MSHyper: Multi-Scale Hypergraph Transformer for Long-Range Time Series Forecasting

Zongjiang Shang, Ling Chen, Binqing Wu, Dongliang Cui

***Abstract*—Demystifying the interactions between temporal patterns of different scales is fundamental to precise long-range time series forecasting. However, previous works lack the ability to model high-order interactions. To promote more comprehensive pattern interaction modeling for long-range time series forecasting, we propose a Multi-Scale Hypergraph Transformer (MSHyper) framework. Specifically, a multi-scale feature extraction (MFE) modlue is introduced to map the input sequence into multi-scale embeddings. Then, a multi-scale hypergraph is designed to provide foundations for modeling high-order pattern interactions and a hyperedge graph is built to enhance hypergraph modeling. In addition, a tri-stage message passing (TMP) mechanism is introduced to aggregate pattern information and learn the interaction strength between temporal patterns of different scales. Extensive experimental results on eight real-world datasets demonstrate that MSHyper achieves the state-of-the-art (SOTA) performance across various settings.**

***Index Terms*—Time series forecasting, transformer, multi-scale modeling, hypergraph neural networks.**

I. Introduction

TIME series forecasting has demonstrated its wide applications across many fields (e.g., energy consumption planning [21], [57], traffic and economics prediction [20], [27], [50], [62], and disease propagation forecasting [1], [9], [45], [52]. In these real-world applications, how to use a substantial amount of previous time-series data and extend the forecasting horizon into the far future (i.e., long-range time series forecasting) is quite meaningful, as it can help decision makers to make schedule planning and optimize resource allocation.

Many time series demonstrate complex and diverse temporal patterns of different scales [7], [8], [18], [23], [38], [40]. For example, due to periodic human activities, traffic occupation and electricity consumption show clear daily and weekly patterns. Considering the interactions between these temporal patterns often leads to more accurate forecasting results than analyzing each pattern separately. For example, the morning rush hour on the first workday after a holiday tends to be more congested (the interactions between daily and weekly patterns), while the evening rush hour on the last workday before a long holiday starts earlier. Therefore, how to model complex temporal patterns of different scales and their interactions is a fundamental problem in long-range time series forecasting.

To model temporal patterns of different scales and their interactions, traditional methods (e.g., seasonal ARIMA [5] and Prophet [35]), use decomposition with heuristic priors to obtain temporal patterns of different scales, but cannot model complex non-linear dependencies of time series. Recently, deep neural networks have demonstrated superiority in capturing non-stationary and non-linear dependencies. Temporal convolutional networks (TCNs) [3], [29], recurrent neural networks (RNNs) [30], [51], [59], graph neural networks (GNNs) [6], [13], [42], [58], and Transformers [8], [36], [41], [60] have been used for time series forecasting. To model temporal patterns of different scales, multi-scale Transformer-based methods [26] attempt to build sub-sequences of different scales from the original input sequence but ignore the interactions between temporal patterns of different scales. To address this issue, recent multi-scale Transformer-based methods [14], [28] introduce special structures (e.g., pyramidal structures) between sub-sequences of different scales. These structures model temporal dependencies within a sub-sequence through intra-scale edges, and model the interactions between temporal patterns of different scales through inter-scale edges.

However, these methods only use edges to model pairwise interactions and lack the ability to model high-order interactions (i.e., the simultaneous interactions between multiple temporal patterns). In reality, temporal patterns of different scales co-exist and exhibit high-order interactions, e.g., the peak household electricity consumption during summer weekend afternoons (high-order interactions between daily, weekly, and monthly patterns), as well as the high but relatively stable household electricity consumption during winter weekends.

To address the above issue, we propose MSHyper, a **M**ulti-**S**cale **Hyper**graph Transformer framework for long-range time series forecasting. MSHyper aggregates the input sequence into sub-sequences of different scales, and models high-order interactions between temporal patterns of different scales by building multi-scale hypergraph structures. To the best of our knowledge, MSHyper is the first work that incorporates hypergraph modeling into long-range time series forecasting. The main contributions are as follows:

- We propose a hypergraph and hyperedge graph construction (H-HGC) module that builds the hypergraph according to the temporal proximity rules, which can model intra-scale, inter-scale, and mixed-scale high-order interactions between temporal patterns. In addition, by treating hyperedges as nodes and building edges based on the sequential relationship and association relationship

Corresponding author: Ling Chen.

This work was supported by the Science Foundation of Donghai Laboratory (Grant No. DH-2022ZY0013).

Zongjiang Shang, Ling Chen, Binqing Wu, and Dongliang Cui are with the State Key Laboratory of Blockchain and Data Security, College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: {zongjiangshang, lingchen, binqingwu, runnercdl}@cs.zju.edu.cn).

between nodes, H-HGC also builds the hyperedge graph to enhance hypergraph modeling.
- We propose a tri-stage message passing (TMP) mechanism that has three message passing phases: Node-hyperedge, hyperedge-hyperedge, and hyperedge-node, which can aggregate pattern information and learn the interaction strength between temporal patterns of different scales.
- We conduct extensive experiments on eight real-world time series datasets, and experimental results demonstrate that MSHyper achieves the state-of-the-art (SOTA) performance across various settings.

II. Related Work

In this section, we provide a brief review of the related work, including methods for time series forecasting, multi-scale Transformers, and hypergraph neural networks related studies.

A. Methods for Time Series Forecasting

Time series forecasting methods can be roughly divided into statistical methods and deep neural network-based methods. Statistical methods (e.g., ARIMA [5] and Prophet [35]) follow arbitrary yet simple assumptions, and fail to model complicated temporal dependencies. Recently, deep neural networks show superiority in modeling complicated temporal dependencies [26], [42]. TAMS-RNNs [11] obtains the periodic temporal dependencies through multi-scale recurrent neural networks (RNNs) with different update frequencies. LSTNet [25] utilizes recurrent-skip connections in combination with convolutional neural networks (CNNs) to capture long- and short-term temporal dependencies. Transformer-based methods [12], [24], [39], [41], [60] take advantage of the attention mechanism and achieve great access in modeling long-range temporal dependencies. Reformer [24] approximates the attention value through local-sensitive hashing (LSH), Informer [60] obtains the dominant query by calculating the KL-divergence to realize the approximation of self-attention, and Autoformer [41] introduces an auto-correlation mechanism to operate at the level of sub-sequences. However, the above methods struggle to obtain the complex temporal patterns of long-range time series.

B. Multi-Scale Transformers

Multi-scale or hierarchical Transformers have been proposed in different fields (e.g., computer vision [16], [37], [55], natural language processing [2], [19], [33], and time series forecasting [8], [32], [61]). Multi-scale ViT [55] realizes image recognition by combining multi-scale image feature embeddings and Transformer. Star-Transformer [19] models intra-scale and inter-scale information interactions by introducing the global node embedding. To address the limitations in the expressiveness of a single global node, ETCformer [2] carries out information interactions between the global and local nodes by introducing a set of global node embeddings and setting fixed-length windows. To further extend the ability to model the interactions between temporal patterns of different scales, Pyraformer [28] extends the two-layer structure into multi-scale embeddings, and models the interactions between nodes of different scales through a pyramid graph. Crossformer [56] combines a two-stage attention with a hierarchical encoder-decoder architecture to capture cross-time and cross-dimension interactions. MSGNet [6] combines an adaptive graph convolution and a temporal multi-head attention mechanism to capture multi-scale inter-series correlations. However, existing methods only model pairwise interactions between nodes, ignoring high-order interactions between temporal patterns of different scales.

C. Hypergraph Neural Networks

Hypergraph neural networks (HGNNs) have been proven to be capable of modeling high-order interactions, which have been applied to various fields (e.g., visual object recognition [48], sequential recommendation [47], [49], trajectory prediction [44], stock selection [31], [53], and citation network classification [4], [34]). HGNN [17] and HyperGCN [46] are the first works to apply graph convolution to hypergraphs, which demonstrate the superiority of hypergraphs over ordinary graph neural networks (GNNs) in modeling high-order interactions. MBHT [49] combines hypergraphs with a Transformer framework for the sequential recommendation, which leverages hypergraphs to capture high-order user-item interaction patterns. GroupNet [44] uses multi-scale hypergraphs for trajectory prediction, which leverages topology inference and representation learning to capture the agent patterns and their high-order interactions. STHAN-SR [31] leverages hypergraph and temporal Hawkes attention mechanism for stock selection, which leverages hypergraph network architecture to model inter stock relations of varing types and degrees.

Considering the ability of HGNNs in high-order interaction modeling, we propose a multi-scale hypergraph Transformer framework to model high-order interactions between temporal patterns of different scales. Specifically, a H-HGC module is introduced to build the hypergraph and hyperedge graph. In addition, a TMP mechanism is proposed to aggregate pattern information and learn the interaction strength between temporal patterns of different scales by three message passing phases.

III. Preliminaries

In this section, we first provide the definition of hypergraph and then formulate the problem.

