HSTMixer: A Hierarchical MLP-Mixer for Large-Scale Traffic Forecasting

Yongyao Wang[1] , **Jingyuan Wang**[1,2*] , **Xie Yu**[1] , **Jiahao Ji**[1] and **Chao Li**[1,3]

[1]School of Computer Science and Engineering, Beihang University, Beijing, China
[2]School of Economics and Management, Beihang University, Beijing, China
[3]Shenzhen Institute of Beihang University, Shenzhen, China

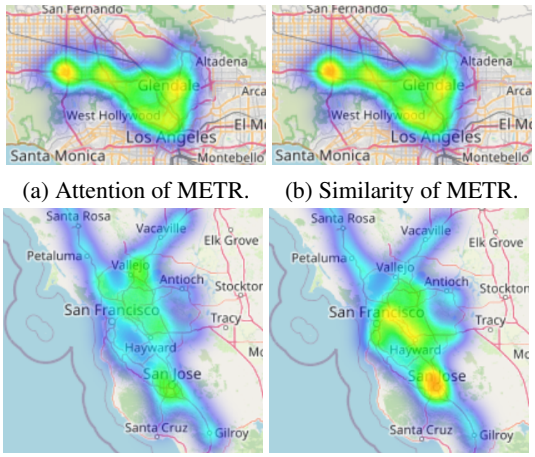
Abstract

Traffic forecasting task is significant to modern urban management. Recently, there is growing attention on large-scale forecasting, as it better reflects the complexity of real-world traffic networks. However, existing models often exhibit quadratic computational complexity, making them impractical for large-scale real-world scenarios. In this paper, we propose a novel framework, **H**ierarchical **S**patio-**T**emporal **Mixer** (HSTMixer), which leverages an all-MLP architecture for efficient and effective large-scale traffic forecasting. HSTMixer employs a hierarchical spatiotemporal mixing block to extract multi-resolution features through bottom-up aggregation and top-down propagation. Furthermore, an adaptive region mixer generates transformation matrices based on regional semantics, enabling our model to dynamically capture evolving spatiotemporal patterns for different regions. Extensive experiments conducted on four large-scale real-world datasets demonstrate that the proposed method not only achieves state-of-the-art performance but also exhibits competitive computational efficiency.

1 Introduction

Traffic forecasting is a fundamental task in intelligent transportation systems, serving as the base for applications like congestion mitigation and route navigation. Typically, the performance depends on how accurately it can capture spatiotemporal correlations within traffic data. Early models like ARIMA [Kumar and Vanajakshi, 2015], SVR [Castro-Neto *et al.*, 2009], and Kalman filtering [Guo *et al.*, 2014] rely on stationary assumptions, leading to limited representation ability. More recently, deep learning models, consisting of GNN-based models and transformer-based models et. al, have become the mainstream, as these models automatically capture more appropriate and dynamic spatiotemporal correlations from large amounts of data.

Despite the success of deep learning models, their compatible datasets are limited in scale compared to real-world traffic data. Applying these models to large-scale datasets remains a problem. Specifically, there are two key challenges: **Scalability**. Current deep-learning models exhibit quadratic computational complexity, causing an unacceptable increase in time and space costs as the data scale grows. **Effectiveness**. Existing models struggle to capture appropriate features when facing large-scale data. Each data sample is correlated with only a small subset of the dataset, while the remaining most act as noise. Current models often overlook this issue, indiscriminately constructing global correlations among samples, which scatters key features among substantial irrelevant noise. As shown in Figure 1, this issue is less evident in small datasets. However, as the scale of dataset increases, the proportion of noise grows significantly, making it increasingly difficult for models to capture key features effectively.



(a) Attention of METR. (b) Similarity of METR.

(c) Attention of GBA. (d) Similarity of GBA.

Figure 1: Attention performance on graphs of different scales. In small-scale graphs with fewer nodes, attention can effectively capture the correlations between nodes. However, as the number of nodes increases in large-scale, the presence of significant noise dilutes the effectiveness.

Pioneering works address above challenges from multiple perspectives. For scalability, there is a tendency to employ linear models, consisting of linear attentions [Liang *et al.*, 2023] and MLP-Mixers [Zhang *et al.*, 2023; Yeh *et al.*, 2024]. Among them, MLP-Mixers stand out for its lightweight and lower training costs. However, above efforts process all samples indiscriminately, introducing massive noise through

*Corresponding author: jywang@buaa.edu.cn

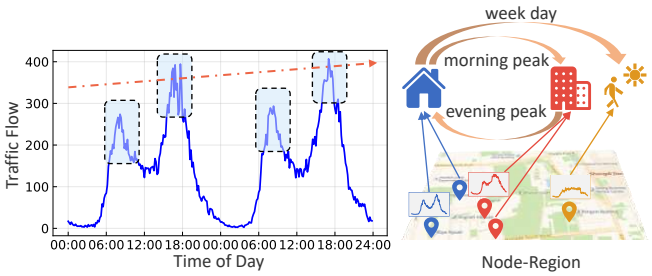


Figure 2: Spatiotemporal Hierarchy. At the macro-level, temporal data is driven by periodicity and trends (left), while spatial data is influenced by regional correlations, such as peak hours between residential and work areas or increased weekend traffic between residential and park areas (right). At the micro-level, neighboring samples exhibit similar values.

global feature integration, which limits their effectiveness on large-scale data. To mitigate this, some approaches [Fang *et al.*, 2024] group highly correlated samples into patches, enabling information exchange at the patch level. While this reduces computational complexity and partially addresses effectiveness, predefined patches fail to capture the evolving correlations of spatiotemporal data. *Thus, developing a linear model that dynamically identifies key features is essential.* Yet, explicitly defining key features is challenging, as they are shaped by complex and evolving spatiotemporal patterns.

Fortunately, spatiotemporal data exhibit a hierarchical structure, which becomes more evident with increasing data scale. As illustrated in Figure 2, at the micro-level, local proximity prevails, with neighboring samples sharing similar values. At the macro-level, temporal data is driven by tendency and periodicity, while spatial data is influenced by semantic relevance. In summary, spatiotemporal data at different data resolutions exhibit distinct dominant patterns, generating specific key features. Such hierarchy is essential to address large-scale data problems, as these dominant patterns enable models to prioritize key samples, effectively filtering out irrelevant noise. Moreover, extracting key features at each individual data resolution is more effective and manageable than directly capturing global key features from the entire dataset. Motivated by the above analysis, we designed a linear model based on a hierarchical paradigm.

Technically, we proposed HSTMixer, a Hierarchical Spatio-Temporal Mixer model built entirely on Multi-Layer Perceptrons (MLPs). The core of HSTMixer is the spatiotemporal mixing block, which hierarchically extracts and integrates spatiotemporal features. This block comprises a temporal aggregation mixer for temporal local features extraction and a spatial cascade mixer modeling hierarchical region-level spatial features. Furthermore, we design an adaptive region mixer that dynamically mixes features based on regional semantics. As a linear model, HSTMixer is computationally efficient and effective to large-scale datasets. By focusing on region-level dependencies, HSTMixer reduces the noise introduced by global modeling. Our contributions are summarized as follows:

- We propose HSTMixer for large-scale traffic prediction. The model generates hierarchical spatiotemporal features through bottom-up aggregation and integrates these features with a top-down propagation mechanism.
- We design an adaptive region mixer that generates transformation matrices based on regional semantics, enabling to dynamically model spatiotemporal patterns for different regions.
- Extensive experiments are conducted on four large-scale datasets, and HSTMixer consistently outperforms state-of-the-arts, with average improvements of 4.41%, 3.15%, and 2.03% in MAE, RMSE, and MAPE, respectively.

