prediction method for realistic network traffic with temporal convolutional network and LSTM. IEEE Transactions on Automation Science and Engineering 19 (2021), 1869–1879.
- [4] Wanlin Cai, Yuxuan Liang, Xianggen Liu, Jianshuai Feng, and Yuankai Wu. 2024. MSGNet: Learning multi-scale inter-series correlations for multivariate time series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence. 11141–11149.
- [5] Ling Chen, Donghui Chen, Zongjiang Shang, Binqing Wu, Cen Zheng, Bo Wen, and Wei Zhang. 2023. Multi-scale adaptive graph neural network for multivariate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering* 35 (2023), 10748–10761.
- [6] Ling Chen, Jiahui Xu, Binqing Wu, and Jianlong Huang. 2023. Group-aware graph neural network for nationwide city air quality forecasting. ACM Transactions on Knowledge Discovery from Data 18 (2023), 1–20.
- [7] Andrea Cini, Danilo Mandic, and Cesare Alippi. 2024. Graph-based time series clustering for end-to-end hierarchical forecasting. In Proceedings of the International Conference on Machine Learning. 8985–8999.
- [8] Kan Guo, Yongli Hu, Yanfeng Sun, Sean Qian, Junbin Gao, and Baocai Yin. 2021. Hierarchical graph convolution network for traffic forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence. 151–159.

- [9] 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. In Proceedings of the International Conference on Neural Information Processing Systems. 46885–46902.
- [10] Xu Huang, Chuyao Luo, Bowen Zhang, Huiwei Lin, Xutao Li, and Yunming Ye. 2024. iTrendRNN: An interpretable trend-aware RNN for meteorological spatiotemporal Prediction. In Proceedings of the AAAI Conference on Artificial Intelligence. AAAI Press, 22132–22140.
- [11] Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. 2023. PDFormer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 4365–4373.
- [12] Renhe Jiang, Zhaonan Wang, Jiawei Yong, Puneet Jeph, Quanjun Chen, Yasumasa Kobayashi, Xuan Song, Shintaro Fukushima, and Toyotaro Suzumura. 2023. Spatio-temporal meta-graph learning for traffic forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence. 8078–8086.
- [13] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 95–104.
- [14] Yuxuan Liang, Kun Ouyang, Yiwei Wang, Zheyi Pan, Yifang Yin, Hongyang Chen, Junbo Zhang, Yu Zheng, David S. Rosenblum, and Roger Zimmermann. 2023. Mixed-Order Relation-Aware Recurrent Neural Networks for Spatio-Temporal Forecasting. IEEE Transactions on Knowledge and Data Engineering 35, 9 (2023), 9254–9268.
- [15] Yuxuan Liang, Kun Ouyang, Yiwei Wang, Zheyi Pan, Yifang Yin, Hongyang Chen, Junbo Zhang, Yu Zheng, David S Rosenblum, and Roger Zimmermann. 2023. Mixed-order relation-aware recurrent neural networks for spatio-temporal forecasting. IEEE Transactions on Knowledge and Data Engineering 35, 9 (2023), 9254–9268.
- [16] Yuxuan Liang, Yutong Xia, Songyu Ke, Yiwei Wang, Qingsong Wen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. 2023. AirFormer: Predicting nationwide air quality in China with transformers. In Proceedings of the AAAI Conference on Artificial Intelligence. 14329–14337.
- [17] Li Lin, Kaiwen Xia, Anqi Zheng, Shijie Hu, and Shuai Wang. 2024. Hierarchical Spatio-Temporal Graph Learning Based on Metapath Aggregation for Emergency Supply Forecasting. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. Association for Computing Machinery, New York, NY, USA, 1410–1419.
- [18] Shengsheng Lin, Weiwei Lin, HU Xinyi, Wentai Wu, Ruichao Mo, and Haocheng Zhong. 2024. CycleNet: Enhancing Time Series Forecasting through Modeling Periodic Patterns. Advances in Neural Information Processing Systems 37 (2024), 106315–106345.
- [19] Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. 2023. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In Proceedings of the 32nd ACM international conference on information and knowledge management. 4125–4129.
- [20] 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 Proceedings of the International Conference on Learning Representations.
- [21] Meinard Müller. 2007. Dynamic time warping. Information Retrieval for Music and Motion (2007), 69–84.
- [22] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A time series is worth 64 words: Long-term forecasting with transformers. In Proceedings of the International Conference on Learning Representations.