A. Hypergraph

A hypergraph is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{E} = \{e_1, e_2, ..., e_M\}$ is the hyperedge set and $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ is the node set. Each hyperedge $e_m \in \mathcal{E}$ connects a set of nodes in $\mathcal{V}$. The hypergraph $\mathcal{G}$ can be denoted as an incidence matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$, where $\mathbf{H}_{nm} = 1$ if the nth node belongs to the mth hyperedge, else $\mathbf{H}_{nm} = 0$. The degree of the nth node is defined as follows:

$$\mathbf{D}_{v_n} = \sum_{m=1}^{M} \mathbf{H}_{nm}. \tag{1}$$

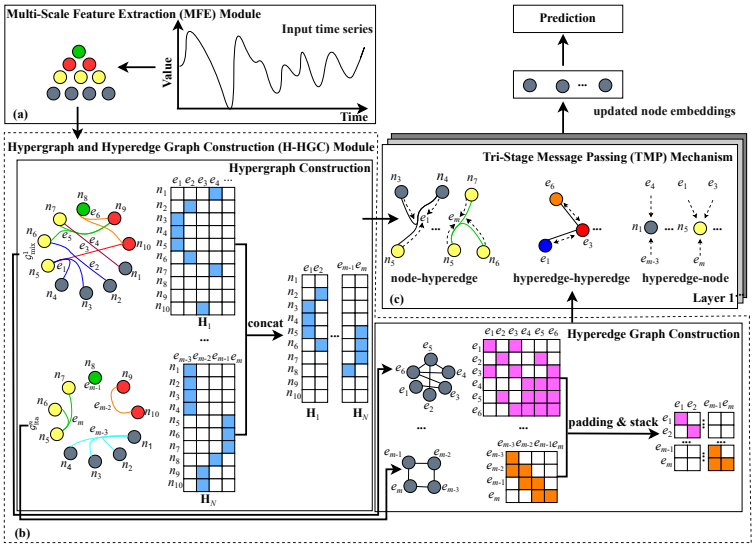


Fig. 1. The framework of MSHyper, which consists of three parts: (a) The HFE module, which maps the input sequence into hierarchical embeddings. (b) The H-HGC module, which provides foundations for modeling high-order interactions between temporal patterns by building the hypergraph and the hyperedge graph. (c) The TMP mechanism, which aggregates pattern information and learns the interaction strength between temporal patterns of different scales.

The degree of the mth hyperedge is defined as follows:

$$\mathbf{D}_{e_m} = \sum_{n=1}^{N} \mathbf{H}_{nm}. \tag{2}$$

The results of node degrees and hyperedge degrees are stored in diagonal matrices $\mathbf{D}_{\mathrm{v}} \in \mathbb{R}^{N\times N}$ and $\mathbf{D}_{\mathrm{e}} \in \mathbb{R}^{M\times M}$, respectively.

B. Problem Statement

The task of long-range time series forecasting is to predict the future H steps given the previous T steps of observed values. Specifically, given the input sequence $\mathbf{X}^{\mathrm{I}}_{1:T} = \left\{\boldsymbol{x}_t \mid \boldsymbol{x}_t \in \mathbb{R}^D, t \in [1,T]\right\}$, where $\boldsymbol{x}_t$ represents the values at time step t, and D is the feature dimension, the prediction task is defined as follows:

$$\widehat{\mathbf{X}}^{\mathrm{O}}_{T+1:T+H} = \mathcal{F}\left(\mathbf{X}^{\mathrm{I}}_{1:T}; \theta\right) \in \mathbb{R}^{H\times D}, \tag{3}$$

where $\widehat{\mathbf{X}}^{\mathrm{O}}_{T+1:T+H}$ denotes the forecasting results, $\mathcal{F}$ denotes the mapping function, and θ denotes the learnable parameters of $\mathcal{F}$.

IV. Methodology

In this section, we give the description of the proposed MSHyper. We first give the framework of MSHyper, and then present the detailed descriptions of each module. In addition, we introduce the detailed process of the tri-stage message passing (TMP) mechanism based on the framework of MSHyper.

A. Framework

As mentioned above, the core of MSHyper is to build multi-scale hypergraph structures, which can explicitly model high-order interactions between temporal patterns of different scales. To accomplish this goal, we first map the input sequence into multi-scale embeddings through the multi-scale feature extraction (MFE) module and then leverage the H-HGC module to build the hypergraph and hyperedge graph. Finally, we employ the TMP mechanism to aggregate pattern information and learn the interaction strength between temporal patterns of different scales. Figure 1 illustrates the framework of MSHyper.

B. Multi-Scale Feature Extraction Module

To get the feature embeddings of different scales, we first map the input sequence into multi-scale embeddings. We use $\mathbf{X}^s = \{\boldsymbol{x}^s_t | \boldsymbol{x}^s_t \in \mathbb{R}^D, t \in [1, h^s]\}$ to represent the subsequence at scale s, where $s = 1, ..., S$ denotes the scale index, and S is the total number of scales. $h^s = \left\lfloor \frac{h^{s-1}}{l^{s-1}} \right\rfloor$ s.t. $s \geq 2$ is the horizon at scale s and l^{s-1} denotes the size of the aggregation window at scale $s-1$. $\mathbf{X}^1 = \mathbf{X}^{\mathrm{I}}_{1:T}$ is the raw input sequence and the aggregation process is defined as follows:

$$\mathbf{X}^{s+1} = Aggregation(\mathbf{X}^s; \theta^s) \in \mathbb{R}^{h^{s+1}\times D}, \tag{4}$$

where $Aggregation$ is the aggregation function (e.g., 1D convolution or average pooling), and θ^s denotes the learnable parameters of the aggregation function at scale s.

C. Hypergraph and Hyperedge Graph Construction Module

Long-range time series data contain a lot of noise and are non-stationary [41]. When modeling high-order interactions between temporal patterns without prior knowledge constraints, the model tends to learn spurious interactions and lacks interpretability. To comprehensively model high-order interactions between temporal patterns of different scales, we build the hypergraph and the hyperedge graph separately by the H-HGC module to model intra-scale, inter-scale, and mixed-scale high-order pattern interactions.

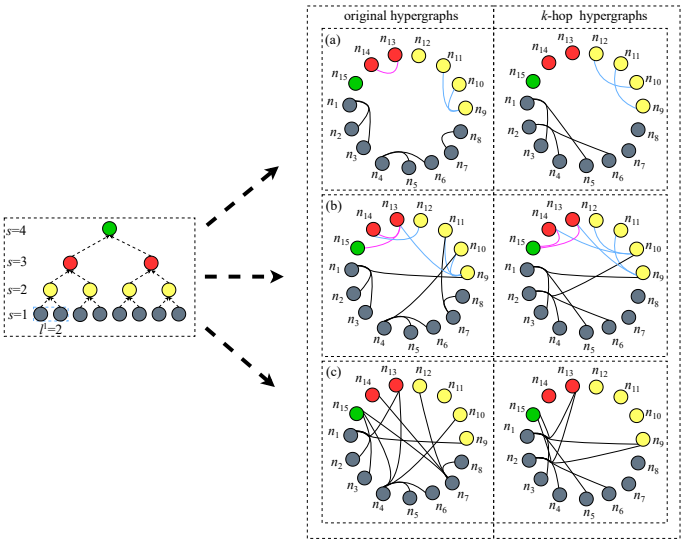


Fig. 2. Hypergraph construction. (a), (b), and (c) represent the intra-scale hypergraph, inter-scale hypergraph, and mixed-scale hypergraph, respectively. In addition to the original connections, we also aggregate information from different ranges of neighbors using k-hop connections.

1) Hypergraph Construction: Despite existing methods [28], [43] being capable of modeling the interactions between temporal patterns of different scales through graph structures, we argue that these methods still face two limitations: (1) These methods fail to model high-order interactions between temporal patterns. (2) These methods limit interactions to the neighboring nodes and thus are incapable of capturing the interactions between nodes that are far away but still show correlations. To address these problems, we build the hypergraph structures according to the temporal proximity rules. As shown in Figure 2, on the one hand, we build different types of hypergraphs to model intra-scale, inter-scale, and mixed-scale high-order interactions. On the other hand, we build the k-hop hypergraph under each type of hypergraph to aggregate information from different ranges of neighbors.

Intra-Scale Hypergraph. As shown in Figure 2(a), in order to capture high-order interactions between intra-scale temporal patterns, we construct the intra-scale hypergraph $\mathcal{G}_{\text{ita}}$, which contains two hypergraphs (i.e., the original intra-scale hypergraph $\mathcal{G}_{\text{o, ita}}$ and k-hop intra-scale hypergraph $\mathcal{G}_{\text{k, ita}}$). We use $\mathcal{G}_{\text{o, ita}} = \{\mathcal{V}, \mathcal{E}_{\text{o, ita}}\}$ to represent the original intra-scale hypergraph, where $\mathcal{E}_{\text{o, ita}} = \{\mathcal{E}^s_{\text{o, ita}}\}_{s\in\{1,\dots,S\}}$ contains original intra-scale hyperedge sets of different scales. The ith hyperedge of scale s $e^s_i \in \mathcal{E}^s_{\text{o, ita}}$ is defined as follows:

$$e^s_i = \left\{v^s_\epsilon, \forall v^s_j \in \mathcal{N}(v^s_\epsilon)\right\} \text{s.t. } 0 < j - \epsilon \leq H_s, \tag{5}$$

where $\epsilon = (i-1)H_s + 1$ is the starting node index under the ith hyperedge of scale s based on original connections. H_s is the number of nodes connected by each hyperedge of scale s and $\mathcal{N}(v^s_\epsilon)$ is the neighboring nodes connected to node v^s_ϵ. Meanwhile, we use $\mathcal{G}_{\text{k, ita}} = \{\mathcal{V}, \mathcal{E}_{\text{k, ita}}\}$ to represent the k-hop intra-scale hypergraph, where $\mathcal{E}_{\text{k, ita}} = \{\mathcal{E}^s_{\text{k, ita}}\}_{s\in\{1,\dots,S\}}$ contains k-hop intra-scale hyperedge sets of different scales. The ith hyperedge of scale s based on k-hop connections $e^s_{i,k} \in \mathcal{E}^s_{\text{k, ita}}$ is defined as follows:

$$e^s_{i,k} = \{v^s_d, v^s_{d+k}, \dots, v^s_{d+(H_s-1)k}\}, \tag{6}$$

where d is the starting node index under the ith hyperedge of scale s based on k-hop connections, which is defined as follows:

$$d = \left\lfloor \frac{i-1}{k} \right\rfloor \times H_s k + (i-1)\%k + 1, \tag{7}$$

where k is the temporal distance between two neighboring nodes. The intra-scale hypergraph $\mathcal{G}_{\text{ita}} = \{\mathcal{V}, \mathcal{E}_{\text{ita}}\}$ based on the original intra-scale hypergraph and the k-hop intra-scale hypergraph is defined as follows:

$$\mathcal{G}_{\text{ita}} = concat(\mathcal{G}_{\text{o, ita}}, \mathcal{G}_{\text{k, ita}}), \tag{8}$$

where $concat$ denotes the concatenation operation. High-order interactions between temporal patterns not only exist between intra-scale temporal patterns, but also between inter-scale temporal patterns (e.g., the interactions between hourly and daily patterns, and the interactions between daily and weekly patterns). In addition, there are temporal pattern interactions across all scales (e.g., the interactions between hourly, daily, and weekly patterns). Therefore, we design the inter-scale hypergraph and mixed-scale hypergraph.