2 Related Work

Traffic Forecasting aims to predict traffic dynamics, e.g., traffic flow [Wang *et al.*, 2022; Ji *et al.*, 2023a; Ji *et al.*, 2022], traffic speed [Liu *et al.*, 2024b; Wang *et al.*, 2016]. Early deep learning-based methods leverage Recurrent Neural Networks (RNNs) [Li *et al.*, 2018; Bai *et al.*, 2020] or Temporal Convolutional Networks (TCNs) [Yu *et al.*, 2018; Wu *et al.*, 2019] to capture temporal dependencies, while Graph Neural Networks (GNNs) are used to model spatial dependencies [Ji *et al.*, 2020; Han *et al.*, 2025; Ji *et al.*, 2023b]. Transformer-based models have been widely applied to various urban spatio-temporal tasks, such as trajectory representation [Jiang *et al.*, 2023b; Wang *et al.*, 2025b; Yu *et al.*,], POI representation [Cheng *et al.*, 2025], and general time-series forecasting [Wang *et al.*, 2023; Wang *et al.*, 2020]. Recently, they have also gained considerable popularity in traffic forecasting [Jiang *et al.*, 2023a; Ji *et al.*, 2025]. However, the intensive convolution operations of GNNs [You *et al.*, 2020] and the quadratic complexity of transformers limit their ability on large-scale urban.

Large-Scale Traffic Forecasting presents significant scalability challenges due to its extensive spatial scope, demanding models that are both efficient and effective. While some pioneering works reduced computational costs through graph decomposition [Wang *et al.*, 2024a; Fang *et al.*, 2024; Wang *et al.*, 2024c] or prior assumptions [Duan *et al.*, 2024], they may compromise information integrity. More recently, MLP-Mixer Models have garnered increasing attention in both computer vision [Tolstikhin *et al.*, 2021] and natural language processing [Fusco *et al.*, 2023] due to their simplicity, effectiveness, and scalability for large-scale data. MLPST [Zhang *et al.*, 2023] first applied the MLP-Mixer architecture to grid-based spatiotemporal data. TimeMixer [Wang *et al.*, 2024b] performs node-level traffic forecasting by feature decomposition, but it lacks consideration of spatial correlations. TSMixer [Chen *et al.*, 2023] and RPMixer [Yeh *et al.*, 2024] mix temporal and feature (i.e. spatial) dimensions, effectively capturing spatiotemporal dependencies. Despite their efficiency, these methods indiscriminately construct global correlations across all data samples. When facing large-scale data, it will overwhelm key features with substantial irrelevant noise.

3 Preliminary

Traffic Data. Traffic data consists of a series of numerical values collected and recorded by sensors in the urban traffic network (e.g., traffic flow, speed, and density). For the

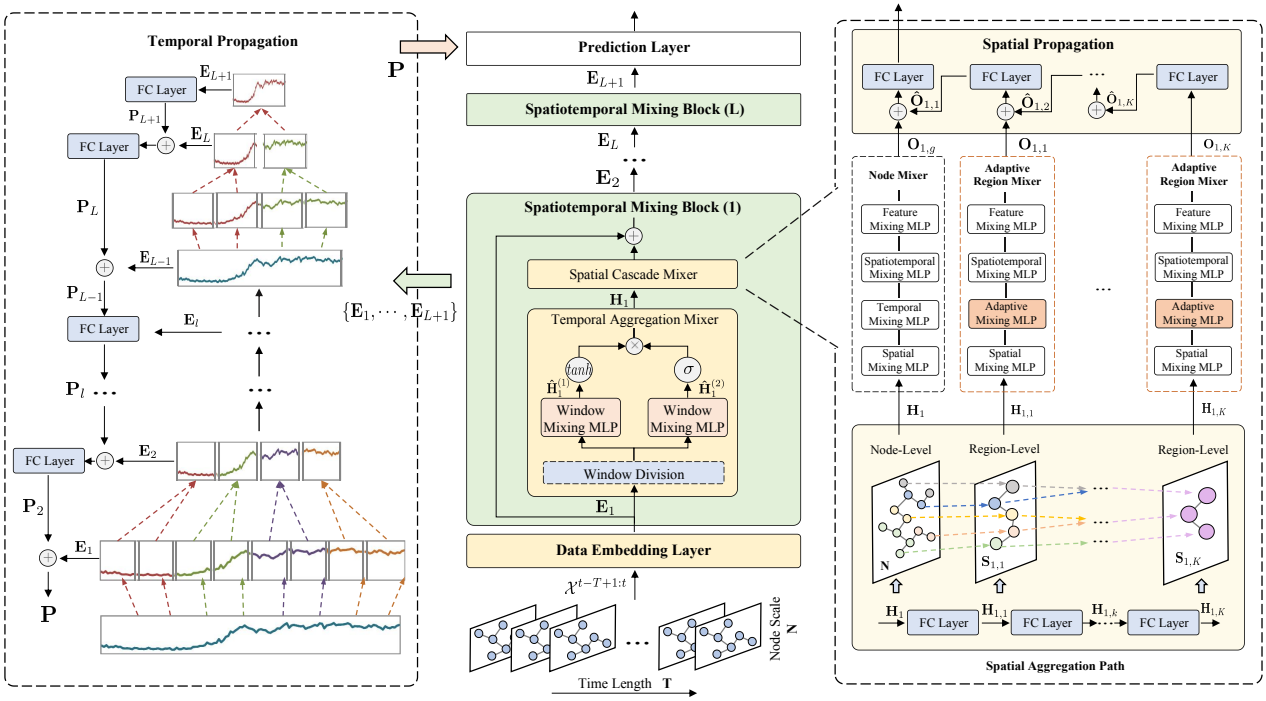


Figure 3: The overall framework of HSTMixer.

node n, $\mathbf{x}_n^h$ denotes traffic data at the h-th time slice. For a traffic network with N nodes, $\mathbf{X}^h \in \mathbb{R}^N$ denotes overall traffic data at the h-th time slice. Further, $\mathcal{X}^{t-T+1:t}$ denotes a traffic data series from $t-T+1$ to t, where $\mathcal{X}^{t-T+1:t} = \left(\mathbf{X}^{t-T+1}, \cdots, \mathbf{X}^t\right) \in \mathbb{R}^{N \times T}$.

Traffic Forecasting. Given a traffic data series $\mathcal{X}^{t-T+1:t}$ with its next T' steps $\mathcal{Y}^{t+1:t+T'} = \mathcal{X}^{t+1:t+T'}$. The objective of traffic forecasting is to learn a function $\mathcal{F}$ that predicts the future traffic data:

$$\hat{\mathcal{Y}}^{t+1:t+T'} = \mathcal{F}(\mathcal{X}^{t-T+1:t}, \Theta), \tag{1}$$

where $\hat{\mathcal{Y}}^{t+1:t+T'}$ is the prediction to $\mathcal{Y}^{t+1:t+T'}$, and Θ denote the parameters to learn.

4 Methodology

Based on above analysis, the primary motivation for large-scale traffic forecasting is to leverage the hierarchy of spatiotemporal data. Therefore, we propose our HSTMixer, a hierarchical MLP-Mixer model. As shown in Figure 3, our HSTMixer consists of three parts: the data embedding layer for input embedding, the stacked spatiotemporal mixing blocks (ST mixing blocks) for multi-scale features extraction, and the prediction layer for task prediction. Details of each part are described in the following sections.

4.1 Data Embedding Layer

In the data embedding layer, raw traffic data is first transformed into high-dimensional embeddings and then integrated with the spatial and temporal embeddings.

