HyperMixer: Specializable Hypergraph Channel Mixing for Long-term Multivariate Time Series Forecasting

Changyuan Tian[1,2,3,4,*], **Zhicong Lu**[1,2,3,4,*], **Zequn Zhang**[1,†]
Heming Yang[1,2,3,4], **Wei Cao**[1], **Zhi Guo**[1,2], **Xian Sun**[1,2,3,4], **Li Jin**[1,2,†]
[1]Aerospace Information Research Institute, Chinese Academy of Sciences
[2]Key Laboratory of Target Cognition and Application Technology (TCAT)
[3]University of Chinese Academy of Sciences
[4]School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences
tianchangyuan21@mails.ucas.edu.cn, zqzhang1@mail.ie.ac.cn, jinlimails@gmail.com

Abstract

Long-term Multivariate Time Series (LMTS) forecasting aims to predict extended future trends based on channel-interrelated historical data. Considering the elusive channel correlations, most existing methods compromise by treating channels as independent or tentatively modeling pairwise channel interactions, making it challenging to handle the characteristics of both higher-order interactions and time variation in channel correlations. In this paper, we propose HyperMixer, a novel specializable hypergraph channel mixing plugin which introduces versatile hypergraph structures to capture group channel interactions and time-varying patterns for long-term multivariate time series forecasting. Specifically, to encode the higher-order channel interactions, we structure multiple channels into a hypergraph, achieving a two-phase message-passing mechanism: channel-to-group and group-to-channel. Moreover, the functionally specializable hypergraph structures are presented to boost the capability of hypergraph to capture the time-varying patterns across periods, further refining modeling of channel correlations. Extensive experimental results on seven available benchmark datasets demonstrate the effectiveness and generalization of our plugin in LMTS forecasting. The visual analysis further illustrates that HyperMixer with specializable hypergraphs tailors channel interactions specific to certain periods.

Introduction

Long-term Multivariate Time Series (LMTS) forecasting aims to predict extended future trends of multiple interrelated variables (*a.k.a*, channels). It has long been studied and applied across diverse fields like traffic, weather, and energy. The channels within multivariate time series potentially denotes to monitoring variables in real systems (e.g., traffic sensors, power system indicators). Therefore, it is significant for LMTS to model intricate correlations entailed in these channels, followed by making effective predictions.

Considering the elusive channel correlations, most existing methods compromise through two main streams: (1) channel-independent (Zeng et al. 2023; Nie et al. 2023; Wang et al. 2024): treating channels as independent and processing each channel separately, (2) channel-mixing (Zhou et al. 2021; Wu et al. 2021; Zhang and Yan 2023): tentatively modeling pairwise channel interactions by well-designed architectures, such as cross-channel attention and graph neural networks. Despite the temporary progress, they fail to completely unleash the power of channel correlations, making it challenging to handle the characteristics of both higher-order interactions and time variation in channel correlations.

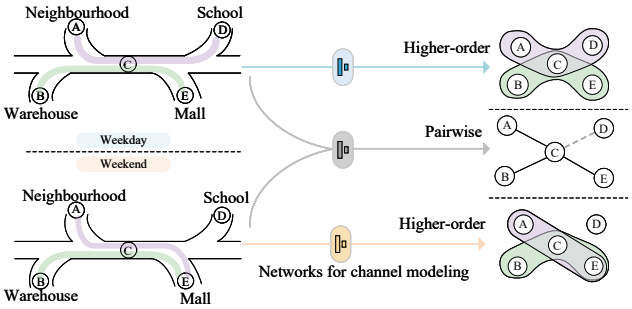


Figure 1: Comparison of previous channel modeling methods (gray pathways) versus our method (colored pathways) in traffic scenarios. Previous methods focus on pairwise channel interactions and expect to model the correlations across different time periods using a general network. Our method targets higher-order channel interactions and models time-varying patterns via multiple specialized networks.

The higher-order interaction refers to complex correlations among more than two channels within LMTS, reflecting obvious group behaviors. Taking the weekday traffic patterns illustrated in Figure 1 as an example, the traffic flow of road D is affected by the flow of road A flowing into road C, collectively forming a traffic route with group interaction behavior: roads A, D, and part of C involved in this group display consistent traffic flow trends. This informative higher-order interaction undoubtedly facilitates predicting the traffic flow of roads A, C, and D. However, it is either ignored by channel-independent methods or reduced to multi-hop pairwise interactions (i.e., roads A to C and roads C to D) by existing channel-mixing methods, which is not

*These authors contributed equally.
†Corresponding author.

conducive to LMTS. In addition, **the time-varying characteristic refers to the channel correlations exhibiting various association patterns over different time periods**. As illustrated in Figure 1, the traffic route (A, C, D) on weekday shifts to the router (A, C, E) on weekends. This demands the model to capture versatile time-varying patterns across periods. Nevertheless, it is not afforded by current inflexible methods which are forced to balance the modeling of pairwise interactions: roads C to D and roads C to E, leading to a generalized compromise across all time periods. This is illustrated by the gray dashed line in Figure 1, indicating the ambiguous correlation between C and D. With the above consideration, it is crucial to capture the higher-order interactions and time-varying patterns for modeling channel correlations.