Inter-Scale Hypergraph. As shown in Figure 2(b), the inter-scale hypergraph $\mathcal{G}_{\text{ite}} = \{\mathcal{V}, \mathcal{E}_{\text{ite}}\}$ is obtained by concatenating the original inter-scale hypergraph and k-hop inter-scale hypergraph. We use $\mathcal{G}_{\text{o, ite}} = \{\mathcal{V}, \mathcal{E}_{\text{o, ite}}\}$ to represent the original inter-scale hypergraph, where $\mathcal{E}_{\text{o, ite}} = \{\mathcal{E}^s_{\text{o, ite}}\}_{s\in\{1,\dots,S\}}$ contains original inter-scale hyperedge sets of different scales. The ith hyperedge of scale s $e^s_i \in \mathcal{E}^s_{\text{o, ite}}$ is defined as follows:

$$e^s_i = \left\{v^{s+1}_{\lceil \epsilon/l^s \rceil}, v^s_\epsilon, v^s_{\epsilon+1}, \dots, v^s_{\epsilon+H_s}\right\}, \tag{9}$$

where l^s denotes the aggregation window size at scale s. Then, we use $\mathcal{G}_{\text{k, ite}} = \{\mathcal{V}, \mathcal{E}_{\text{k, ite}}\}$ to represent the k-hop inter-scale hypergraph, where $\mathcal{E}_{\text{k, ite}} = \{\mathcal{E}^s_{\text{k, ite}}\}_{s\in\{1,\dots,S\}}$ contains k-hop inter-scale hyperedge sets of different scales. The ith hyperedge of scale s based on k-hop connections $e^s_{i,k} \in \mathcal{E}^s_{\text{k,ite}}$ is defined as follows:

$$e^s_{i,k} = \{v^{s+1}_{\lceil d/l^s \rceil}, v^s_d, v^s_{d+k}, \dots, v^s_{d+(H_s-1)k}\}. \tag{10}$$

The inter-scale hypergraph $\mathcal{G}_{\text{ite}} = \{\mathcal{V}, \mathcal{E}_{\text{ite}}\}$ based on the original inter-scale hypergraph and the k-hop inter-scale hypergraph is defined as follows:

$$\mathcal{G}_{\text{ite}} = concat(\mathcal{G}_{\text{o, ite}}, \mathcal{G}_{\text{k, ite}}). \tag{11}$$

Mixed-Scale Hypergraph. As shown in Figure 2(c), the mixed-scale hypergraph $\mathcal{G}_{\text{mix}} = \{\mathcal{V}, \mathcal{E}_{\text{mix}}\}$ is obtained by concatenating the original mixed-scale hypergraph and k-hop mixed-scale hypergraph. We use $\mathcal{G}_{\text{o, mix}} = \{\mathcal{V}, \mathcal{E}_{\text{o, mix}}\}$ to represent the original mixed-scale hypergraph, where $\mathcal{E}_{\text{o, mix}}$ denotes the original mixed-scale hyperedge set. The ith hyperedge $e_i \in \mathcal{E}_{\text{o, mix}}$ is defined as follows:

$$e_i = \left\{v^S_{\delta_S}, v^{S-1}_{\delta_{S-1}}, \dots, v^1_{\delta_1}, v^1_{\delta_1+1}, \dots, v^1_{\delta_1+H-1}\right\}, \tag{12}$$

where $\delta_s = \lceil \epsilon / (1 \times \prod_{\alpha=2}^{s} l^\alpha) \rceil$ is the starting node index of scale s under the ith hyperedge based on original connections.

Then, we use $\mathcal{G}_{\text{k, mix}} = \{\mathcal{V}, \mathcal{E}_{\text{k, mix}}\}$ to represent the k-hop mixed-scale hypergraph, where $\mathcal{E}_{\text{k, mix}}$ denotes the k-hop mixed-scale hyperedge set. The ith hyperedge based on k-hop connections $e_{i,k} \in \mathcal{E}_{\text{k, mix}}$ is defined as follows:

$$e_{i,k} = \left\{ v^{S}_{\delta_{S,k}}, v^{S-1}_{\delta_{S-1,k}}, \ldots, v^{1}_{\delta_1}, v^{1}_{\delta_1+k}, \ldots, v^{1}_{\delta_1+(H-1)k} \right\}, \quad (13)$$

where $\delta_{s,k} = \lceil d/(1 \times \prod_{\alpha=2}^{s} l^{\alpha}) \rceil$ is the starting node index of scale s under the ith hyperedge based on k-hop connections.

The mixed-scale hypergraph $\mathcal{G}_{\text{mix}} = \{\mathcal{V}, \mathcal{E}_{\text{mix}}\}$ based on the original mixed-scale hypergraph and the k-hop mixed-scale hypergraph is defined as follows:

$$\mathcal{G}_{\text{mix}} = concat(\mathcal{G}_{\text{o,mix}}, \mathcal{G}_{\text{k,mix}}). \quad (14)$$

The hypergraph construction module generates the hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ by considering the intra-scale hypergraph, the inter-scale hypergraph, and the mixed-scale hypergraph, which is defined as follows:

$$\mathcal{G} = concat(\mathcal{G}_{\text{ita}}, \mathcal{G}_{\text{ite}}, \mathcal{G}_{\text{mix}}). \quad (15)$$

2) Hyperedge Graph Construction: After building the hypergraph, to enhance hypergraph modeling, we build the hyperedge graph to model the interactions between hyperedges. In GNNs, Chen et al. [10] proposes the concept of the line graph to enhance graph modeling through edge interactions. However, since the line graph is unweighted and considers two edges to be correlated only if the target node of one edge is the source node of the other edge, it lacks in characterizing various edge interaction patterns and is not suitable for modeling hyperedge interactions that connect multiple nodes. Thus, we build the hyperedge graph by representing hyperedges in $\mathcal{G}$ as nodes and considering the sequential relationship and association relationship. We define the hyperedge graph as $G = (V, E, \mathbf{A})$, where $V = \{v_{e_i} | v_{e_i} \in \mathcal{E}\}$ is the node set of G and $E = \{(v_{e_i}, v_{e_j}) | v_{e_i}, v_{e_j} \in \mathcal{E}\}$ is the edge set of G. $\mathbf{A}$ is the weighted adjacency matrix defined based on the sequential relationship and association relationship between hyperedges.

Sequential Relationship. Within the intra-scale hypergraph, two hyperedges that connect nodes with temporal order exhibit the sequential relationship. For example, as shown in Figure 3(a), hyperedge e_1 connects the previous three nodes, and hyperedge e_2 connects the next three nodes. Since long-range time series is a collection of data points arranged in chronological order, intuitively, changes (e.g., increase, decrease, or fluctuation) in the values of nodes connected by e_1 may influence the values of nodes connected by e_2. We use $G_{\text{sr}} = (V_{\text{sr}}, E_{\text{sr}}, \mathbf{A}_{\text{sr}})$ to represent the hyperedge graph constructed based on the sequential relationship, where $V_{\text{sr}} = \{v_{e_{\text{sr}}} | v_{e_{\text{sr}}} \in \mathcal{E}_{\text{ita}}\} \in \mathbb{R}^{D_{\text{sr}} \times D}$ is the node set and D_{sr} is the number of intra-scale hyperedges. $E_{\text{sr}} = \{(v_{e_{i,\text{sr}}}, v_{e_{j,\text{sr}}}) | v_{e_{i,\text{sr}}}, v_{e_{j,\text{sr}}} \in \mathcal{E}_{\text{ita}}, 0 \le i - j \le 1\}$ denotes the edge set constructed based on the sequential relationship. As shown in Figure 3(c), the adjacency matrix constructed based on the sequential relationship $\mathbf{A}_{\text{sr}} \in \mathbb{R}^{D_{\text{sr}} \times D_{\text{sr}}}$ is defined as follows:

$$\mathbf{A}_{\text{sr}} = \{\mathbf{A}_{ij,\text{sr}} | \mathbf{A}_{ij,\text{sr}} = 1 \text{ if } (v_{e_{i,\text{sr}}}, v_{e_{j,\text{sr}}}) \in E_{\text{sr}}, \text{ else } 0\}. \quad (16)$$

Association Relationship. Within the inter-scale hypergraph and mixed-scale hypergraph, hyperedges with common nodes have an association relationship. For example, as shown in Figure 3(b), e_5 and e_7 share a common node n_{13}. The value changes of nodes connected by e_5 may influence the values of nodes connected by e_7 through the common node n_{13}. We use $G_{\text{ar}} = (V_{\text{ar}}, E_{\text{ar}}, \mathbf{A}_{\text{ar}})$ to represent the hyperedge graph constructed based on the association relationship, where $V_{\text{ar}} = \{v_{e_{\text{ar}}} | v_{e_{\text{ar}}} \in \{\mathcal{E}_{\text{ite}}, \mathcal{E}_{\text{mix}}\}\} \in \mathbb{R}^{D_{\text{ar}} \times D}$ is the node set of G_{ar} and D_{ar} is the number sum of inter-scale hyperedges and mixed-scale hyperedges. $E_{\text{ar}} = \{(v_{e_{i,\text{ar}}}, v_{e_{j,\text{ar}}}) | v_{e_{i,\text{ar}}}, v_{e_{j,\text{ar}}} \in \{\mathcal{E}_{\text{ite}}, \mathcal{E}_{\text{mix}}\}, |v_{e_{i,\text{ar}}} \cap v_{e_{j,\text{ar}}}| \ge 1\}$ denotes the edge set of G_{ar}. As shown in Figure 3(d), the adjacency matrix based on the association relationship $\mathbf{A}_{\text{ar}} \in \mathbb{R}^{D_{\text{ar}} \times D_{\text{ar}}}$ is defined as follows:

$$\mathbf{A}_{\text{ar}} = \{\mathbf{A}_{ij,\text{ar}} | \mathbf{A}_{ij,\text{ar}} = 1 \text{ if } (v_{e_{i,\text{ar}}}, v_{e_{j,\text{ar}}}) \in E_{\text{ar}}, \text{ else } 0\}. \quad (17)$$

The adjacency matrix $\mathbf{A}$ based on the above two relationships is defined as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{\text{sr}} & \mathbf{\Gamma}_1 \\ \mathbf{\Gamma}_2 & \mathbf{A}_{\text{ar}} \end{bmatrix} \in \mathbb{R}^{D_{\varphi} \times D_{\varphi}}, \quad (18)$$

where $\mathbf{\Gamma}_1 \in \mathbb{R}^{D_{\text{sr}} \times D_{\text{ar}}}$ and $\mathbf{\Gamma}_2 \in \mathbb{R}^{D_{\text{ar}} \times D_{\text{sr}}}$ are matrices consisting entirely of zeros, and D_{φ} is the sum of D_{ar} and D_{sr}.