The spatial embeddings reflect the correlations between nodes in the traffic network, comprising static and dynamic components. Specifically, the static component consists of node embeddings extracted from network topology through Node2Vec [Grover and Leskovec, 2016], denoted as $\mathbf{E}_{\text{static}} \in \mathbb{R}^{N \times d}$, where d is the feature dimension. The dynamic component includes learnable vectors $\mathbf{E}_{\text{dynamic}} \in \mathbb{R}^{N \times d}$ to capture dynamic variations of each node. The overall spatial embeddings $\mathbf{E}_{sp} \in \mathbb{R}^{N \times d}$ are then obtained by:

$$\mathbf{E}_{sp} = \mathbf{E}_{\text{static}} + \mathbf{E}_{\text{dynamic}}. \tag{2}$$

The temporal embeddings complement periodic patterns for traffic data, including two temporal components: minute-of-day (1 to 60×24) embeddings $\mathbf{E}_{\text{day}} \in \mathbb{R}^{T \times d}$ and day-of-week (1 to 7) embeddings $\mathbf{E}_{\text{week}} \in \mathbb{R}^{T \times d}$. Subsequently, the temporal embeddings $\mathbf{E}_{te} \in \mathbb{R}^{T \times d}$ are computed as:

$$\mathbf{E}_{te} = \mathbf{E}_{\text{day}} + \mathbf{E}_{\text{week}}. \tag{3}$$

Overall, the raw traffic data $\mathcal{X}^{t-T+1:t}$ is first transformed by a fully connected (FC) layer into $\mathbf{E}_{tr} \in \mathbb{R}^{N \times T \times d}$. Then, $\mathbf{E}_{tr}$ incorporates $\mathbf{E}_{sp}$ and $\mathbf{E}_{te}$ to generate the input embeddings $\mathbf{E}_1 \in \mathbb{R}^{N \times T \times d}$ for subsequent ST mixing blocks:

$$\mathbf{E}_1 = \mathbf{E}_{tr} + \mathbf{E}_{sp} + \mathbf{E}_{te}. \tag{4}$$

4.2 Spatiotemporal Mixing Block

Motivation. Spatiotemporal data exhibits hierarchical characteristics, with dominated patterns vary with data resolutions. Our method aims to generate multi-resolution features through a bottom-up feature aggregation mechanism. The lower layers retain local characteristics, while the higher layers progressively incorporate global contextual features. This structured progression captures the transition between local and global properties and fully models patterns at different spatial scales. Specifically, we proposed the spatiotemporal mixing block (ST mixing block) based on the MLP-Mixer. By stacking multiple ST mixing blocks, hierarchical feature pyramids are generated efficiently.

Overall Structure. As shown in Figure 3, each ST mixing block comprises two sequential modules: a temporal aggregation mixer and a spatial cascade mixer. The temporal mixer captures local temporal features and aggregates them to generate macro-level features, forming a temporal feature pyramid across ST mixing blocks. The spatial cascade mixer focuses on spatial hierarchies, employing a cascade of mixers to construct a spatial feature pyramid and capture spatiotemporal patterns at specific time scale within each ST mixing block. Stacking multiple ST mixing blocks produces spatiotemporal features at various resolutions, which are integrated to obtain the final spatiotemporal representation.

Supposed $\mathbf{E}_l \in \mathbb{R}^{N \times T_{l-1} \times d}$ denotes the input of the l-th ST mixing block, where N is the number of nodes and T_{l-1} is the temporal length. Initially, the temporal aggregation mixer takes $\mathbf{E}_l$ as input and outputs $\mathbf{H}_l$. Then, $\mathbf{H}_l$ is processed by the spatial cascade mixer.

Temporal Aggregation Mixer

Firstly, $\mathbf{E}_l$ is divided into multiple p-length windows, where $\mathbf{E}_l \in \mathbb{R}^{N \times (T_l \times p) \times d}$, and $T_l = \lceil T_{l-1}/p \rceil$ is the window size. Each window is processed by a shared structure which comprises two parallel window mixing MLPs. Each MLP applies two FC layers along the window-feature dimension ($p \times d$).

For each window mixing MLP, $\mathbf{E}_l$ is aggregated as $\tilde{\mathbf{E}}_l \in \mathbb{R}^{N \times T_l \times h}$ by the first FC layer, where h is the intermediate feature dimension. Further, the positional embedding $\mathbf{E}_{\text{pe}} \in \mathbb{R}^{N \times T_l \times h}$ is added to $\tilde{\mathbf{E}}_l$ to identify the position information of each window. The second FC layer processes $\tilde{\mathbf{E}}_l$ and outputs $\hat{\mathbf{H}}_l \in \mathbb{R}^{N \times T_l \times d}$:

$$\hat{\mathbf{H}}_l = \text{FC}_2\left(\phi\left(\text{FC}_1\left(\mathbf{E}_l\right) + \mathbf{E}_{\text{pe}}\right)\right), \tag{5}$$

where $\phi(\cdot)$ is the activation function. Considering there are two independent window mixing MLPs, let $\hat{\mathbf{H}}_l^{(1)}$ and $\hat{\mathbf{H}}_l^{(2)}$ denote the output of these two MLPs respectively. To improve the expressive power of the model, we employ the gated mechanism [Dauphin *et al.*, 2017] to generate the output $\mathbf{H}_l \in \mathbb{R}^{N \times T_l \times d}$ of the temporal aggregation mixer.

$$\mathbf{H}_l = \tanh\left(\hat{\mathbf{H}}_l^{(1)}\right) \odot \sigma\left(\hat{\mathbf{H}}_l^{(2)}\right), \tag{6}$$

where $\tanh(\cdot)$ represents the tanh function, $\sigma(\cdot)$ denotes the sigmoid function, and $\odot$ is dot-product.

Unlike typical Mixers that preserve the input's shape, our temporal aggregation mixers integrate features within the same window. When p is small, values within each window exhibit gentle variations, dominated by the local similarity characteristics. As a result, feature aggregation can effectively capture local patterns and progressively adjust feature scale by reducing the number of windows across ST mixing blocks. For subsequent blocks, each window progressively contains richer information and reveals macro-level correlations like tendency. After the l-th layer, the input $\mathbf{E}_l \in \mathbb{R}^{N \times (T_l \times p) \times d}$ is aggregated as $\mathbf{H}_l \in \mathbb{R}^{N \times T_l \times d}$.

Spatial Cascade Mixer

Overall Structure. Similar to temporal correlations, spatial correlations are also multi-scale. At the node-level, correlations are dominated by local similarity. While at the region-level, they depend on the scale and semantics of regions.

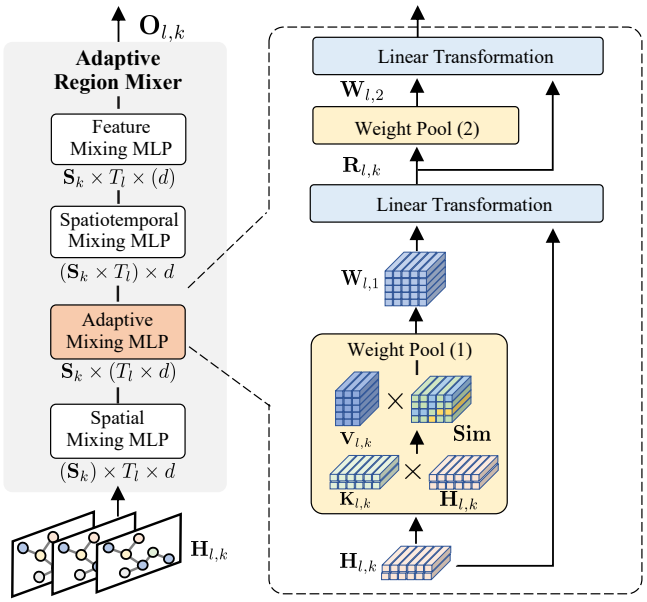


Figure 4: Structure of the Adaptive Region Mixer, where $(S_k) \times T_l \times d$ represents mixing along the spatial dimension (S_k).