In this paper, we propose HyperMixer, a novel specializable hypergraph channel mixing plugin which introduces versatile hypergraph structures to capture group channel interactions and time-varying patterns for long-term multivariate time series forecasting. HyperMixer comprises two core components: a **H**ypergraph **C**hanel **M**ixer (HCM) and a **S**pecializable **H**ypergraph **S**tructure **L**earning module (HS^2L). Specifically, to encode the higher-order interactions, we structure multiple channels into a channel hypergraph, where each channel serves as a node and hyperedges connect multiple nodes simultaneously, forming channel groups. HCM is introduced to encode the channel hypergraph by conducting two-phase message-passing: channel-to-group and group-to-channel. To capture the time-varying patterns, inspired by the specialization concept in the mixture of experts (Jacobs et al. 1991), we devise HS^2L to develop specialized hypergraph structures tailored to specific periods, utilizing a router layer and multiple experts. Ultimately, inserting the plugin HyperMixer into existing representative methods incorporates channel correlations featured by higher-order interactions and time-varying characteristics, improving the predictive performance for long-term multivariate time series. The main contributions of this paper are summarized as follows:

- We propose HyperMixer, a novel specializable hypergraph channel mixing plugin which introduces versatile hypergraph structures to capture group channel interactions and time-varying patterns for long-term multivariate time series forecasting.
- To encode the higher-order channel interactions, we structure multiple channels into a hypergraph, achieving a two-phase message-passing mechanism: channel-to-group and group-to-channel. To capture the time-varying patterns across periods, the functionally specializable hypergraph structures are presented to boost the capability of hypergraph.
- Extensive experimental results on seven available benchmark datasets demonstrate the effectiveness and generalization of our plugin in LMTS forecasting. The visual analysis further illustrates that HyperMixer with specializable hypergraphs tailors channel interactions specific to certain periods.

Related Work

Long-Term Multivariate Time Series Forecasting

Deep learning-based approaches dominate recent research, introducing a variety of designs suitable for Long-term multivariate time series (LMTS) forecasting (Zheng et al. 2020; Zhou et al. 2022; Nie et al. 2023). Recurrent Neural Networks (RNNs) (Che et al. 2018; Salinas et al. 2020) and Temporal Convolutional Networks (TCNs) (Lea et al. 2017) are utilized for modeling time series data to capture temporal dependencies. However, they encounter challenges when modeling long-term dependencies. Graph Neural Networks (GNNs) (Kipf and Welling 2017; Veličković et al. 2018) have also gained prominence due to their ability to explicitly capture spatiotemporal correlation in LMTS (Diao et al. 2019; Cao et al. 2020; Cai et al. 2024). However, these methods suffer from over-smoothing and overfitting, and are also limited to pairwise interactions (Lambiotte, Rosvall, and Scholtes 2019; Battiston et al. 2020). Recently, linear-based LMTS forecasting methods (Zeng et al. 2023; Wang et al. 2024; Huang et al. 2024) have achieved surprisingly strong performance with streamlined architectures. For example, LTSF-Linear (Zeng et al. 2023) is proposed to surpass previous complex methods through simple one-layer linear network. On the other hand, a series of transformer-based time series methods have been proposed (Wu et al. 2021; Zhou et al. 2021; Nie et al. 2023; Liu et al. 2024), notably advancing the development of LMTS forecasting. Representative PatchTST (Nie et al. 2023) employs subseries-level embedding techniques and the channel independence assumption to significantly improve the accuracy of LMTS forecasting. Despite significant progress, most current methods prioritize capturing temporal dependencies, underestimating the channel modeling in LMTS.

Channel Modeling in LMTS Forecasting

Current methods can be divided into channel-independent methods and channel-mixed methods. Channel-independent methods assume that channels are independent and process each channel separately, thereby simplifying the task. However, they fail to exploit the potential benefits of channel correlations. In contrast, channel-mixed methods expect to capture the channel correlations through well-designed architectures. For example, methods based on GNN (Cao et al. 2020; Deng and Hooi 2021; Cai et al. 2024) organize channels into a graph, where edges facilitate message passing between pairs of channels. Patch-based methods (Zhang and Yan 2023; Huang et al. 2024) decompose each time series of the LMTS into multiple patches, enabling the exploration of dependencies between patches across different channels using cross-channel attention or multi-layer perceptrons (MLPs). Additionally, several techniques have been developed to explicitly represent specific channel relationships. For instance, channel clustering (Chen et al. 2024) is proposed to captures channel similarity by minimizing intra-cluster series distance, while LIFT (Zhao and Shen 2024) identifies lead-lag relationships between channels by analyzing cross-correlation coefficients, which indicate the degree of lead and lag between two channels. Despite ad-