- [23] Xihao Piao, Zheng Chen, Taichi Murayama, Yasuko Matsubara, and Yasushi Sakurai. 2024. Fredformer: Frequency debiased transformer for time series forecasting. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, 2400–2410.
- [24] Zongjiang Shang and Ling Chen. 2024. MSHyper: Multi-scale hypergraph transformer for long-range time series forecasting. arXiv preprint arXiv:2401.09261 (2024).
- [25] Zongjiang Shang, Ling Chen, Binqing Wu, and Dongliang Cui. 2024. Ada-MSHyper: Adaptive Multi-Scale Hypergraph Transformer for Time Series Forecasting. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, Vol. 37. 33310–33337.
- [26] Bo Shen, Ruyu Shang, and Yapeng Qi. 2024. MvHSTM: A Multi-view Hypergraph Spatio-Temporal Model for Traffic Speed Forecasting. In OpenReview.
- [27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).
- [28] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. 2024. TimeMixer: Decomposable multiscale mixing for time series forecasting. In Proceedings of the International Conference on Learning Representations.
- [29] Senzhang Wang, Meiyue Zhang, Hao Miao, Zhaohui Peng, and Philip S. Yu. 2022. Multivariate correlation-aware spatio-temporal graph convolutional networks

- for multi-scale traffic prediction. ACM Transactions on Intelligent Systems and Technology 13, 3 (2022), 1–22.
- [30] Shun Wang, Yong Zhang, Xuanqi Lin, Yongli Hu, Qingming Huang, and Baocai Yin. 2024. Dynamic hypergraph structure learning for multivariate time series forecasting. IEEE Transactions on Big Data 10, 4 (2024), 556–567.
- [31] Yulong Wang, Yushuo Liu, Xiaoyi Duan, and Kai Wang. 2025. FilterTS: Comprehensive Frequency Filtering for Multivariate Time Series Forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 39. AAAI Press, 21375–21383.
- [32] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. arXiv preprint arXiv:1410.3916 (2014).
- [33] Binqing Wu and Ling Chen. 2023. DSTCGCN: Learning dynamic spatial-temporal cross dependencies for traffic forecasting. arXiv preprint arXiv:2307.00518 (2023).
- [34] Binqing Wu, Weiqi Chen, Wengwei Wang, Bingqing Peng, Liang Sun, and Ling Chen. 2024. WeatherGNN: Exploiting meteo-and spatial-dependencies for local numerical weather prediction bias-correction. In Proceedings of the International Joint Conference on Artificial Intelligence. 2433–2441.
- [35] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. TimesNet: Temporal 2D-variation modeling for general time series analysis. In Proceedings of the International Conference on Learning Representations.
- [36] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Proceedings of the International Conference on Neural Information Processing Systems, Vol. 34. 22419–22430.
- [37] Haixu Wu, Hang Zhou, Mingsheng Long, and Jianmin Wang. 2023. Interpretable weather forecasting for worldwide stations with a unified deep model. *Nature Machine Intelligence* 5, 6 (2023), 602–611.
- [38] 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 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 753–763.
- [39] Kun Yi, Qi Zhang, Wei Fan, Hui He, Liang Hu, Pengyang Wang, Ning An, Long-bing Cao, and Zhendong Niu. 2024. FourierGNN: Rethinking multivariate time series forecasting from a pure graph perspective. In Proceedings of the International Conference on Neural Information Processing Systems. 69638–69660.
- [40] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting? In Proceedings of the AAAI conference on Artificial Intelligence. 11121–11128.
- [41] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer utilizing crossdimension dependency for multivariate time series forecasting. In Proceedings of the International Conference on Learning Representations.
- [42] Yusheng Zhao, Xiao Luo, Wei Ju, Chong Chen, Xian-Sheng Hua, and Ming Zhang. 2023. Dynamic hypergraph structure learning for traffic flow forecasting. In Proceedings of the IEEE International Conference on Data Engineering. 2303–2316.
- [43] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency enhanced decomposed Transformer for long-term series forecasting. In Proceedings of the International Conference on Machine Learning. 27268–27286.
- [44] Chao Zhu, Jing Chen, Rui Zhu, Zhengqiong Wang, Shihan Liu, and Jishu Wang. 2023. Asthgcn: Adaptive spatio-temporal hypergraph convolutional network for traffic forecasting. In 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 972–979.