Therefore, we designed the spatial cascade mixer, consisting of a spatial aggregation path, a node mixer and K adaptive region mixers corresponding to K region scales.

The spatial aggregation path consists of K FC layers. Given the dynamic nature of spatial correlations and the lack of predefined regions, $\mathbf{H}_l \in \mathbb{R}^{N \times T_l \times d}$ is first progressively aggregated into a set of region-level features across different scales through the aggregation path, represented as $\{\mathbf{H}_{l,1}, \cdots, \mathbf{H}_{l,K}\}$. Suppose $\mathbf{H}_{l,k} \in \mathbb{R}^{S_k \times T_l \times d}$ is the input of the k-th adaptive region mixer, S_k is the amount of regions in this mixer, and $(S_{k+1} < S_k < \cdots < S_1 < N)$.

Subsequently, each mixer processes the input along spatial, temporal, and feature dimensions. The specific structure of each mixer is illustrated in Figure 3. Differed from the node mixer, the region mixer employs an adaptive mixing MLP to capture each region's unique features. As each region represents a distinct semantic unit, the adaptive mixing MLP ensures semantically similar regions share similar parameters, and vice versa. Each mixer generates its individual output, and the final output integrates features from both node-level and multiple region-level features.

Adaptive Region Mixer. As shown in Figure 4, the k-th adaptive region mixer processes $\mathbf{H}_{l,k} \in \mathbb{R}^{S_k \times T_l \times d}$ through four MLPs: the spatial mixing MLP, the adaptive mixing MLP, the spatiotemporal mixing MLP, and the feature mixing MLP to capture spatiotemporal dependencies at the corresponding scale. Notably, the adaptive mixing MLP is different as it employs a parameter pool to adaptively generate transformation matrices tailored to each region's semantics.

Specifically, the parameter pool contains $\mathbf{K}_{l,k} \in \mathbb{R}^{M_k \times T_l}$ as keys and $\mathbf{V}_{l,k} \in \mathbb{R}^{M_k \times T_l \times h}$ as base weights, where h is the intermediate feature dimension and M_k is the size of base weights. Then, the demanding parameters $\mathbf{W}_{l,k} \in \mathbb{R}^{S_k \times T_l \times h}$

are generated as follows:

$$\begin{aligned}\mathbf{Sim} &= \text{Softmax}(\mathbf{H}_{l,k} * \mathbf{K}_{l,k}^{\top}),\\ \mathbf{W}_{l,k} &= \textstyle\sum_{i=1}^{d} \mathbf{Sim}^{(i)} * \mathbf{V}_{l,k},\end{aligned} \tag{7}$$

where $*$ denotes the matrix multiplication, $\mathbf{Sim} \in \mathbb{R}^{S_k \times d \times M_k}$ are weight scores, and $\mathbf{Sim}^{(i)}$ is the i-th tensor of $\mathbf{Sim}$ over the feature dimension (d).

In the adaptive mixing MLP, both two FC layers are generated from the parameter pool, denoted as $\mathbf{W}_{l,k,1}$ and $\mathbf{W}_{l,k,2}$. Then, the input $\mathbf{H}_{l,k}$ is transformed as:

$$\begin{aligned}\mathbf{R}_{l,k}^{(j)} &= \mathbf{H}_{l,k}^{(j)\top} * \mathbf{W}_{l,k,1}^{(j)},\\ \mathbf{H}_{l,k}^{(j)} &:= \mathbf{W}_{l,k,2}^{(j)} * \mathbf{R}_{l,k}^{(j)\top},\end{aligned} \tag{8}$$

where $\mathbf{H}_{l,k}^{(j)}$, $\mathbf{W}_{l,k,1}^{(j)}$ are both the j-th tensor over the spatial dimension, and $\mathbf{R}_{l,k} \in \mathbb{R}^{S_k \times d \times h}$ denotes the hidden state.

To increase the semantic distinctiveness of each region, we employ the parameter orthogonal loss (POL) during training. It minimizes the similarity between each $\mathbf{W}_{l,k}^{(i)}$ and $\mathbf{W}_{l,k}^{(j)}$ to decrease the semantic similarity among regions. Specifically, the POL in the l-th ST mixing block can be formalized as:

$$\text{POL}_l = \textstyle\sum_{k=1}^{K} \sum_{i=1}^{S_k} \sum_{j=1}^{S_k} \frac{1}{S_k^2} \text{CS}(\mathbf{W}_{l,k}^{(i)}, \mathbf{W}_{l,k}^{(j)}), \tag{9}$$

where $\text{CS}(\cdot)$ represents the cosine similarity. The total orthogonal loss can be denoted as:

$$\mathcal{L}_{\text{POL}} = \textstyle\sum_{l=1}^{L} \text{POL}_l. \tag{10}$$

Output. The spatial cascade mixer generates an output set, including $\{\mathbf{O}_{l,g}, \mathbf{O}_{l,1}, \cdots, \mathbf{O}_{l,K}\}$, where $\mathbf{O}_{l,g} \in \mathbb{R}^{N \times T_l \times d}$ belongs to the node mixer, and $\mathbf{O}_{l,k} \in \mathbb{R}^{S_k \times T_l \times d}$ belongs to the k-th adaptive region mixer.

Hierarchical Feature Propagation

Since HSTMixer processes spatiotemporal data hierarchically, it generates various feature pyramids: **(1)** Stacked ST mixing blocks form a feature pyramid across temporal resolutions, and the higher features reflect the trend of the lower features. **(2)** In each ST mixing block, the spatial cascade mixer generates a feature pyramid across spatial resolutions, and the higher regions reflect the interactions between lower nodes. Therefore, the hierarchical feature propagation is proposed for feature integration. It contains two top-down paths: the spatial propagation and the temporal propagation.

Spatial Propagation. The top-down spatial feature propagation starts with the output $\mathbf{O}_{l,K}$. It is transformed by an FC layer into $\hat{\mathbf{O}}_{l,K} \in \mathbb{R}^{S_{k-1} \times T_l \times d}$. As shown in Figure 3, $\mathbf{O}_{l,k}$ is transformed into $\hat{\mathbf{O}}_{l,k}$ as follows:

$$\begin{aligned}\hat{\mathbf{O}}_{l,k} &= \text{FC}\left(\mathbf{O}_{l,k} + \hat{\mathbf{O}}_{l,k+1}\right),\\ \mathbf{E}_{l+1} &= \text{FC}\left(\mathbf{O}_{l,g} + \hat{\mathbf{O}}_{l,1}\right) + \mathbf{E}_l,\end{aligned} \tag{11}$$

where $\mathbf{E}_{l+1}$ denotes the final result of the spatial propagation, which serves as the input of the $(l+1)$-th ST mixing block.

Temporal Propagation. Stacked ST mixing blocks will generate a feature pyramid across temporal resolutions, denoted as $\{\mathbf{E}_1, \cdots, \mathbf{E}_l, \cdots, \mathbf{E}_{L+1}\}$. Additionally, $\mathbf{E}_{L+1}$ is the output of the L-th ST mixing block. Similar as the spatial propagation, temporal propagation begins with $\mathbf{E}_{L+1}$, which is first projected by an FC layer into $\mathbf{P}_{L+1}$. As shown in Figure 3, $\mathbf{E}_l$ is transformed into $\mathbf{P}_l \in \mathbb{R}^{N \times T_l \times d}$ as follows:

$$\begin{aligned}\mathbf{P}_l &= \text{FC}\left(\mathbf{E}_l + \mathbf{P}_{l+1}\right),\\ \mathbf{P} &= \text{FC}\left(\mathbf{E}_1 + \mathbf{P}_2\right),\end{aligned} \tag{12}$$

where $\mathbf{P} \in \mathbb{R}^{N \times T \times d}$ denotes the final result after the temporal propagation.