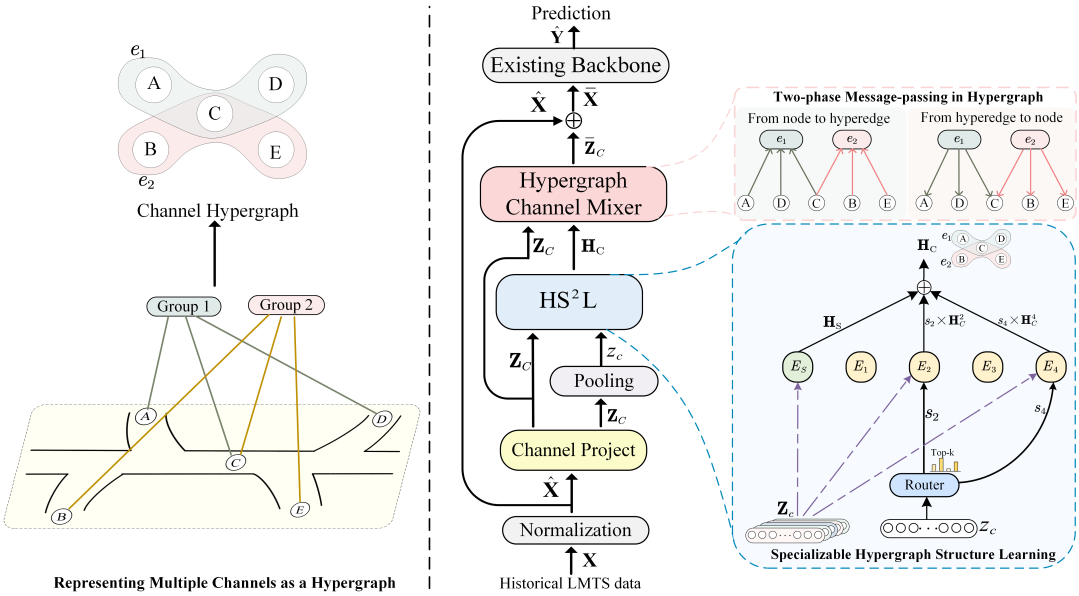


Figure 2: The left panel illustrates the process of representing multiple channels as a hypergraph. Five channels, involving two types of group interactions, are naturally represented as a hypergraph with five nodes and two hyperedges. The right panel presents the pipeline of HyperMixer.

vancements in capturing channel correlations through various designs, these methods primarily focus on pairwise interactions and attempt to capture interaction patterns across different periods using a shared, inflexible network. This approach presents difficulties in addressing the characteristics of both higher-order interactions and the time-varying patterns inherent in channel correlations.

Proposed Method

The problem of LMTS forecasting is formulated as: Let $\mathbf{X} \in \mathbb{R}^{L \times N}$ represent the historical data, where L denotes the number of historical time steps and N denotes the number of channels. The objective of LMTS forecasting is to predict a matrix $\hat{Y} \in \mathbb{R}^{T \times N}$, where T represents the prediction length, i.e., the number of future time steps to be forecasted.

Our HyperMixer is illustrated in Figure 2, which involves channel hypergraph construction, specializable hypergraph structure learning, and a hypergraph channel mixing.

Representing Multiple Channels as a Hypergraph

Compared to ordinary graphs, hypergraphs offer the advantage of hyperedges that can connect more than two nodes simultaneously, enabling them to naturally represent complex higher-order interactions (Feng et al. 2019; Jiang et al. 2019). A hypergraph is typically represented as $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ represents the set of hyperedges, with each hyperedge $e \in \mathcal{E}$ is a non-empty subset of nodes. The hypergraph structure can be futher formalized using an incidence matrix $\mathbf{H} \in \mathbb{R}_{\geq 0}^{|\mathcal{V}| \times |\mathcal{E}|}$, where each element $\mathbf{H}(v, e)$ indicates the degree to which node v is associated with hyperedge e.

To model potential higher-order interactions among multiple channels, we organize them into a channel hypergraph, $\mathcal{H}_C = (\mathcal{C}, \mathcal{E}_C)$. Here, $\mathcal{C}$ denotes the set of channels in the LMTS, and $\mathcal{E}_C$ represents the hyperedges, with a hyperparameter m as the total number of hyperedges. Figure 2 further illustrates this construction process.

Specializable Hypergraph Structure Learning

After constructing the channel hypergraph $\mathcal{H}_C$, Specializable Hypergraph Structure Learning (HS^{2}L) develops a hypergraph structure $\mathbf{H}_C$ for $\mathcal{H}_C$, tailored to specific periods by using a router layer and multiple hypergraph structure learning experts.

The processing workflow of HS^{2}L is as follows.

Channel Projection. First, we obtain the initial representations of channels in the feature space through a channel projection layer which maps the original time series $\mathbf{X}$ to a set of vectors. Formally,

$$\mathbf{Z}_c = \text{ChannelProj}(\mathbf{X}), \tag{1}$$

here, $\mathbf{Z}_C \in \mathbb{R}^{N \times d_c}$ represents the channel representation, where N is the number of channels, and d_c denotes the dimensionality of the channel representation. ChannelProj$(\cdot)$ is implemented using a one-layer linear transformation.