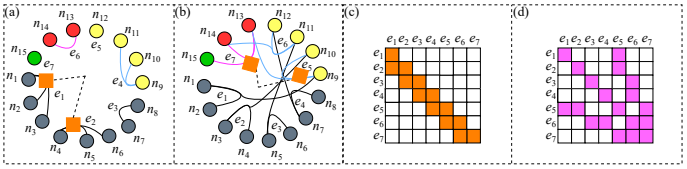


Fig. 3. Hyperedge graph construction. (a) and (b) are the sequential relationship and association relationship, respectively. (c) and (d) are the constructed adjacency matrix based on the sequential relationship and association relationship, respectively.

D. Tri-Stage Message Passing Mechanism

After building the hypergraph and hyperedge graph, to aggregate pattern information and learn the interaction strength between temporal patterns of different scales, we propose a TMP mechanism, which contains the node-hyperedge, hyperedge-hyperedge, and hyperedge-node phases.

Node-Hyperedge Phase. Given the sequences based on the MFE module $\mathbf{X} = \{\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^S\} \in \mathbb{R}^{N \times D}$, where N denotes the number sum of input time steps and aggregated feature values of different scales. We first get the initialized node embeddings $\boldsymbol{\mathcal{V}} = f(\mathbf{X}) \in \mathbb{R}^{N \times D}$, where f can be implemented by the multi-layer perceptron (MLP). As shown in Figure 1(c), we get the initialized hyperedge embeddings by the aggregation operation based on the hypergraph $\mathcal{G}$. Specifically, for the ith hyperedge $e_i \in \mathcal{E}$, its initialized embedding is defined as follows:

$$\boldsymbol{v}_{e_i} = \sum_{v_j \in \mathcal{N}(e_i)} \boldsymbol{v}_j \in \mathbb{R}^{D}, \quad (19)$$

where $\mathcal{N}(e_i)$ denotes neighboring nodes connected by e_i.

Hyperedge-Hyperedge Phase. After getting initialized hyperedge embeddings, we proceed to update their embeddings

through the constructed hyperedge graph. Specifically, for the given initialized hyperedge embeddings $\mathbf{V} = \{\boldsymbol{v}_{e_i} | \boldsymbol{v}_{e_i} \in \mathcal{E}\} \in \mathbb{R}^{M \times D}$, we transform it into query $\widetilde{\mathbf{Q}} = \mathbf{V}\mathbf{W}^{\mathrm{q}}$, key $\widetilde{\mathbf{K}} = \mathbf{V}\mathbf{W}^{\mathrm{k}}$, and value $\widetilde{\mathbf{V}} = \mathbf{V}\mathbf{W}^{\mathrm{v}}$, where $\mathbf{W}^{\mathrm{q}}$, $\mathbf{W}^{\mathrm{k}}$, and $\mathbf{W}^{\mathrm{v}}$ are learnable weight matrices. For the ith row $\widetilde{\boldsymbol{q}}_i \in \widetilde{\mathbf{Q}}$, the updated hyperedge embedding $\widetilde{\boldsymbol{v}}_{e_i} \in \widetilde{\mathbf{V}}_{\mathrm{e}}$ is defined as follows:

$$\begin{aligned} \widetilde{\boldsymbol{v}}_{e_i} &= \sum_{j=1}^{M} \frac{exp(e_{ij})}{\Sigma_{\ell=1}^{M} exp(e_{i\ell})} \widetilde{\boldsymbol{v}}_j \\ e_{ij} &= \frac{\widetilde{\boldsymbol{q}}_i \widetilde{\boldsymbol{k}}_j^T}{\sqrt{D}} - (1 - \mathbf{A}_{ij})\mathrm{C}, \end{aligned} \tag{20}$$

where $\widetilde{\boldsymbol{k}}_j^T$ denotes the transpose of the jth row in $\widetilde{\mathbf{K}}$ and $\widetilde{\boldsymbol{v}}_j$ denotes the jth row in $\widetilde{\mathbf{V}}$. $\mathbf{A}_{ij} \in \mathbf{A}$ is a binary value and C is a large constant.

Hyperedge-Node Phase. After obtaining the updated hyperedge embeddings, we update the node embeddings by considering all the related hyperedges. Considering the constructed hypergraph $\mathcal{G}$, we use the hypergraph convolution to update the node embeddings. Specifically, the symmetric normalized hypergraph Laplacian convolution is defined as follows:

$$\widetilde{\boldsymbol{\mathcal{V}}} = \sigma(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}\boldsymbol{\mathcal{V}}\mathbf{P}) \in \mathbb{R}^{N \times D_c}, \tag{21}$$

where $\widetilde{\boldsymbol{\mathcal{V}}}$ is the output of the hypergraph convolution and D_c is the output dimension. $\mathbf{P} \in \mathbb{R}^{D \times D_c}$ denotes the learnable parameters and σ is the activation function (e.g., LeakyReLU and ELU). To capture the interaction strength of each node $v_i \in \boldsymbol{\mathcal{V}}$ and its related hyperedges, we dynamically update $\mathbf{H}$ using the node embedding and updated hyperedge embeddings, which is defined as follows:

$$\mathbf{H}_{ij}^{\mathrm{att}} = \frac{exp(\sigma(f_t[\boldsymbol{v}_i, \widetilde{\boldsymbol{v}}_{e_j}]))}{\sum_{k \in \mathcal{N}_i} exp(\sigma(f_t[\boldsymbol{v}_i, \widetilde{\boldsymbol{v}}_{e_k}])))}, \tag{22}$$

where $[.,.]$ denotes the concatenation operation of the node and its related hyperedges. f_t is a trainable MLP, and $\mathcal{N}_i$ is the neighboring hyperedges connected to v_i, which can be accessed using the constructed hypergraph $\mathcal{G}$. Then, we use Equation 21 to aggregate pattern information of different scales by replacing $\mathbf{H}$ with $\mathbf{H}^{\mathrm{att}}$. The multi-head attention mechanism is also used to stabilize the training process, which is defined as follows:

$$\widetilde{\boldsymbol{\mathcal{V}}} = \bigoplus_{j=1}^{\mathcal{J}} (\sigma(\mathbf{D}_v^{-1/2}\mathbf{H}_j^{\mathrm{att}}\mathbf{D}_e^{-1}\mathbf{H}_j^{\mathrm{att}^T}\mathbf{D}_v^{-1/2}\boldsymbol{\mathcal{V}}\mathbf{P}_j)), \tag{23}$$

where $\bigoplus$ is the aggregation function used for combing the outputs of multi-head (e.g., concatenation or average pooling). $\mathbf{H}_j^{\mathrm{att}}$ and $\mathbf{P}_j$ are the enriched incidence matrix and the learnable weight matrix of the jth head, respectively. $\mathcal{J}$ is the number of heads.

E. Objective Function

After obtaining the updated node embeddings encoded by the multi-scale hypergraph, we concatenate the last node embeddings of each sub-sequence from different scales and then put them into a linear layer for prediction. We choose MSE as our objective function, which is defined as follows:

$$\mathcal{L} = \frac{1}{H} \left\| \widehat{\mathbf{X}}_{T+1:T+H}^{\mathrm{O}} - \mathbf{X}_{T+1:T+H}^{\mathrm{O}} \right\|_2^2, \tag{24}$$

where $\widehat{\mathbf{X}}_{T+1:T+H}^{\mathrm{O}}$ and $\mathbf{X}_{T+1:T+H}^{\mathrm{O}}$ are forecasting results and ground truth, respectively.

TABLE I
DATASET STATISTICS.

Datasets	#Variates	ACF values	Frequency	Information
Flight	7	0.73	Hourly	Flight
Weather	21	0.35	10 minutes	Meteorological
ETTh1, ETTh2	7	0.46, 0.41	Hourly	Temperature
ETTm1, ETTm2	7	0.39, 0.09	15 minutes	Temperature
Electricity	321	0.84	Hourly	Electricity
Exchange-Rate	8	0.02	Daily	Economy

F. Complexity Analysis

The time complexity of MSHyper consists of three main parts. For the node-hyperedge phase, the time complexity is $\mathcal{O}(M)$, where M is the number of hyperedges. For the hyperedge-hyperedge phase, the time complexity is $\mathcal{O}(M^2)$. For the hyperedge-node phase, since $\mathbf{D}_v$ and $\mathbf{D}_e$ are diagonal matrices, and the sparse operation in *torch_geometric* of PyTorch, the time complexity is $\mathcal{O}(MN)$, where N is the number of nodes. In practical operation, since the large aggregation window l^s and a hyperedge can connect multiple nodes (large H_s), M is smaller than N. As a result, the total complexity of MSHyper is bounded by $\mathcal{O}(MN)$.