Datasets	Nodes	Time steps	Time Range
SD	716	35040	1/1/2019-1/1/2020
GBA	2352	35040	1/1/2019-1/1/2020
GLA	3834	35040	1/1/2019-1/1/2020
CA	8600	35040	1/1/2019-1/1/2020

Table 1: Statistics of datasets.

4.3 Prediction Layer

The prediction layer, takes $\mathbf{P} \in \mathbb{R}^{N \times T \times d}$ and $\mathbf{E}_{L+1} \in \mathbb{R}^{N \times T_L \times d}$ as inputs and regresses the prediction values as:

$$\hat{\mathcal{Y}}^{t+1:\,t+T'} = \text{FC}_3\left(\phi\left(\text{FC}_2\left(\mathbf{P} + \text{FC}_1\left(\mathbf{E}_{L+1}\right)\right)\right)\right). \tag{13}$$

To optimize the model, we employ the mean absolute error (MAE) as the regression loss, which is denoted as $\mathcal{L}_{REG}$. The overall loss combines the regression loss and the parameter orthogonal loss, which can be formalized as $\mathcal{L}$:

$$\mathcal{L} = \alpha \mathcal{L}_{REG} + \beta \mathcal{L}_{POL}, \tag{14}$$

where α and β are hyper-parameters for weight balance.

5 Experiments

5.1 Experiment Setup

Datasets. We evaluate HSTMixer on four datasets within LargeST [Liu *et al.*, 2024a], including SD, GBA, GLA, and CA. Details of each dataset are described in in Table. 1. Specifically, we aggregate the time intervals from 5-minutes into 15-minutes. Additionally, we divide each dataset according to four seasons, and then splite the training, validation and test sets with the ratio of 6:2:2 in each season.

Baselines. The HSTMixer is compared with 20 baseline models, including 11 *Previous SOTAs* and 9 *Scalable Models for Large-Scale Settings*. Details of each baseline are described in Appendix. A.1.

Experiment Details. All experiments are conducted on Ubuntu 20.04.6 LTS with an NVIDIA RTX A6000 48GB GPU, based on the LibCity [Wang *et al.*, 2021]. Our HSTMixer takes 12 historical time slices as input to predict the next 12 time slices. We adopted a grid search to determine the optimal hyperparameters, as provided in Appendix. A.1.

Metrics. We use the mean absolute error (MAE), the root mean square error (RMSE) and the mean absolute percentage error (MAPE) as evaluation metrics.

Model	SD			GBA			GLA			CA		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN [Li *et al.*, 2018]	17.96	28.65	11.85	21.86	34.65	20.19	23.36	36.19	14.19	28.24	44.01	22.23
STGCN [Yu *et al.*, 2018]	16.87	28.84	9.54	19.96	32.78	16.27	18.61	30.18	11.44	17.87	29.14	13.56
ASTGCN [Guo *et al.*, 2019]	19.99	32.19	12.99	22.97	36.35	20.39	21.46	34.21	13.29	–	–	–
GWNET [Wu *et al.*, 2019]	16.93	27.97	9.74	19.73	31.99	17.54	18.94	30.16	11.58	19.73	31.32	15.42
STGODE [Fang *et al.*, 2021]	17.55	28.67	11.25	19.73	32.45	16.28	18.86	30.49	11.81	17.76	29.13	13.37
DSTAGNN [Lan *et al.*, 2022]	17.37	27.68	11.27	20.14	32.63	16.37	18.62	30.00	11.20	–	–	–
D2STGNN [Shao *et al.*, 2022b]	16.49	26.59	9.52	18.56	31.12	12.92	–	–	–	–	–	–
HIEST [Ma *et al.*, 2023]	17.94	29.03	10.33	22.50	35.75	14.02	22.37	35.26	11.84	22.06	35.03	14.18
PDFormer [Jiang *et al.*, 2023a]	15.84	25.91	9.70	–	–	–	–	–	–	–	–	–
DGCRN [Li *et al.*, 2023]	15.50	25.90	9.93	19.11	31.21	17.17	–	–	–	–	–	–
TESTAM [Lee and Ko, 2024]	18.51	30.42	10.42	19.57	32.32	13.94	–	–	–	–	–	–
STID [Shao *et al.*, 2022a]	18.21	29.97	10.18	20.82	34.43	12.93	19.83	32.33	10.38	19.11	31.45	11.68
FreTS [Yi *et al.*, 2023]	23.95	37.12	14.89	26.87	40.55	18.39	27.76	42.00	15.85	27.58	42.50	18.47
TSMixer [Chen *et al.*, 2023]	16.81	29.79	10.52	19.93	34.83	16.56	17.86	30.89	10.96	16.78	29.42	12.43
TimeMixer [Wang *et al.*, 2024b]	17.15	28.30	9.74	21.74	35.15	13.85	19.76	31.96	10.81	18.16	30.11	11.31
RPMixer [Yeh *et al.*, 2024]	16.72	29.64	10.63	19.05	32.75	15.26	17.57	30.42	10.56	16.75	29.28	12.36
EiFormer [Sun *et al.*, 2025]	17.90	32.06	10.79	20.39	35.35	15.58	18.91	33.05	11.00	18.22	32.07	12.89
GSNet [Kong *et al.*, 2025]	16.65	27.33	9.43	19.37	31.98	13.11	18.81	30.65	9.89	17.61	29.15	10.88
LSTNN [Wang *et al.*, 2025a]	15.83	27.08	9.44	18.28	31.59	12.99	17.22	29.11	9.65	16.48	28.24	10.90
HSTMixer	**14.80**	**25.06**	**9.22**	**17.73**	**30.67**	**12.71**	**16.45**	**28.03**	**9.53**	**15.55**	**27.05**	**10.55**

Table 2: Prediction performance of each model.The best results are highlighted in **bold**, while the second-best results are underlined. Additionally, due to the limited scalability of GNN-based and transformer-based methods, some methods fail to handle large-scale datasets, with their results denoted as a dash (–) in the table.

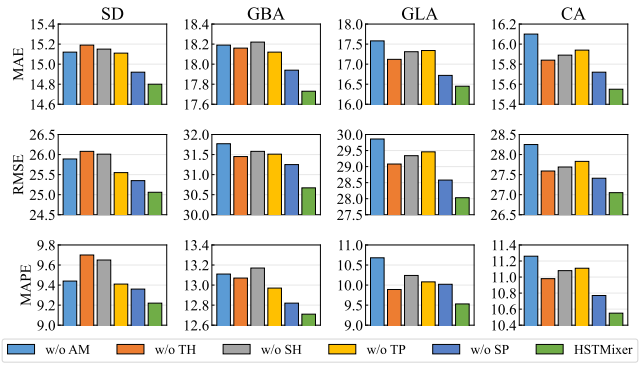


Figure 5: Results of ablation study. Verified the effectiveness of each component.

5.2 Performance Comparisons

Table. 2 presents the overall performance of our model and all the baselines with respect to MAE, RMSE, and MAPE on four datasets. Additionally, we run all models five times and report the mean results. Overall, HSTMixer consistently outperforms other baselines on all four datasets. Specifically, HSTMixer obtains average improvements of 4.41%, 3.15%, and 2.03% beyond the best baselines in MAE, RMSE and MAPE, respectively. Furthermore, based on the experimental results, we can make the following observations:

Hierarchical Processing Benefits Performance. In most cases, the performance of scalable baselines is much inferior to previous transformer-based models. However, HSTMixer enables a linear MLP-Mixer model to outperform complex transformer-based baselines across all four datasets. This is due to hierarchical processing, which allows the model to learn dominant features at different scales, generating more effective representations.

Advantages of Scalability. The experiments explicitly validate the scalability challenges inherent in many previous SOTAs. Several prominent GNN- and transformer-based models are unable to complete the experiments on larger datasets due to prohibitive computational and memory costs. In contrast, HSTMixer demonstrates excellent scalability, efficiently handling all datasets.