Router. The router is designed to identify features of input time series data and direct it to the appropriate expert.

For the channel representations $\mathbf{Z}_C \in \mathbb{R}^{N \times d_c}$, we first derive a compact representation vector $\mathbf{z}_c \in \mathbb{R}^{d_c}$ using an average pooling operation:

$$\mathbf{z}_c = \text{AveragePool}(\mathbf{Z}_C). \tag{2}$$

This representation vector is then fed into the router, which includes a simple linear layer. The output is normalized with the Softmax function to compute scores for each expert:

$$\mathbf{s} = \text{Softmax}(\text{Router}(\mathbf{z}_c)), \tag{3}$$

where $\mathbf{s} \in \mathbb{R}^K$ represents the score distribution across the K experts.

Finally, a top-k activation mechanism is employed to select the top k experts with the highest scores for the input data.

Hypergraph Structure Learning Expert. Upon assignment by the router, each sample is directed to a specific expert E, which is responsible for learning a hypergraph structure $\mathbf{H}_C^E$ based on the sample's channel representation $\mathbf{Z}_c$. This process is formally defined as follows:

$$\mathbf{H}_C^E = \text{Expert}(\mathbf{Z}_C), \tag{4}$$

where $\mathbf{H}_C^E \in \mathbb{R}^{N\times m}$ denotes the hypergraph structure learned by expert E. Here, m is a hyperparameter that specifies the number of hyperedges in the channel hypergraph. Each hypergraph structure learning expert E is implemented as a single-layer linear transformation followed by a ReLU activation function.

In more detail, HyperMixer includes one expert shared across all LMTS data and K additional experts that can be selectively activated.

Learned Hypergraph Structures. The hypergraph structure output by the HS^{2}L consists of two parts. The first part is the hypergraph structure learned by a shared hypergraph structure learning expert, denoted as $\mathbf{H}_S$. The second part is the combination of the outputs from the top-k experts.

Let $\{E_{i_1}, E_{i_2}, \ldots, E_{i_k}\}$ be the k experts selected for the LMTS data sample i in a batch by the top-k selection mechanism. Each expert E_{i_j} outputs a hypergraph structure $\mathbf{H}_C^{i_j}$, where $j = 1, 2, \ldots, k$. The final hypergraph structure $\mathbf{H}_C$ is represented as the weighted sum of the shared expert's output and these experts' outputs:

$$\mathbf{H}_C = \mathbf{H}_S + \sum_{j=1}^{k} s_{i_j}\mathbf{H}_C^{i_j}, \tag{5}$$

where s_{i_j} is the score corresponding to expert E_{i_j}, which is the i_jth element in the expert score vector $\mathbf{s}$.

Hypergraph Chanel Mixer

The Hypergraph Channel Mixer (HCM), implemented by a hypergraph neural network (Feng et al. 2019; Jiang et al. 2019), aims to encode the channel hypergraph $\mathcal{H}_C$ and its structure $\mathbf{H}_C$, effectively mapping the nodes of the hypergraph into a d-dimensional vector space. This encoding captures complex channel correlations by leveraging higher-order interactions inherent in the hypergraph structure.

To achieve this, HCM employs a two-phase message-passing process as illustrated in Figure 2. In the first phase, messages are transmitted from nodes to hyperedges (i.e., groups), allowing for the aggregation of collective node information. In the second phase, messages are sent from hyperedges back to nodes, thereby updating and refining the node representations.

From Node to Hyperedge. First, HCM performs message passing from nodes to hyperedges. To obtain the messages passed from nodes to hyperedges, we employ a single-layer nonlinear transformation to convert the original node representations. Specifically,

$$\mathbf{Z}_{\mathcal{C}\to\mathcal{E}_C} = \text{ReLU}(\mathbf{W}_1\mathbf{Z}_C), \tag{6}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_c\times d_c}$ denotes the learnable parameters, and $\mathbf{Z}_{\mathcal{C}\to\mathcal{E}_C} \in \mathbb{R}^{N\times d_c}$ represents the messages passed from nodes to hyperedges.

Next, we aggregate the messages from the nodes to obtain the hyperedge representations. This process is expressed as:

$$\mathbf{Z}_{\mathcal{E}_C} = \mathbf{D}_{\mathcal{E}_C}^{-1}\mathbf{H}_C^T\mathbf{Z}_{\mathcal{C}\to\mathcal{E}_C}, \tag{7}$$

where $\mathbf{D}_{\mathcal{E}_C}^{-1} \in \mathbb{R}^{m\times m}$ represents the degree matrix of the hyperedges, which is a diagonal matrix with $\mathbf{D}_{\mathcal{E}_C}^{-1}(e,e) = (\sum_{c\in\mathcal{C}}\mathbf{H}_C(c,e))^{-1}$; $\mathbf{H}_C^T \in \mathbb{R}^{m\times N}$ denotes the transpose of the hypergraph structure, and $\mathbf{Z}_{\mathcal{E}_C} \in \mathbb{R}^{m\times d_c}$ represents the hyperedge representations.