V. EXPERIMENTS

In this section, we first present experiments to verify the performance of MSHyper on eight public time series forecasting datasets. Then, we conduct ablation studies, case studies, and parameter studies to verify the effect of different module designs and parameter choices. In addition, we provide computation cost to verify the effectiveness of MSHyper.

A. Datasets and Settings

Datasets. We conduct experiments on eight commonly used long-range time series forecasting datasets, including *ETT* (*ETTh* and *ETTm*), *Electricity*, *Weather*, *Flight*, and *Exchange-Rate* datasets. Table I shows the summarized dataset statistics. The auto-correlation function (ACF) values is used to evaluate the correlation between a time series and its lagged values [22]. Higher ACF values typically indicates greater predictability. The detailed descriptions about the eight datasets are given as follows:

- *ETT* [60]: This dataset contains the oil temperature and load data collected by electricity transformers, including *ETTh* (*ETTh1* and *ETTh2*) and *ETTm* (*ETTm1* and *ETTm2*), which are sampled hourly and every 15 minutes, respectively.
- *Electricity*[1]: This dataset contains the electricity consumption of 321 clients from the UCI Machine Learning Repository, which is sampled hourly.

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

TABLE II
SETTINGS OF NNI.

	Parameters	Choise
Search space	Batch size	{8, 16, 32, 64, 128}
	Dropout rate	{0.05, 0.1, 0.5}
	Dimension of model	{64, 128, 256, 512}
	Initial learning rate	{0.0001, 0.005, 0.01, 0.1}
Configures	Max trial number	120
	Optimization algorithm	Tree-structured Parzen Estimator
	Early stopping strategy	Curvefitting

- *Flight* [6]: This dataset contains changes in flight data from 7 major European airports provided by OpenSky[2], which is sampled hourly.
- *Weather*[3]: This dataset contains 21 meteorological measurements data from the Weather Station of the Max Planck Biogeochemistry, which is sampled every 10 minutes.
- *Exchange-Rate* [25]: This dataset contains the exchange rate data from 8 foreign countries, which is sampled daily.

Following existing works [6], [28], [56], [60], we split each dataset into training, validation, and testing sets based on chronological order. The ratio is 6:2:2 for the *ETT* dataset and 7:2:1 for the others.

Experimental Settings. MSHyper is implemented in Python with PyTorch 1.13.1 and trained/tested on a single NVIDIA Geforce RTX 3090 GPU, and the source code of MSHyper is released on GitHub [4]. We repeat all experiments 3 times and use the mean of the metrics as the final results. Following existing works [6], [29], [54], we use instance normalization to normalize all datasets. Adam is set as the optimizer. The aggregation windows are set to 4 for *ETT* dataset and 3 for other datasets. S and H in MSHyper are set to 4 in all experiments. For other hyperparameters, we use Neural Network Intelligence (NNI)[5] toolkit to automatically search the best hyperparameters, which can greatly reduce computation cost compared to the grid search approach. The detailed search space of hyperparameters and the configurations of NNI are given in Table II.

For the long-range time series forecasting, we have two kinds of settings (i.e., multivariate settings and univariate settings), which are used to forecast all feature dimensions and the last feature dimension, respectively. Following existing works [54], [61], we use all eight datasets for multivariate long-range time series forecasting and *ETT* dataset for univariate long-range time series forecasting, respectively.

Evaluation metrics. Mean Square Error (MSE) and Mean Absolute Error (MAE) are used as evaluation metrics, which are defined as follows:

$$\begin{aligned} \mathrm{MSE} &= \frac{1}{H}\left\|\widehat{\mathbf{X}}^{\mathrm{O}}_{T+1:T+H} - \mathbf{X}^{\mathrm{O}}_{T+1:T+H}\right\|^2 \\ \mathrm{MAE} &= \frac{1}{H}\left\|\widehat{\mathbf{X}}^{\mathrm{O}}_{T+1:T+H} - \mathbf{X}^{\mathrm{O}}_{T+1:T+H}\right\|, \end{aligned} \tag{25}$$

where T and H are the input and output lengths, $\widehat{\mathbf{X}}^{\mathrm{O}}_{T+1:T+H}$ and $\mathbf{X}^{\mathrm{O}}_{T+1:T+H}$ are forecasting results and ground truth, respectively. Lower MSE and MAE results mean better performance.

[2]https://opensky-network.org/
[3]https://www.bgc-jena.mpg.de/wetter/
[4]https://github.com/shangzongjiang/MSHyper
[5]https://nni.readthedocs.io/en/latest/

B. Methods for Comparison

Baselines. We carefully choose nine well-acknowledged forecasting models as our baselines, including (1) Transformer-based methods: Crossformer [56], Pyraformer [28], Autoformer [41], and Informer [60]; (2) Graph-based methods: MSGNet [6] and MTGNN [42]; (3) Linear-based methods: DLinear [54] and TiDE [15]; and (4) TCN-based methods: TimesNet [29]. The detailed descriptions about the nine baselines are as follows:

- MSGNet [6]: It captures inter-scale correlations at different scales by leveraging frequency domain analysis and adaptive graph convolution.
- TimesNet [29]: It transforms the 1D input sequence into 2D tensors and uses 2D convolution kernels to capture mixed-scale pattern interactions.
- Crossformer [56]: It combines a two-stage attention with a hierarchical encoder-decoder architecture to capture cross-time and cross-dimension interactions.
- DLinear [54]: It decomposes the input sequence into seasonal and trend components and employs two one-layer linear layers to model each component separately.
- TiDE [15]: It uses dense MLP-based encoder-decoder architectures to handle covariates and non-linear dependencies.
- Pyraformer [28]: It extends the two-layer structure into multi-scale embeddings, which models the interactions between nodes of different scales through a pyramid graph.
- Autoformer [41]: It introduces an auto-correlation mechanism to realize seasonal-trend decomposition at the level of sub-sequence.
- Informer [60]: It obtains the dominant query by calculating the KL-divergence to reduce the computational complexity.
- MTGNN [42]: It utilizes a graph learning module to measure inter-variable dependencies and a temporal convolution module to capture temporal pattern interactions.

C. Main Results

Multivariate Results. Table III shows the multivariate long-range time series forecasting results of MSHyper compared with baselines on all eight datasets, from which we can discern the following tendencies:

- MSHyper gets the SOTA results on all eight datasets and even achieves better performance on the datasets with low auto-correlation function (ACF) values (e.g., *Exchange-Rate* and *ETTm1* datasets). Specifically, MSHyper reduces prediction errors by an average of 4.06% and 5.27% over the best baseline in MSE and MAE, respectively. We attribute this to the reason that MSHyper can get multi-scale embeddings and capture high-order interactions between temporal patterns of different scales. In

TABLE III
MULTIVARIATE LONG-RANGE TIME SERIES FORECASTING RESULTS ON EIGHT REAL-WORLD DATASETS. THE INPUT LENGTH IS SET AS $I = 96$, AND THE PREDICTION LENGTH O IS SET AS 96, 192, 336, AND 720. THE BEST RESULTS ARE **BOLDED** AND THE SECOND BEST RESULTS ARE UNDERLINED.