Advantages of Robustness. The performance gap between HSTMixer and the second-best baseline widens with larger datasets. On the GBA dataset, HSTMixer outperforms the second-best baseline by 1.62% in MAPE. On the largest dataset, this gap increases to 3.03% in MAPE. This demonstrates HSTMixer's robustness with large-scale datasets.

5.3 Ablation Study

We conduct ablation studies to analyze the effectiveness of each component in HSTMixer. These five variants are listed as below: **(1)** *w/o AM:* Replaces the adaptive mixing MLP with a standard mixing MLP. **(2)** *w/o TH:* Removes the modeling of temporal hierarchy, i.e., the time length in each ST mixing block remains the same. **(3)** *w/o SH:* Removes the modeling of spatial hierarchy, i.e., the number of nodes in the spatial cascade mixer remains the same. **(4)** *w/o TP:* Removes the temporal propagation. **(5)** *w/o SP:* Removes the spatial propagation. Ablation results are shown in Figure. 5, which can be concluded as:

Benefits Brought by Hierarchical Modeling. *w/o SH* and *w/o TH* are both inferior to HSTMixer, which demonstrates the effectiveness of hierarchical modeling in both spatial and temporal dimensions. Additionally, HSTMixer performs better than *w/o AM*, because adaptive mixing improves semantic differences between regions, resulting in more structured hierarchical features.

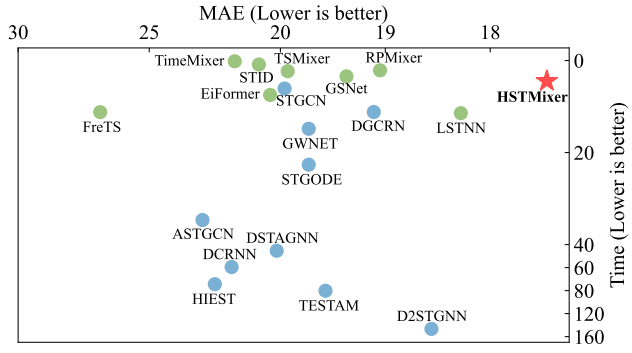


Figure 6: MAE and the total training time (Unit: hours) of each model on the GBA dataset.

Effectiveness of Feature Propagation. As HSTMixer outperforms *w/o TP* and *w/o SP*, it indicates that features at different scales contain distinct semantic information. The feature propagation enables multi-scale features to complement each other, providing more comprehensive representations.

5.4 Efficiency Analysis

In this section, we evaluate HSTMixer and baselines in terms of performance and efficiency, and present results on MAE and the total training time on the GBA dataset. As shown in Figure. 6, HSTMixer significantly outperforms linear baselines and even surpasses SOTA transformer-based models (e.g., EiFormer). Meanwhile, HSTMixer maintains efficiency comparable to mainstream scalable models (e.g., RPMixer, TSMixer). This demonstrates that HSTMixer balances performance and efficiency, offering excellent scalability on large-scale datasets. Additionally, we report the specific training and inference costs in Appendix. A.2.

Further, we analyze the computational cost of HSTMixer. Specifically, the mixing operation mainly consists of two linear transformations with a time complexity of $O(dhNT)$. Generating adaptive transformation weights incurs a complexity of $O(MdhNT)$, and applying these weights for linear transformations adds another $O(dhNT)$. Consequently, the total complexity is $O(LKMdhNT)$, where L and K represent the number of ST mixing blocks and region scales, respectively. This linear complexity with respect to N ensures scalability for large-scale spatiotemporal prediction.

5.5 Parameter Sensitivity

We conduct sensitivity experiments on two key hyperparameters: the size of base weights in the first region scale M_1 and the length of temporal window p. The results are illustrated in Figure. 7. The experimental results reveal similar trends across different datasets. Specifically, the model performance improves as M_1 increases, reaching its peak when $M_1 = 32$. However, further increasing M_1 to 64 results in a performance decline. This decline can be attributed to the increased optimization difficulty caused by the larger number of transformation weights. Regarding the length of temporal window p, HSTMixer achieves the best performance when $p = 2$. As p increases, performance gradually decreases. This is because the input historical time steps are fixed at 12, increasing p significantly reduces the number of temporal scales, thereby

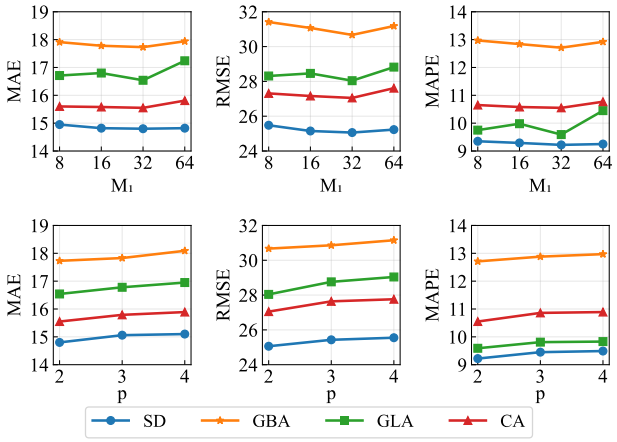
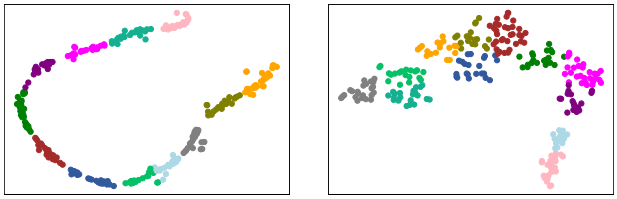


Figure 7: Sensitivity experiment results for M_1 and p.



(a) Visualization of HSTMixer. (b) Visualization of *w/o AM*.

Figure 8: t-SNE visualization of embedding $\mathbf{O}_{L,1}$ of HSTMixer and *w/o AM* on GBA.

weakening the model's ability to capture hierarchical temporal dependencies, leading to degraded performance. Furthermore, we conducted sensitivity experiments on the depth of ST mixing blocks L and the number of region scales K, with the detailed results presented in Appendix. A.2.

5.6 Visualization

To evaluate HSTMixer's ability to capture regional semantics, we leverage t-SNE to visualize the spatial representations $\mathbf{O}_{L,1}$ produced by the HSTMixer and its *w/o AM* variant. As illustrated in Figure 8, the representations generated by HSTMixer are more compact with well-defined boundaries between clusters. In contrast, the *w/o AM* variant yields representations with substantial overlap between clusters. This comparison highlights HSTMixer's greater representation ability, which ensures nodes within the same region type share similar spatiotemporal patterns, while nodes across different types maintain distinct characteristics.

6 Conclusion

We present HSTMixer, a simple yet effective MLP-Mixer model, which leverages hierarchical modeling to tackle large-scale traffic forecasting. HSTMixer surpasses all current baselines and introduces enhancements to the MLP-Mixer architecture, making it competitive with more complex transformer-based models. Comprehensive experiments on extensive traffic datasets demonstrate its scalability and effectiveness. Future work will extend its application to multivariate time series prediction and anomaly detection.

References

[Bai *et al.*, 2020] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33:17804–17815, 2020.

[Castro-Neto *et al.*, 2009] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee D Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications*, 36(3):6164–6173, 2009.

[Chen *et al.*, 2023] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan Ö. Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *CoRR*, abs/2303.06053, 2023.

[Cheng *et al.*, 2025] Jiawei Cheng, Jingyuan Wang, Yichuan Zhang, Jiahao Ji, Yuanshao Zhu, Zhibo Zhang, and Xiangyu Zhao. Poi-enhancer: An llm-based semantic enhancement framework for poi representation learning. In *AAAI*, 2025.