From Hyperedge to Node. After obtaining hyperedge representations, HCM performs message passing from hyperedges to nodes. Specifically, to obtain the message from hyperedge e to node c, HCM first performs a concatenation operation and then uses a single-layer nonlinear transformation to convert the concatenated vector:

$$\mathbf{z}_{e\to c} = \text{ReLU}([\mathbf{Z}_C(c), \mathbf{Z}_{\mathcal{E}_C}(e)]\mathbf{W}_2), \tag{8}$$

where $[\cdot,\cdot]$ denotes the concatenation operation of vectors, $\mathbf{W}_2 \in \mathbb{R}^{2d_c\times d_c}$ represents the learnable weight parameters, and $\mathbf{z}_{e\to c} \in \mathbb{R}^{d_c}$ represents the message from hyperedge e to node c. $\mathbf{Z}_C(c)$ denotes the representation vector of node c and $\mathbf{Z}_{\mathcal{E}_C}(e)$ denotes the representation vector of hyperedge e.

Next, we need to aggregate messages from all hyperedges to update the representation of node c. This can be represented as:

$$\mathbf{Z}'_C(c) = \sum_{e\in\mathcal{E}_C}\mathbf{H}_C(c,e)\mathbf{z}_{e\to c}, \tag{9}$$

where $\mathbf{Z}'_C(c)$ represents the new representation of node c.

To maintain the stability of node representations, the aggregated node representation is normalized as follows:

$$\mathbf{Z}''_C = \mathbf{D}_C^{-1}\mathbf{Z}'_C, \tag{10}$$

here, $\mathbf{D}_C$ represents the degree matrix of nodes, which is a diagonal matrix with diagonal elements representing the degree of each node, defined as $\mathbf{D}_C(i,i) = \sum_{e\in\mathcal{E}}\mathbf{H}(i,e)$.

Update Channel Representation. The updated channel representation is obtained as follows:

$$\bar{\mathbf{Z}}_c = \mathbf{Z}_c + \mathbf{Z}''_c. \tag{11}$$

Dataset	# Channels	# Timesteps	Frequency
ETTh1	7	17,420	1 hour
ETTh2	7	17,420	1 hour
ETTm1	7	69,680	15 min
ETTm2	7	69,680	15 min
Weather	21	52,696	10 min
Electricity	321	26,304	1 hour
Traffic	862	17,544	1 hour

Table 1: The statistics of datasets.

This enhanced channel representation is then utilized to refine the original multivariate time series representation, which is specifically expressed as:

$$\mathbf{X}' = \hat{\mathbf{X}} + \bar{\mathbf{Z}}_c \mathbf{W}_o, \tag{12}$$

where $\mathbf{W}_o \in \mathbb{R}^{d_c \times L}$ are learnable parameters.

The representation of the multivariate time series data improved by HyperMixer can be fed into various backbones, such as TimeMixer (Wang et al. 2024), PatchTST (Nie et al. 2023), and DLiner (Zeng et al. 2023), to enhance predictive performance.

Experiment

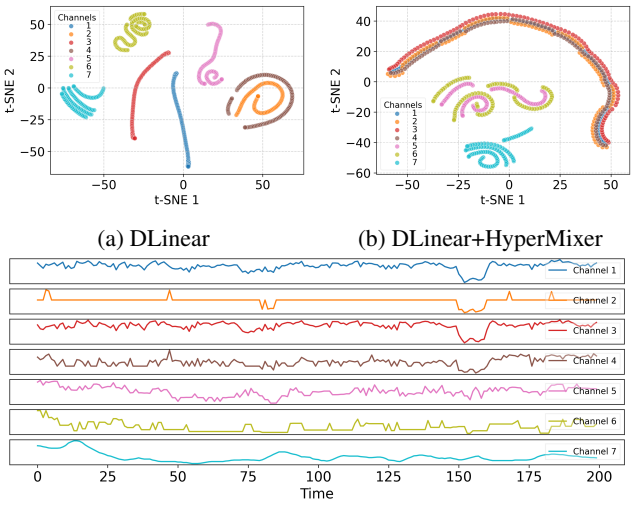
Datasets

We select seven mainstream long-term multivariate time series datasets, including ETT (ETTh1, ETTh2, ETTm1, ETTm2) (Zhou et al. 2021), Traffic (Lai et al. 2018), Weather (Zeng et al. 2023), and Electricity (Wu et al. 2020). Table 1 presents the statistics of the seven datasets.

Experimental Details

The proposed HyperMixer serves as a channel mixing plugin that can be incorporated into a variety of existing LMTS forecasting backbones. To validate the enhancement brought by HyperMixer, we select four current state-of-the-art LMTS forecasting models as our backbones. These include the linear-based methods TimeMixer(Wang et al. 2024) and DLinear(Zeng et al. 2023), the transformer-based method PatchTST(Nie et al. 2023), and the convolution-based method TimesNet(Wu et al. 2023). Additionally, we compare our approach with another competitive channel modeling method, CCM(Chen et al. 2024), which is based on a channel clustering strategy. For a fair comparison, the optimal hyperparameter settings provided in the official implementations of the backbones are used for both the backbones and the HyperMixer-enhanced models. For CCM, we use the experimental results provided in the original paper for PatchTST, DLinear, and TimesNet, while the results for the recently proposed TimeMixer are based on our own reproduction.