Models		MSHyper (Ours)		MSGNet (2024)		TimesNet (2023)		Crossformer* (2023)		DLinear (2023)		TiDE* (2023)		Pyraformer* (2021)		Autoformer (2021)		Informer (2021)		MTGNN (2020)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.383**	**0.392**	0.390	0.411	0.384	0.402	0.418	0.438	0.386	0.400	0.479	0.464	0.664	0.510	0.449	0.459	0.865	0.713	0.440	0.450
	192	**0.435**	**0.423**	0.442	0.442	0.436	0.429	0.539	0.517	0.437	0.432	0.525	0.492	0.790	0.687	0.500	0.482	1.008	0.792	0.449	0.433
	336	**0.480**	**0.445**	**0.480**	0.468	0.491	0.469	0.709	0.638	0.481	0.459	0.565	0.515	0.887	0.738	0.521	0.496	1.107	0.809	0.598	0.554
	720	**0.482**	**0.467**	0.494	0.488	0.521	0.500	0.733	0.639	0.519	0.516	0.594	0.558	0.976	0.784	0.514	0.512	1.181	0.865	0.685	0.620
ETTh2	96	**0.291**	**0.338**	0.328	0.371	0.340	0.374	0.425	0.463	0.333	0.387	0.400	0.440	1.392	0.954	0.346	0.388	3.755	1.525	0.496	0.509
	192	**0.376**	**0.391**	0.402	0.414	0.402	0.414	0.473	0.500	0.477	0.476	0.528	0.509	3.515	1.561	0.456	0.452	5.602	1.931	0.716	0.616
	336	**0.419**	**0.430**	0.435	0.443	0.452	0.452	0.581	0.562	0.594	0.541	0.643	0.571	4.471	1.835	0.482	0.486	4.721	1.835	0.718	0.614
	720	**0.417**	**0.435**	**0.417**	0.441	0.462	0.468	0.775	0.665	0.831	0.657	0.874	0.679	4.190	1.770	0.515	0.511	3.647	1.625	1.161	0.791
ETTm1	96	0.331	**0.350**	**0.319**	0.366	0.338	0.375	0.361	0.403	0.345	0.372	0.364	0.387	0.535	0.510	0.505	0.475	0.672	0.571	0.381	0.415
	192	**0.374**	**0.373**	0.376	0.397	**0.374**	0.387	0.387	0.422	0.380	0.389	0.398	0.404	0.580	0.549	0.553	0.496	0.795	0.669	0.442	0.451
	336	**0.408**	**0.395**	0.417	0.422	0.410	0.411	0.605	0.572	0.413	0.413	0.428	0.425	0.736	0.646	0.621	0.537	1.212	0.871	0.475	0.475
	720	**0.473**	**0.431**	0.481	0.458	**0.473**	0.450	0.703	0.645	0.474	0.453	0.487	0.461	1.056	0.782	0.671	0.561	1.166	0.823	0.531	0.507
ETTm2	96	0.179	**0.257**	**0.177**	0.262	0.187	0.267	0.275	0.358	0.193	0.292	0.207	0.305	0.361	0.450	0.255	0.339	0.365	0.453	0.240	0.343
	192	**0.247**	**0.305**	**0.247**	0.307	0.249	0.309	0.345	0.400	0.284	0.362	0.290	0.364	0.720	0.667	0.281	0.340	0.533	0.563	0.398	0.454
	336	**0.309**	**0.344**	0.312	0.346	0.321	0.351	0.657	0.528	0.369	0.427	0.377	0.422	1.294	0.882	0.339	0.372	1.363	0.887	0.568	0.555
	720	**0.401**	**0.398**	0.414	0.403	0.408	0.403	1.208	0.753	0.554	0.522	0.558	0.524	4.147	1.547	0.433	0.432	3.379	1.338	1.072	0.767
Flight	96	**0.155**	**0.260**	0.183	0.301	0.237	0.350	0.410	0.449	0.221	0.337	0.247	0.440	0.452	0.508	0.204	0.319	0.333	0.405	0.196	0.316
	192	**0.155**	**0.258**	0.189	0.306	0.224	0.337	0.503	0.512	0.220	0.336	0.303	0.493	0.458	0.507	0.200	0.314	0.358	0.421	0.272	0.379
	336	**0.164**	**0.266**	0.206	0.320	0.289	0.394	0.544	0.532	0.229	0.342	0.354	0.533	0.483	0.522	0.201	0.318	0.398	0.446	0.260	0.369
	720	**0.202**	**0.299**	0.253	0.358	0.310	0.408	0.702	0.635	0.263	0.366	0.572	0.689	0.521	0.536	0.345	0.426	0.476	0.484	0.390	0.449
Weather	96	**0.157**	**0.198**	0.163	0.212	0.172	0.220	0.232	0.302	0.196	0.255	0.202	0.261	0.211	0.295	0.266	0.336	0.300	0.384	0.171	0.231
	192	**0.207**	**0.244**	0.212	0.254	0.219	0.261	0.371	0.410	0.237	0.296	0.242	0.298	0.240	0.316	0.307	0.367	0.598	0.544	0.215	0.274
	336	**0.265**	**0.286**	0.272	0.299	0.280	0.306	0.495	0.515	0.283	0.335	0.287	0.335	0.295	0.356	0.359	0.395	0.578	0.523	0.266	0.313
	720	**0.342**	**0.339**	0.350	0.348	0.365	0.359	0.526	0.542	0.345	0.381	0.351	0.386	0.368	0.407	0.419	0.428	1.059	0.741	0.344	0.375
Exchange	96	**0.083**	**0.199**	0.102	0.230	0.107	0.234	0.256	0.367	0.088	0.218	0.094	0.218	1.468	1.002	0.197	0.323	0.847	0.752	0.267	0.378
	192	**0.173**	**0.294**	0.195	0.317	0.226	0.344	0.470	0.509	0.176	0.315	0.184	0.307	1.583	1.061	0.300	0.369	1.204	0.895	0.590	0.578
	336	**0.310**	**0.411**	0.359	0.436	0.367	0.448	1.268	0.883	0.313	0.427	0.349	0.431	1.733	1.110	0.509	0.524	1.672	1.036	0.939	0.749
	720	0.846	**0.692**	0.940	0.738	0.964	0.746	1.767	1.068	**0.839**	0.695	0.852	0.698	2.000	1.184	1.447	0.941	2.478	1.310	1.107	0.834
Electricity	96	**0.152**	**0.252**	0.165	0.274	0.168	0.272	0.219	0.314	0.197	0.282	0.237	0.329	0.372	0.441	0.201	0.317	0.274	0.368	0.211	0.305
	192	**0.171**	**0.271**	0.184	0.292	0.184	0.289	0.231	0.322	0.196	0.285	0.236	0.330	0.370	0.440	0.222	0.334	0.296	0.386	0.225	0.319
	336	**0.187**	**0.284**	0.195	0.302	0.198	0.300	0.246	0.337	0.209	0.301	0.249	0.344	0.368	0.440	0.231	0.338	0.300	0.394	0.247	0.340
	720	0.224	**0.316**	0.231	0.332	**0.220**	0.320	0.280	0.363	0.245	0.333	0.284	0.373	0.373	0.446	0.254	0.361	0.373	0.439	0.287	0.373

* indicates that some methods do not have uniform prediction lengths with other methods. To ensure a fair comparison, we utilize their official code and adjust prediction lengths. Other results are from MSGNet.

addition, prediction errors increase smoothly and slowly when the forecasting horizon is increased, which means that the MSHyper retains better long-range robustness. These observations provide empirical guidance for the success of using MSHyper in multivariate long-range time series forecasting.

- Traditional transformer-based methods (i.e., Informer, Autoformer, and Pyraformer) exhibit relatively poor predictive performance. This is because time series forecasting requires modeling temporal pattern interactions, while vanilla attention or simplistic decomposition techniques are insufficient in capturing such interactions.
- Although using different backbones, existing multi-scale decomposition methods (i.e., MTGNN, DLinear, TiDE, Crossformer, TimesNet, and MSGNet) all achieve competitive performance. Specially, by considering multi-scale pattern interactions on the decomposed sequences, MSGNet and TimesNet achieve promising forecasting results. However, they fail to consider high-order pattern interactions and get worse performance than MSHyper in most cases.

Univariate Results. Table IV summarizes the univariate long-range time series forecasting results of MSHyper compared with baselines on *ETT* dataset, from which we can discern the following tendencies:

- MSHyper still achieves the SOTA performance for univariate long-range time series forecasting, and the prediction errors increase steadily and slowly when increasing the forecasting horizon, which demonstrates the effectiveness of MSHyper in improving the capability of univariate long-range time series forecasting.
- Although MLP-based methods perform better than traditional Transformer-based methods, they fail to capture multi-scale pattern interactions and get worse performance than MSGNet, TimesNet, and MSHyper.
- MSGNet and TimesNet are the best baselines that use multi-head attention mechanism and 2D convolution kernels to learn multi-scale pattern interactions, respectively. However, they only consider one type of multi-scale (i.e., intra-scale or mixed-scale) pattern interactions and get worse performance than MSHyper. In contrast, MSHyper leverages multi-scale hypergraph structures to capture intra-scale, inter-scale, and mixed-scale pattern interactions.

D. Ablation Studies

We conduct ablation studies to verify the impact of different components on long-range time series forecasting. All ablation studies are conducted on *ETTh1* dataset.

Multi-Scale Feature Extraction. Our multi-scale feature extraction is implemented by 1D convolution. To investigate the effect of the multi-scale feature extraction, we conduct ablation studies by carefully designing the following two variants:

- MSHyper-avg: It replaces 1D convolution with avg-pooling to extract multi-scale features.
- MSHyper-max: It replaces 1D convolution with max-pooling to extract multi-scale features.

The experimental results are shown in Table V, from which we can observe that MSHyper achieves the best performance

TABLE IV
UNIVARIATE LONG-RANGE TIME SERIES FORECASTING RESULTS ON *ETT* DATASET. THE INPUT LENGTH IS SET AS I=96, AND THE PREDICTION LENGTH O IS SET AS 96, 192, 336, AND 720. THE BEST RESULTS ARE **BOLDED** AND THE SECOND BEST RESULTS ARE UNDERLINED.

Models		MSHyper (Ours)		MSGNet (2024)		TimesNet (2023)		Crossformer (2023)		DLinear (2023)		TiDE (2023)		Pyraformer (2021)		Autoformer (2021)		Informer (2021)		MTGNN (2020)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.055**	**0.178**	0.059	0.186	0.058	0.185	0.076	0.216	0.061	0.184	0.068	0.193	0.461	0.621	0.449	0.459	0.865	0.713	0.382	0.560
	192	**0.076**	**0.209**	0.078	0.214	0.077	0.213	0.085	0.225	0.083	0.214	0.083	0.216	0.620	0.731	0.500	0.482	1.008	0.792	0.470	0.624
	336	**0.090**	**0.237**	0.096	0.242	0.104	0.249	0.106	0.257	0.100	0.245	0.098	0.244	0.494	0.650	0.521	0.496	1.107	0.809	0.381	0.553
	720	0.098	0.246	0.126	0.278	**0.086**	**0.232**	0.128	0.287	0.197	0.369	0.178	0.347	0.697	0.784	0.514	0.512	1.181	0.865	0.313	0.498
ETTh2	96	**0.127**	0.274	0.144	0.295	0.141	0.293	0.138	0.289	0.128	**0.271**	0.145	0.296	0.405	0.519	0.358	0.397	3.755	1.525	0.516	0.620
	192	**0.178**	**0.327**	0.198	0.353	0.192	0.345	0.188	0.341	0.182	0.328	0.179	0.346	1.528	1.102	0.456	0.452	5.602	1.931	1.021	0.902
	336	**0.217**	0.372	0.224	0.381	0.240	0.394	0.229	0.384	0.231	0.377	0.226	**0.367**	1.546	1.110	0.482	0.486	4.721	1.835	1.496	1.108
	720	0.228	0.383	**0.222**	**0.380**	0.238	0.390	0.250	0.404	0.322	0.464	0.319	0.462	1.302	1.016	0.515	0.511	3.647	1.625	1.969	1.306
ETTm1	96	**0.028**	**0.125**	**0.028**	0.126	0.029	0.127	0.071	0.193	0.034	0.135	0.034	0.136	0.089	0.239	0.505	0.475	0.672	0.571	0.112	0.271
	192	**0.042**	**0.156**	0.044	0.160	0.047	0.163	0.106	0.245	0.054	0.174	0.055	0.174	0.214	0.388	0.553	0.496	0.795	0.669	0.217	0.393
	336	**0.056**	**0.181**	0.059	0.187	0.058	0.187	0.104	0.287	0.075	0.204	0.073	0.202	0.332	0.505	0.621	0.537	1.212	0.871	0.389	0.556
	720	**0.078**	**0.215**	0.081	0.218	0.081	0.220	0.183	0.334	0.113	0.254	0.100	0.236	0.543	0.662	0.671	0.561	1.166	0.823	0.633	0.719
ETTm2	96	**0.069**	**0.189**	0.074	0.206	0.075	0.201	0.071	0.194	0.070	0.191	0.073	0.197	0.167	0.314	0.255	0.339	0.365	0.453	0.156	0.313
	192	**0.099**	**0.235**	0.108	0.249	0.107	0.248	0.105	0.242	0.104	0.238	0.107	0.242	0.550	0.603	0.281	0.340	0.533	0.563	0.194	0.341
	336	**0.127**	**0.270**	0.140	0.287	0.142	0.289	0.135	0.281	0.134	0.277	0.138	0.282	0.873	0.757	0.339	0.372	1.363	0.887	0.816	0.809
	720	**0.176**	**0.326**	0.194	0.343	0.206	0.352	0.189	0.340	0.186	0.331	0.188	0.333	2.449	1.400	0.422	0.419	3.379	1.338	1.193	0.884

TABLE V
RESULTS OF DIFFERENT MULTI-SCALE FEATURE EXTRACTION METHODS.