[Dauphin *et al.*, 2017] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*. PMLR, 2017.

[Duan *et al.*, 2024] Wenying Duan, Tianxiang Fang, Hong Rao, and Xiaoxi He. Pre-training identification of graph winning tickets in adaptive spatial-temporal graph neural networks. In *30th KDD*, 2024.

[Fang *et al.*, 2021] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ODE networks for traffic flow forecasting. In *KDD*. ACM, 2021.

[Fang *et al.*, 2024] Yuchen Fang, Yuxuan Liang, Bo Hui, Zezhi Shao, Liwei Deng, Xu Liu, Xinke Jiang, and Kai Zheng. Efficient large-scale traffic forecasting with transformers: A spatial data management perspective. *arXiv preprint arXiv:2412.09972*, 2024.

[Fusco *et al.*, 2023] Francesco Fusco, Damian Pascual, Peter W. J. Staar, and Diego Antognini. pnlp-mixer: an efficient all-mlp architecture for language. In *ACL (industry)*, pages 53–60. Association for Computational Linguistics, 2023.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM, 2016.

[Guo *et al.*, 2014] Jianhua Guo, Wei Huang, and Billy M Williams. Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, 43:50–64, 2014.

[Guo *et al.*, 2019] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, pages 922–929. AAAI Press, 2019.

[Han *et al.*, 2025] Chengkai Han, Jingyuan Wang, Yongyao Wang, Xie Yu, Hao Lin, Chao Li, and Junjie Wu. Bridging traffic state and trajectory for dynamic road network and trajectory representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[Ji *et al.*, 2020] Jiahao Ji, Jingyuan Wang, Zhe Jiang, Jingtian Ma, and Hu Zhang. Interpretable spatiotemporal deep learning model for traffic flow prediction based on potential energy fields. In *ICDM*, 2020.

[Ji *et al.*, 2022] Jiahao Ji, Jingyuan Wang, Zhe Jiang, Jiawei Jiang, and Hu Zhang. Stden: Towards physics-guided neural networks for traffic flow prediction. In *AAAI*, 2022.

[Ji *et al.*, 2023a] Jiahao Ji, Jingyuan Wang, Chao Huang, Junjie Wu, Boren Xu, Zhenhe Wu, Junbo Zhang, and Yu Zheng. Spatio-temporal self-supervised learning for traffic flow prediction. In *AAAI*, 2023.

[Ji *et al.*, 2023b] Jiahao Ji, Jingyuan Wang, Yu Mou, and Cheng Long. Multi-factor spatio-temporal prediction based on graph decomposition learning. *arXiv preprint arXiv:2310.10374*, 2023.

[Ji *et al.*, 2025] Jiahao Ji, Wentao Zhang, Jingyuan Wang, and Chao Huang. Seeing the unseen: Learning basis confounder representations for robust traffic prediction. *arXiv Preprint*, 2025.

[Jiang *et al.*, 2023a] Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. In *AAAI*, pages 4365–4373. AAAI Press, 2023.

[Jiang *et al.*, 2023b] Jiawei Jiang, Dayan Pan, Houxing Ren, Xiaohan Jiang, Chao Li, and Jingyuan Wang. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In *39th ICDE*, 2023.

[Kong *et al.*, 2025] Weiyang Kong, Kaiqi Wu, Sen Zhang, and Yubao Liu. Graphsparsenet: a novel method for large scale traffic flow prediction. *arXiv preprint arXiv:2502.19823*, 2025.

[Kumar and Vanajakshi, 2015] S Vasantha Kumar and Lelitha Vanajakshi. Short-term traffic flow prediction using seasonal arima model with limited input data. *European Transport Research Review*, 7:1–9, 2015.

[Lan *et al.*, 2022] Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. DSTAGNN: dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*. PMLR, 2022.

[Lee and Ko, 2024] Hyunwook Lee and Sungahn Ko. TESTAM: A time-enhanced spatio-temporal attention model with mixture of experts. In *ICLR*. OpenReview.net, 2024.

[Li *et al.*, 2018] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR (Poster)*. OpenReview.net, 2018.

[Li *et al.*, 2023] Fuxian Li, Jie Feng, Huan Yan, Guangyin Jin, Fan Yang, Funing Sun, Depeng Jin, and Yong Li. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Trans. Knowl. Discov. Data*, 17(1):9:1–9:21, 2023.

[Liang *et al.*, 2023] Yuxuan Liang, Yutong Xia, Songyu Ke, Yiwei Wang, Qingsong Wen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. Airformer: Predicting nationwide air quality in china with transformers. In *AAAI*, 2023.

[Liu *et al.*, 2024a] Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. Largest: A benchmark dataset for large-scale traffic forecasting. *NeurIPS*, 2024.

[Liu *et al.*, 2024b] Zehua Liu, Jingyuan Wang, Zimeng Li, and Yue He. Full bayesian significance testing for neural networks in traffic forecasting. In *33rd IJCAI*, 2024.

[Ma *et al.*, 2023] Qian Ma, Zijian Zhang, Xiangyu Zhao, Haoliang Li, Hongwei Zhao, Yiqi Wang, Zitao Liu, and Wanyu Wang. Rethinking sensors modeling: Hierarchical information enhanced traffic forecasting. In *CIKM*, 2023.

[Shao *et al.*, 2022a] Zezhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *31st CIKM*, 2022.

[Shao *et al.*, 2022b] Zezhi Shao, Zhao Zhang, Wei Wei, Fei Wang, Yongjun Xu, Xin Cao, and Christian S. Jensen. Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. *Proc. VLDB Endow.*, 15(11), 2022.

[Sun *et al.*, 2025] Jiarui Sun, Chin-Chia Michael Yeh, Yujie Fan, Xin Dai, Xiran Fan, Zhimeng Jiang, Uday Singh Saini, Vivian Lai, Junpeng Wang, Huiyuan Chen, et al. Towards efficient large scale spatial-temporal time series forecasting via improved inverted transformers. *arXiv preprint arXiv:2503.10858*, 2025.

[Tolstikhin *et al.*, 2021] Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, pages 24261–24272, 2021.

[Wang *et al.*, 2016] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. Traffic speed prediction and congestion source exploration: A deep learning method. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 499–508. IEEE, 2016.

[Wang *et al.*, 2020] Jingyuan Wang, Zhen Peng, Xiaoda Wang, Chao Li, and Junjie Wu. Deep fuzzy cognitive maps for interpretable multivariate time series prediction. *IEEE transactions on fuzzy systems*, 2020.

[Wang *et al.*, 2021] Jingyuan Wang, Jiawei Jiang, Wenjun Jiang, Chao Li, and Wayne Xin Zhao. Libcity: An open library for traffic prediction. In *SIGSPATIAL/GIS*, 2021.

[Wang *et al.*, 2022] Jingyuan Wang, Jiahao Ji, Zhe Jiang, and Leilei Sun. Traffic flow prediction based on spatiotemporal potential energy fields. *IEEE Transactions on Knowledge and Data Engineering*, 35(9), 2022.

[Wang *et al.*, 2023] Jingyuan Wang, Chen Yang, Xiaohan Jiang, and Junjie Wu. When: A wavelet-dtw hybrid attention network for heterogeneous time series analysis. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 2023.

[Wang *et al.*, 2024a] Binwu Wang, Pengkun Wang, Zhengyang Zhou, Zhe Zhao, Wei Xu, and Yang Wang. Make bricks with a little straw: large-scale spatio-temporal graph learning with restricted gpu-memory capacity. In *33rd IJCAI*, 2024.

[Wang *et al.*, 2024b] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. In *ICLR*, 2024.