For our proposed HyperMixer, the number of experts T is explored within the range of [4, 12], with the number of activated experts, k, set to 2. The number of hyper-edges, m, is searched within the range of [4, 10]. Mean Squared Error (MSE) and Mean Absolute Error (MAE) are used as metrics



(a) DLinear (b) DLinear+HyperMixer

(c) Original multivariate time series data

Figure 3: t-SNE visualization of channel representations on the ETTh2 Dataset.

for evaluating predictive performance. For all datasets in our experiments, the prediction lengths are set to {96, 192, 336, 720}. We use the L2 loss function for training. All experiments are implemented using PyTorch and executed on an NVIDIA A800-80GB GPU.

Main Results

Table 2 presents the experimental results of HyperMixer across four representative backbones and seven datasets. The results clearly demonstrate that HyperMixer significantly enhances the performance of the backbones, achieving an average performance boost of 3.46%. Notably, models equipped with HyperMixer achieve the best performance in most cases, outperforming both the base models and those equipped with CCM. In details, ETTh1 is more challenging than ETTm1 due to its larger sampling intervals and resulting more pronounced fluctuations. HyperMixer's substantial improvements on ETTh1 highlight the importance of channel correlations in predicting unstable trends, with similar patterns seen in ETTm2 and ETTh2. Furthermore, in datasets like Weather, Electricity, and Traffic—characterized by a higher number of channels—HyperMixer achieves even greater average improvements compared to other ETT datasets with fewer channels. This is attributed to its powerful capability in modeling complex channel correlations.

Qualitative Visualization

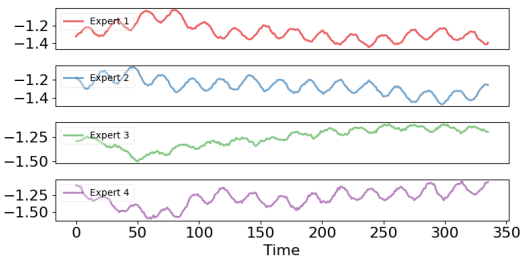
Channel Representation Visualization. To understand why HyperMixer works, we further conduct a visualization analysis of HyperMixer on the ETTh2 dataset, which includes 7 channels. Figures 3a and Figure 3b respectively present the t-SNE visualizations of channel representations learned by the base model DLinear and the DLinear model equipped with HyperMixer. A noticeable transformation occurs with HyperMixer: previously dispersed channel repre-

Model	Metric	ETTh1	ETTh2	ETTm1	ETTm2	Weather	Electricity	Traffic
PatchTST	MSE	0.417	0.332	0.354	0.270	0.227	0.167	0.396
	MAE	0.431	0.383	0.385	0.329	0.264	0.260	0.267
+CCM	MSE	0.412	0.330	0.353	0.262	0.225	0.167	0.389
	MAE	**0.430**	**0.372**	0.381	0.322	0.263	0.261	**0.259**
+HyperMixer	MSE	**0.408**	**0.327**	**0.346**	**0.256**	**0.223**	**0.162**	**0.381**
	MAE	**0.430**	0.379	**0.380**	**0.315**	**0.260**	**0.255**	0.262
TimeMixer	MSE	0.463	0.387	**0.385**	**0.276**	0.245	0.182	0.485
	MAE	0.446	0.409	0.400	**0.323**	0.275	0.273	0.298
+CCM	MSE	0.455	0.383	0.388	0.277	0.246	0.189	0.486
	MAE	0.443	0.408	0.400	**0.323**	**0.274**	0.277	0.304
+HyperMixer	MSE	**0.450**	**0.376**	0.386	**0.276**	**0.243**	**0.171**	**0.464**
	MAE	**0.440**	**0.403**	**0.399**	**0.323**	**0.274**	**0.264**	**0.294**
DLinear	MSE	0.433	0.431	0.359	0.265	0.246	0.166	0.434
	MAE	0.445	0.444	0.381	0.329	0.299	0.263	**0.295**
+CCM	MSE	0.423	0.400	**0.355**	0.289	0.255	0.173	0.435
	MAE	0.437	0.428	0.378	0.349	0.303	0.275	0.296
+HyperMixer	MSE	**0.414**	**0.349**	0.357	**0.261**	**0.228**	**0.165**	**0.421**
	MAE	**0.425**	**0.395**	**0.375**	**0.316**	**0.263**	**0.257**	0.297
TimesNet	MSE	0.458	0.414	0.400	0.291	0.259	0.193	0.620
	MAE	0.450	0.427	0.406	0.333	0.287	0.295	0.336
+CCM	MSE	0.454	0.411	0.399	0.288	0.256	0.179	0.571
	MAE	0.445	0.422	0.405	0.330	0.281	0.279	0.339
+HyperMixer	MSE	**0.447**	**0.383**	**0.393**	**0.286**	**0.249**	**0.178**	**0.540**
	MAE	**0.439**	**0.406**	**0.403**	**0.326**	**0.277**	**0.273**	**0.312**
Imp (%)		2.54%	6.15%	1.02%	2.37%	3.88%	4.02%	4.20%