Methods		Predicition Length			
		96	192	336	720
MSHyper-max	MSE	0.384	0.437	0.483	0.487
	MAE	0.393	0.426	0.446	**0.464**
MSHyper-avg	MSE	**0.382**	0.436	**0.478**	0.485
	MAE	0.393	0.426	0.447	0.468
MSHyper	MSE	0.383	**0.435**	0.480	**0.482**
	MAE	**0.392**	**0.423**	**0.445**	0.467

TABLE VI
RESULTS OF DIFFERENT MULTI-SCALE HYPERGRAPH CONSTRUCTION METHODS.

Methods		Predicition Length			
		96	192	336	720
MSHyper-FCG	MSE	0.699	0.723	0.732	0.740
	MAE	0.562	0.574	0.583	0.622
MSHyper-PG	MSE	0.453	0.494	0.532	0.534
	MAE	0.449	0.471	0.493	0.551
MSHyper-SSH	MSE	0.432	0.453	0.510	0.518
	MAE	0.427	0.461	0.470	0.491
MSHyper	MSE	**0.383**	**0.435**	**0.480**	**0.482**
	MAE	**0.392**	**0.423**	**0.445**	**0.467**

in almost all cases, which indicates that convolution kernels can extract more complex features.

Multi-Scale Hypergraph. To investigate the effectiveness of the multi-scale hypergraph, we conduct ablation studies by carefully designing the following three variants:

- MSHyper-FCG: It replaces the H-HGC model with the fully-connected graph.
- MSHyper-PG: It replaces the H-HGC model with the pyramid graph used in Pyraformer [28].
- MSHyper-SSH: It connects the input sequence with intra-scale hyperedges, and thus the multi-scale hypergraph turns to the single-scale hypergraph.

The experimental results are shown in Table VI, from which we can observe that MSHyper performs the best in all cases, which indicates the importance of multi-scale hypergraph in modeling high-order interactions between temporal patterns of different scales.

Hyperedge Graph. To investigate the effectiveness of the hyperedge graph, we conduct ablation studies by carefully designing the following three variants:

- MSHyper-w/o AR: It removes the association relationship of the hyperedge graph.
- MSHyper-w/o SR: It removes the sequential relationship of the hyperedge graph.
- MSHyper-w/o H-H: It removes the hyperedge graph (i.e., without the hyperedge-hyperedge phase).

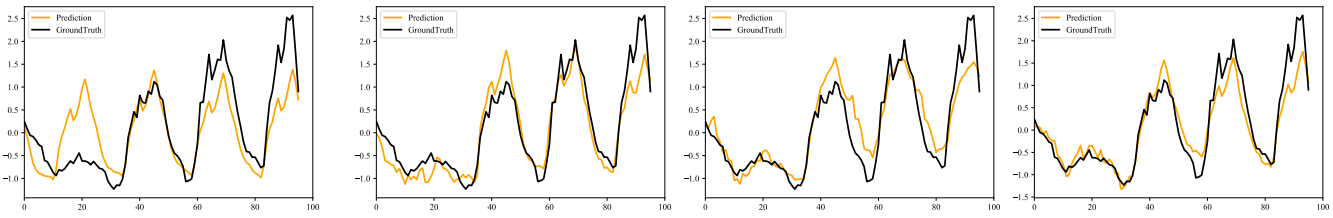
TABLE VII
RESULTS OF DIFFERENT HYPEREDGE GRAPH CONSTRUCTION METHODS.

Methods		Predicition Length			
		96	192	336	720
MSHyper-w/o AR	MSE	0.391	0.441	0.499	0.503
	MAE	0.399	0.429	0.486	0.493
MSHyper-w/o SR	MSE	0.385	**0.435**	0.487	0.496
	MAE	0.393	**0.422**	0.453	0.487
MSHyper-w/o H-H	MSE	0.433	0.479	0.519	0.524
	MAE	0.434	0.459	0.477	0.498
MSHyper	MSE	**0.383**	**0.435**	**0.480**	**0.482**
	MAE	**0.392**	0.423	**0.445**	**0.467**

The experimental results are shown in Table VII, from which we can discern the following tendencies: 1) MSHyper performs better than MSHyper-w/o AR and MSHyper-w/o SR, showing the effectiveness of the association relationship and sequential relationship, respectively. 2) Removing the hyperedge-hyperedge phase gets the worst forecasting results, which demonstrates the superiority of the hyperedge graph in enhancing hypergraph modeling.

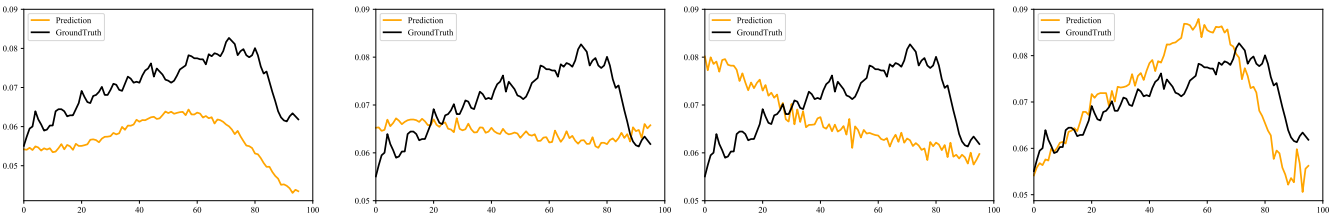
E. Case Studies

To evaluate the prediction performance of different models, we plot the forecasting results on *Weather* and *Electricity* datasets for qualitative comparison. The visualization of forecasting results are shown in Figure 4 and 5, from which we can observe that the existing models have poorer fitting ability on *Weather* dataset than on *Electricity* dataset. To explore the reason, Figure 6 shows the auto-correlation graphs of sampled variables on *Electricity* and *Weather* datasets. On *Electricity* dataset, clear daily and weekly patterns can be observed, while on *Weather* dataset, it is difficult to identify obvious daily or weekly patterns. In addition, MSHyper is still able to achieve good predictive performance on *Weather* dataset, which may be due to its ability to capture more diverse long-term and short-term patterns by modeling high-order temporal pattern interactions.



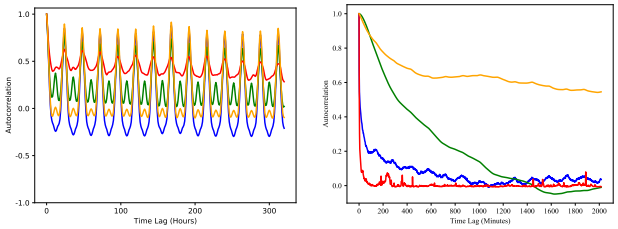
(a) DLinear (b) TimesNet (c) MSGNet (d) MSHyper

Fig. 4. Forecasting results of different models on *Electricity* dataset under the input-96-predict-96 setting. The black line represents the ground truth and the orange line represents the predicted results.



(a) DLinear (b) TimesNet (c) MSGNet (d) MSHyper

Fig. 5. Forecasting results of different models on *Weather* dataset under the input-96-predict-96 setting. The black line represents the ground truth and the orange line represents the predicted results.



(a) Electricity dataset (b) Weather dataset

Fig. 6. The auto-correlation graphs on *Electricity* and *Weather* datasets.

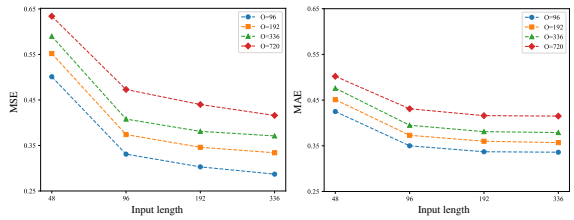


Fig. 7. Results of MSHyper with different input length on *ETTm1* dataset.

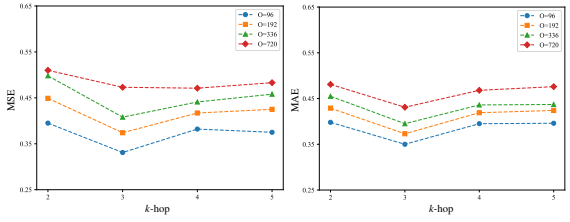


Fig. 8. Results of MSHyper with different k-hop on *ETTm1* dataset.

TABLE VIII
COMPUTATION COSTS OF DIFFERENT METHODS.

Methods	Training Time/epoch	GPU Occupation	MSE	MAE
MSHyper	13.51s	2186	**0.152**	**0.252**
MSGNet	1519.10s	11550	0.165	0.274
TimesNet	587.35s	3726	0.168	0.272
Crossformer	88.36s	9468	0.219	0.314
DLinear	**4.35s**	**756**	0.197	0.282

F. Parameter Studies

Input Length. To investigate the impact of input length, we set input length $T \in \{48, 96, 192, 336\}$ and record the results of MSHyper for multivariate long-range time series forecasting. The experimental results are shown in Figure 7, from which we can observe that with the increase of input length, the performance of MSHyper shows an upward trend. The reason is that MSHyper can capture more historical pattern information with the increase of input length.