[Wang *et al.*, 2024c] Zhenghong Wang, Yi Wang, Furong Jia, Fan Zhang, Nikita Klimenko, Leye Wang, Zhengbing He, Zhou Huang, and Yu Liu. Spatiotemporal fusion transformer for large-scale traffic forecasting. *Information Fusion*, 107:102293, 2024.

[Wang *et al.*, 2025a] Dongjing Wang, Gangming Guo, Tianpei Ouyang, Dongjin Yu, Haiping Zhang, Bao Li, Rong Jiang, Guandong Xu, and Shuiguang Deng. A lightweight spatio-temporal neural network with sampling-based time series decomposition for traffic forecasting. *TITS*, 2025.

[Wang *et al.*, 2025b] Jingyuan Wang, Yujing Lin, and Yudong Li. Gtg: Generalizable trajectory generation model for urban mobility. In *AAAI*, 2025.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, 2019.

[Yeh *et al.*, 2024] Chin-Chia Michael Yeh, Yujie Fan, Xin Dai, Uday Singh Saini, Vivian Lai, Prince Osei Aboagye, Junpeng Wang, Huiyuan Chen, Yan Zheng, Zhongfang Zhuang, Liang Wang, and Wei Zhang. Rpmixer: Shaking up time series forecasting with random projections for large spatial-temporal data. In *KDD*. ACM, 2024.

[Yi *et al.*, 2023] Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Ning An, Defu Lian, Longbing Cao, and Zhendong Niu. Frequency-domain mlps are more effective learners in time series forecasting. In *NeurIPS*, 2023.

[You *et al.*, 2020] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. L2-GCN: layer-wise and learned efficient training of graph convolutional networks. In *CVPR*. Computer Vision Foundation / IEEE, 2020.

[Yu *et al.*,] Xie Yu, Jingyuan Wang, Yifan Yang, Qian Huang, and Ke Qu. BIGCity: A Universal Spatiotemporal Model for Unified Trajectory and Traffic State Data Analysis . In *41st ICDE*. IEEE Computer Society, May.

[Yu *et al.*, 2018] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*. ijcai.org, 2018.

[Zhang *et al.*, 2023] Zijian Zhang, Ze Huang, Zhiwei Hu, Xiangyu Zhao, Wanyu Wang, Zitao Liu, Junbo Zhang, S. Joe Qin, and Hongwei Zhao. MLPST: MLP is all you need for spatio-temporal prediction. In *CIKM*, pages 3381–3390. ACM, 2023.

A Appendix

A.1 Implementation Details

Baselines. We compare HSTMixer with the following 19 SOTA methods from two categories: **(1)** Previous SOTAs contains ASTGCN[Guo *et al.*, 2019], DCRNN[Li *et al.*, 2018], STGCN[Yu *et al.*, 2018], GWNET[Wu *et al.*, 2019], STGODE[Fang *et al.*, 2021], DSTAGNN[Lan *et al.*, 2022], D^{2}STGNN[Shao *et al.*, 2022b], HIEST[Ma *et al.*, 2023], PDFormer[Jiang *et al.*, 2023a], DGCRN[Li *et al.*, 2023] and TESTAM[Lee and Ko, 2024]. **(2)** Scalable Models consist of STID[Shao *et al.*, 2022a], FreTS[Yi *et al.*, 2023], TSMixer[Chen *et al.*, 2023], TimeMixer[Wang *et al.*, 2024b], RPMixer[Yeh *et al.*, 2024], EiFormer[Sun *et al.*, 2025], LSTNN[Wang *et al.*, 2025a] and GSNet[Kong *et al.*, 2025]

Hyperparameter	Values
lr	5e-4, 1e-3, 5e-3
d	32, 64, 128
h	64, 128, 256
p	2, 3, 4
L	2, 3, 4
K	1, 2, 3
S_k	16, 32, 64, 128, 256
M_k	2, 4, 8, 16, 32

Table 3: Hyperparameter search values.

Hyperparameter Search. We employ a grid search strategy to identify the optimal hyperparameter configurations for our HSTMixer in Table. 3. For the general settings, we use an initial learning rate of 1e-3. The feature embedding dimension d is fixed at 64, while the intermediate feature dimension h is set to 128. The temporal window size p for the temporal aggregation mixer is set to 2, and the model architecture consists of $L = 4$ stacked ST mixing blocks. For the dataset-specific settings, we tune the parameters governing the spatial cascade mixer. Specifically, the number of region scales K is searched within the set 1, 2, 3. The number of aggregated regions S_k for each layer is searched from 16, 32, 64, 128, 256, and the size of the base weights M_k is explored over 2, 4, 8, 16, 32. The final optimal values selected for each dataset are summarized in Table. 4.

Dataset	K	S_k	M_k
SD	1	[128]	[32]
GBA	2	[256, 32]	[32, 4]
GLA	2	[128, 16]	[32, 2]
CA	2	[256, 32]	[32, 4]

Table 4: Optimal dataset-specific hyperparameter settings.

A.2 Supplementary experiments

Efficiency Analysis. We further report the training and inference costs on the GBA dataset in Table. 5. Compared to traditional GNN-based and transformer-based models, our method achieves a dramatic reduction in both training time and inference latency, which validates its excellent scalability for large-scale scenarios. Against other scalable models, HSTMixer's time consumption is highly competitive, positioning it in the top-tier of efficiency. While its memory usage is moderately higher than some models, it remains well within practical limits. Crucially, the entire training process can be comfortably accommodated on a single NVIDIA RTX A6000 48GB GPU. Overall, HSTMixer effectively balances SOTA performance with the linear complexity required for large-scale deployment.

		Training			Infer	
	bs	s/epoch	time(h)	Mem(MiB)	time(s)	Mem
FreTS	64	101	11.22	21300	13.01	13076
TSMixer	64	397	2.32	12878	27.83	1296
RPMixer	64	256	2.13	10342	38.49	2886
GSNet	64	124	3.44	7780	27.41	4278
LSTNN	64	412	11.44	16780	58.30	4918
EiFormer	64	598	7.48	39282	89.77	6426
DCRNN	64	2141	59.47	43084	293.04	4238
DSTAGNN	27	2066	45.33	47235	146.63	5100
D2STGNN	4	6594	146.53	47312	791.59	5948
DGCRN	12	3152	74.42	26521	660.09	2086
HSTMixer	64	358	4.47	32792	29.74	8914

Table 5: The training and inference costs on the GBA dataset.

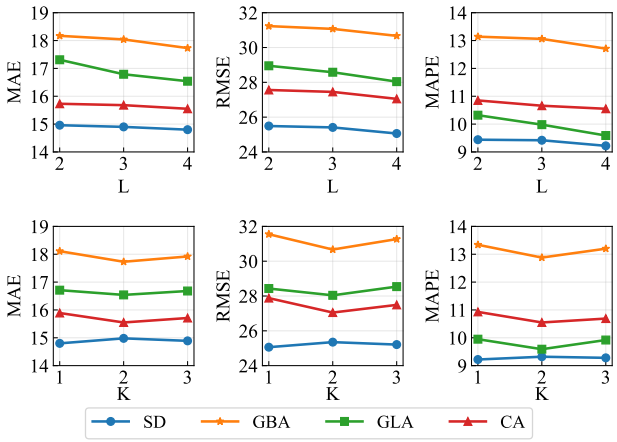


Figure 9: Sensitivity experiment results for L and K.

Parameter Sensitivity. We further analyze the impact of hierarchy depth, as shown in Figure. 9. For the temporal depth L, increasing the number of ST mixing blocks enriches the multi-scale temporal features and consistently improves performance. Given an input length of 12 and a window size of 2, L is limited to 4. For the spatial depth K, optimal performance is typically achieved at $K = 2$ on most datasets. Further increasing K expands the model size, which can complicate the training process and degrade performance. On the smaller SD dataset, however, $K = 1$ is optimal, likely because its smaller network does not require a deep spatial hierarchy.