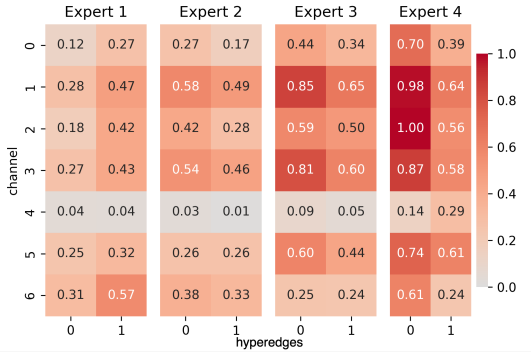
Table 2: Forecasting results, averaged over four prediction lengths: $\{96, 192, 336, 720\}$. A smaller MSE or MAE signifies a more accurate prediction. A bold result signifies the best performance, while an underlined result indicates the second-best performance. The look-up window length L aligns with each backbone's official setting: 512 for PatchTST, 336 for DLinear, and 96 for both TimeMixer and TimesNet. For HyperMixer, the number of hyperedges is set to 10 and the number of experts to 4 for the ETTh2, ETTm1 and Traffic datasets, while 8 for others. *Imp* represents the average percentage improvement achieved by HyperMixer in the MSE and MAE metrics across all backbones.

sentations become more tightly grouped and concentrated within the feature space. To further illustrate HyperMixer's capability in identifying channel group behavior, we visualize the 7 channels in Figure 3c. This figure shows the time series data over 200 time steps from the ETTh2 dataset. Channels 1, 2, 3, and 4, which exhibit similar behaviors, are closely aligned in the feature space as demonstrated in Figure 3b. Likewise, channels 5 and 6 show similar clustering patterns. In contrast, channel 7, which displays unique behavior, stands apart with an independent feature representation in Figure 3b. These visualizations demonstrate that HyperMixer with specializable hypergraphs excels in identifying and encoding intricate channel correlations, resulting in more expressive and informative channel representations.

Visualization of the HS^{2}L. To analyze the behavior of HS^{2}L, we present the distinguishable periods identified by the router in HS^{2}L and the specialized hypergraph structures that correspond to these periods. Each time series data depicted in Figure 4a is generated by averaging multiple instances assigned by the router to a specific expert, serving as a representative of all data handled by that expert. From Figure 4a , we can observe that the router can identifying distinct periods in the LMTS data and assigning them to the appropriate experts for processing. Specifically, experts 1 and 2 are responsible for handling LTMS data with downward trends, whereas experts 3 and 4 deal with upward trends. Moreover, although both experts 1 and 2 process data with downward trends, their focal points differ: expert 1 processes data that end in a trough position, whereas expert 2 processes data that end in a peak position, which imply opposite future trend changes. Experts 3 and 4 exhibit similar distinctions. Figure 4b illustrates the normalized hypergraph structures output by the four experts; larger values



(a) The distinguishable periods identified by the router in HS^{2}L



(b) Specialized hypergraphs with 2 hyperedges

Figure 4: Visualization of the HS^{2}L using the DLinear base model on the ETTh2 dataset

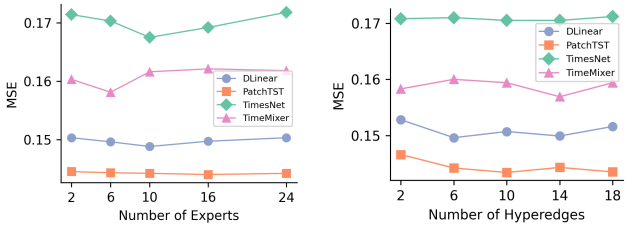
indicate stronger higher-order channel interactions. It is observed that the hypergraph structure values for experts 3 and 4, who handle more volatile data, are generally higher. This may suggest that when LMTS data changes more dramatically (e.g., rapid increases), accurate predictions rely more heavily on the information provided by channel correlation.

Ablation Study

To validate the effectiveness of HyperMixer components, we conduct ablation experiments using TimesNet as the base model. We incrementally integrate HCM and HS^{2}L to assess their contributions across ETTh1, ETTm2, and Weather datasets. Observations from Table 3 are as follows. First, starting from the base model, the variant with HCM and the variant with HCM + HS^{2}L (i.e., TimesNet+HyperMixer) show progressively better predictive performance, highlighting each component's effectiveness. Secondly, HCM notably contributes to datasets with more channels like Weather, indicating its ability to capture complex correlations. Thirdly, HS^{2}L yields greater gains on datasets like Weather (*1.64%*) and ETTm2 (*1.10%*) , which contain richer time-varying patterns due to their minute-level sampling frequency. This suggests that the HS^{2}L can effectively capture the time-varying patterns in LMTS data, thus strengthening the model's predictive capability.