***k*-hop.** We also perform parameter studies to measure the impact of the k-hop on *ETTm1* dataset. Figure 8 shows the results of MSHyper on *ETTm1* dataset by varying the k-hop from 2 to 5, form which we can observe that the best performance can be obtained when the k-hop is 3. The reason is that a small k-hop limits interactions to the neighboring nodes, and a large k-hop would introduce noises.

G. Computation Cost

To evaluate the computation cost of MSHyper, we compare the training time, GPU occupation, and forecasting performance of MSHyper, MSGNet, TimesNet, Crossformer, and DLinear on *Electricity* dataset with the forecasting horizon of 96. The experimental results are shown in Table VIII, from which we can observe that DLinear has lower GPU occupation and runs fastest in these methods. But it gets worse forecasting performance than MSHyper, MSGNet, and TimesNet. Compared with MSGNet, TimesNet, and Crossformer, MSHyper runs faster and gets the best forecasting performance. Overall, comprehensively considering the significant forecasting performance improvement and the computation cost, MSHyper demonstrates the superiority over existing methods.

VI. Conclusions and Future Work

In this paper, we propose MSHyper for long-range time series forecasting. Specifically, the H-HGC module is introduced to provide foundations for modeling high-order interactions between temporal patterns. The TMP mechanism is employed to aggregate high-order pattern information and learn the interaction strength between temporal patterns of different scales. Extensive experiments on eight real-world datasets show the superiority of MSHyper.

In future work, it is of interest to extend MSHyper in the following two aspects: First, since the prior knowledge-based hypergraph structures may limit the scalability of MSHyper, we will design an adaptive hypergraph module to learn the best hypergraph structures. Second, to tackle complex scenariors with diverse temporal pattern interactions, we will design a neural architecture search framework to search the best combination of hypergraph structures, capturing high-order correlations among temporal patterns of different scales automatically.

References

[1] Aniruddha Adiga, Lijing Wang, Benjamin Hurt, Akhil Peddireddy, Przemyslaw Porebski, Srinivasan Venkatramanan, Bryan Leroy Lewis, and Madhav Marathe. All models are useful: Bayesian ensembling for robust high resolution COVID-19 forecasting. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2505–2513, 2021.

[2] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*, pages 268–284, 2020.

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

[4] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110(1):1–8, 2021.

[5] George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society Series C*, 17(2):91–109, 1968.

[6] 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.

[7] 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.

[8] Weiqi Chen, Wenwei Wang, Bingqing Peng, Qingsong Wen, Tian Zhou, and Liang Sun. Learning to rotate: Quaternion transformer for complicated periodical time series forecasting. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 146–156, 2022.

[9] Yiru Chen and Silu Huang. TSExplain: Surfacing evolving explanations for time series. In *Proceedings of the ACM Conference on Management of Data*, pages 2686–2690, 2021.

[10] Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. In *Proceedings of the International Conference on Learning Representations*, 2019.

[11] Zipeng Chen, Qianli Ma, and Zhenxi Lin. Time-aware multi-scale RNNs for time series modeling. In *Proceedings of the International Joint Conference on Artifical Intelligence*, pages 2285–2291, 2021.

[12] Zonglei Chen, Minbo Ma, Tianrui Li, Hongjun Wang, and Chongshou Li. Long sequence time-series forecasting with deep learning: A survey. *Information Fusion*, 97(1):1–36, 2023.

[13] Yunyao Cheng, Peng Chen, Chenjuan Guo, Kai Zhao, Qingsong Wen, Bin Yang, and Christian S Jensen. Weakly guided adaptation for robust time series forecasting. *Proceedings of the VLDB Endowment*, 17(4):766–779, 2023.

[14] Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting. In *Proceedings of the International Joint Conference on Artifical Intelligence*, pages 1–10, 2022.

[15] 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.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*, 2021.

[17] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3558–3565, 2019.

[18] Like Gao and X Sean Wang. Continually evaluating similarity-based pattern queries on a streaming time series. In *Proceedings of the ACM Conference on Management of Data*, pages 370–381, 2002.

[19] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. In *Proceedings of the International Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1315–1325, 2019.

[20] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5415–5428, 2021.

[21] Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2024.

[22] Peter R Hansen and Asger Lunde. Estimating the persistence and the autocorrelation function of a time series that is measured with error. *Econometric Theory*, 30(1):60–93, 2014.

[23] Jingyi Hou, Zhen Dong, Jiayu Zhou, and Zhijie Liu. Discovering predictable latent factors for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2023.

[24] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proceedings of the International Conference on Learning Representations*, 2020.

[25] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 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*, pages 95–104, 2018.

[26] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, pages 1–11, 2019.

[27] Yan Li, Xinjiang Lu, Haoyi Xiong, Jian Tang, Jiantao Su, Bo Jin, and Dejing Dou. Towards long-term time-series forecasting: Feature, pattern, and distribution. In *Proceedings of the International Conference on Data Engineering*, pages 1611–1624, 2023.

[28] 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 *Proceedings of the International Conference on Learning Representations*, 2021.

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

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

[31] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, Tyler Derr, and Rajiv Ratn Shah. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 497–504, 2021.

[32] Amin Shabani, Amir Abdi, Lili Meng, and Tristan Sylvain. Scaleformer: Iterative multi-scale refining transformers for time series forecasting. In *Proceedings of the International Conference on Learning Representations*, 2022.

[33] Sandeep Subramanian, Ronan Collobert, Marc'Aurelio Ranzato, and Y-Lan Boureau. Multi-scale transformer language models. *arXiv preprint arXiv:2005.00581*, 2020.

[34] Xiangguo Sun, Hong Cheng, Bo Liu, Jia Li, Hongyang Chen, Guandong Xu, and Hongzhi Yin. Self-supervised hypergraph representation learning for sociological analysis. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11860–11871, 2023.

[35] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[36] Shiyu Wang. Neuralreconciler for hierarchical time series forecasting. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 731–739, 2024.

[37] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

[38] Qingsong Wen, Kai He, Liang Sun, Yingying Zhang, Min Ke, and Huan Xu. RobustPeriod: Robust time-frequency mining for multiple periodicity detection. In *Proceedings of the International Conference on Management of Data*, pages 2328–2337, 2021.

[39] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[40] Binqing Wu, Weiqi Chen, Wengwei Wang, Bingqing Peng, Liang Sun, and Ling Chen. WeatherGNN: Exploiting meteo- and spatial-dependencies for local numerical weather prediction bias-correction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2433–2441, 2024.

[41] 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*, pages 22419–22430, 2021.

[42] 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 ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 753–763, 2020.

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

[44] Chenxin Xu, Maosen Li, Zhenyang Ni, Ya Zhang, and Siheng Chen. GroupNet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6507, 2022.

[45] Jiawei Xue, Takahiro Yabe, Kota Tsubouchi, Jianzhu Ma, and Satish Ukkusuri. Multiwave COVID-19 prediction from social awareness using web search and mobility data. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4279–4289, 2022.

[46] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A new method for training graph convolutional networks on hypergraphs. *Advances in Neural Information Processing Systems*, pages 1–12, 2019.

[47] Xiaodong Yan, Tengwei Song, Yifeng Jiao, Jianshan He, Jiaotuan Wang, Ruopeng Li, and Wei Chu. Spatio-temporal hypergraph learning for next POI recommendation. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 403–412, 2023.

[48] Yichao Yan, Jie Qin, Jiaxin Chen, Li Liu, Fan Zhu, Ying Tai, and Ling Shao. Learning multi-granular hypergraphs for video-based person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2899–2908, 2020.

[49] Yuhao Yang, Chao Huang, Lianghao Xia, Yuxuan Liang, Yanwei Yu, and Chenliang Li. Multi-behavior hypergraph-enhanced transformer for sequential recommendation. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2263–2274, 2022.

[50] Zhangjing Yang, Weiwu Yan, Xiaolin Huang, and Lin Mei. Adaptive temporal-frequency network for time-series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(4):1576–1587, 2020.

[51] Zhangjing Yang, Weiwu Yan, Xiaolin Huang, and Lin Mei. Adaptive temporal-frequency network for time-series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(4):1576–1587, 2020.

[52] Kun Yi, Qi Zhang, Hui He, Kaize Shi, Liang Hu, Ning An, and Zhendong Niu. Deep coupling network for multivariate time series forecasting. *ACM Transactions on Information Systems*, 42(5):1–28, 2024.

[53] Nan Yin, Fuli Feng, Zhigang Luo, Xiang Zhang, Wenjie Wang, Xiao Luo, Chong Chen, and Xian-Sheng Hua. Dynamic hypergraph convolutional network. In *Proceedings of the International Conference on Data Engineering*, pages 1621–1634, 2022.

[54] 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, 2023.

[55] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3008, 2021.

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

[57] Zhiqiang Zhang, Dandan Zhang, and Yun Wang. Fast long sequence time-series forecasting for edge service running state based on data drift and non-stationarity. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2024.

[58] Kai Zhao, Chenjuan Guo, Yunyao Cheng, Peng Han, Miao Zhang, and Bin Yang. Multiple time series forecasting with dynamic graph modeling. *Proceedings of the VLDB Endowment*, 17(4):753–765, 2023.

[59] Wendong Zheng, Putian Zhao, Gang Chen, Huihui Zhou, and Yonghong Tian. A hybrid spiking neurons embedded LSTM network for multivariate time series learning under concept-drift environment. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):6561–6574, 2022.

[60] 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.

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

[62] Zhengyang Zhou, Jiahao Shi, Hongbo Zhang, Qiongyu Chen, Xu Wang, Hongyang Chen, and Yang Wang. CreST: A credible spatiotemporal learning framework for uncertainty-aware traffic forecasting. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 985–993, 2024.