Model	Metric	ETTh1	ETTm2	Weather
TimesNet	MSE	0.458	0.291	0.259
	MAE	0.450	0.333	0.287
+HCM	MSE	0.451	0.290	0.254
	MAE	0.440	0.329	0.281
	Gain (%)	*1.87%*	*0.75%*	*1.88%*
+HCM+HS^{2}L	MSE	**0.447**	**0.286**	**0.249**
	MAE	**0.439**	**0.326**	**0.277**
	Gain (%)	*0.61%*	*1.10%*	*1.64%*

Table 3: Ablation experiment based on the base model TimesNet. *Gain* represents the average improvement percentage over the previous setup for MSE and MAE metrics.



(a) Analysis of the number of experts K (b) Analysis analysis of the number of hyperedges m

Figure 5: Hyperparameter analysis on the weather dataset with a prediction length of 96 across different bachbones: PatchTST, TimeMixer, DLinear, and TimesNet

Hyperparameter Analysis

To explore the impact of two important hyperparameters: the number of experts K and the number of hyperedges m, we conduct a hyperparameter analysis. As shown in Figure 5a, with an increase in the number of experts, the performance of all models enhanced by HyperMixer initially improves and then deteriorates. This suggests that establishing an appropriate number of experts to avoid redundancy is necessary, with a range of 2 to 10 being advisable. Similarly, Figure 5b shows that choosing 6 to 14 hyperedges is a suitable starting point.

Conclusion

In this paper, we propose HyperMixer, a novel channel mixing plugin with specializable hypergraphs, to effectively manage the multiple interrelated channels in LMTS forecasting. Empowered by hypergraph channel mixing and period-oriented hypergraph structure specialization, HyperMixer excels in channel modeling by encoding high-order interactions and capturing time-varying patterns. Moreover, HyperMixer can incorporate this channel modeling capability into arbitrary forecasting models as a plugin. Extensive experiments demonstrate the superiority of HyperMixer in LMTS forecasting. Further visualization and ablation studies are presented to provide insights for our HyperMixer.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (Grant No. 62206267)

References

Battiston, F.; Cencetti, G.; Iacopini, I.; Latora, V.; Lucas, M.; Patania, A.; Young, J.-G.; and Petri, G. 2020. Networks beyond pairwise interactions: Structure and dynamics. *Physics reports*, 874: 1–92.

Cai, W.; Liang, Y.; Liu, X.; Feng, J.; and Wu, Y. 2024. Msgnet: Learning multi-scale inter-series correlations for multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 11141–11149.

Cao, D.; Wang, Y.; Duan, J.; Zhang, C.; Zhu, X.; Huang, C.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. In *Advances in neural information processing systems*, volume 33, 17766–17778.

Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1): 6085.

Chen, J.; Lenssen, J. E.; Feng, A.; Hu, W.; Fey, M.; Tassiulas, L.; Leskovec, J.; and Ying, R. 2024. From Similarity to Superiority: Channel Clustering for Time Series Forecasting. arXiv:2404.01340.

Deng, A.; and Hooi, B. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 4027–4035.

Diao, Z.; Wang, X.; Zhang, D.; Liu, Y.; Xie, K.; and He, S. 2019. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 890–897.

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; and Gao, Y. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3558–3565.

Huang, Q.; Shen, L.; Zhang, R.; Cheng, J.; Ding, S.; Zhou, Z.; and Wang, Y. 2024. Hdmixer: Hierarchical dependency with extendable patch for multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 12608–12616.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87.

Jiang, J.; Wei, Y.; Feng, Y.; Cao, J.; and Gao, Y. 2019. Dynamic Hypergraph Neural Networks. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 2635–2641. ijcai.org.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, 95–104.

Lambiotte, R.; Rosvall, M.; and Scholtes, I. 2019. From networks to optimal higher-order models of complex systems. *Nature physics*, 15(4): 313–320.

Lea, C.; Flynn, M. D.; Vidal, R.; Reiter, A.; and Hager, G. D. 2017. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 156–165.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.

Salinas, D.; Flunkert, V.; Gasthaus, J.; and Januschowski, T. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3): 1181–1191.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, S.; Wu, H.; Shi, X.; Hu, T.; Luo, H.; Ma, L.; Zhang, J. Y.; and ZHOU, J. 2024. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *The Eleventh International Conference on Learning Representations*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in neural information processing systems*, volume 34, 22419–22430.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 753–763.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9): 11121–11128.

Zhang, Y.; and Yan, J. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In *The Eleventh International Conference on Learning Representations*.

Zhao, L.; and Shen, Y. 2024. Rethinking Channel Dependence for Multivariate Time Series Forecasting: Learning from Leading Indicators. In *The Twelfth International Conference on Learning Representations*.

Zheng, C.; Fan, X.; Wang, C.; and Qi, J. 2020. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 1234–1241.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, 27268–27286. PMLR.