Scalable Pre-Training of Compact Urban Spatio-Temporal Predictive Models on Large-Scale Multi-Domain Data

Jindong Han
Shandong University
hanjindong01@gmail.com

Hao Wang
The Hong Kong University of Science and Technology (Guangzhou)
hwang574@connect.hkust-gz.edu.cn

Hui Xiong
The Hong Kong University of Science and Technology (Guangzhou)
The Hong Kong University of Science and Technology
xionghui@ust.hk

Hao Liu*
The Hong Kong University of Science and Technology (Guangzhou)
The Hong Kong University of Science and Technology
liuh@ust.hk

ABSTRACT

Spatio-Temporal Prediction (STP) is crucial for various smart city applications, such as traffic management and resource allocation. However, training samples can be scarce in data-constrained scenarios, which often degrades the predictive capability of existing deep STP models. Although recent STP foundation models excel in few-shot and zero-shot learning through extensive pre-training on large-scale, multi-domain spatio-temporal data, they often rely on large parameter scale to achieve enhanced performance, resulting in high computational demands that hinder practical deployment. In response, we develop CompactST, an efficient, compact, and versatile pre-trained model for STP in data-scarce settings. Recognizing the complexities posed by large-scale, heterogeneous pre-training datasets, CompactST integrates three specialized components: (1) a mixture-of-normalizers module to address domain and spatial heterogeneity, (2) a multi-scale spatio-temporal mixer that captures diverse patterns from datasets with varying spatio-temporal resolutions, and (3) an adaptive dataset-oriented tuning module that transfers the handling of dataset-specific parameters from pre-training to fine-tuning stage. These tailored designs enable CompactST to maximize generalizability across diverse datasets while maintaining a compact model size (*i.e.*, only 300K parameters). To validate its effectiveness, we pre-train CompactST on a substantial corpus of public spatio-temporal datasets spanning over 10 domains and encompassing 300 million data points. Extensive experimental results on ten real-world datasets demonstrate CompactST's significantly improved prediction accuracy and efficiency in data-scarce scenarios.

PVLDB Reference Format:
Jindong Han, Hao Wang, Hui Xiong, and Hao Liu. Scalable Pre-Training of Compact Urban Spatio-Temporal Predictive Models on Large-Scale Multi-Domain Data. PVLDB, 18(7): 2149 - 2158, 2025.
doi:10.14778/3734839.3734851

*Corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 7 ISSN 2150-8097.
doi:10.14778/3734839.3734851

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/usail-hkust/CompactST.

1 INTRODUCTION

With the rapid development of sensing and ubiquitous computing technologies, massive spatio-temporal data have been generated in urban spaces, capturing various aspects of human activities and environmental dynamics [65]. For instance, air pollutant concentrations are tracked over time by measurement stations deployed in cities [13], while urban traffic monitoring systems continuously record real-time vehicle speeds and flows within the road network [40]. Spatio-Temporal Prediction (STP) is the task of predicting future states of these urban phenomena based on historical data, which can assist in optimizing smart city services, improving resource management, and supporting sustainable urban development. Given its substantial practical value, STP has attracted great research interest [9, 40, 49, 50, 56].

Unlike standard time series data, spatio-temporal data are characterized by their strong correlations and dependencies along the spatial dimension [2, 14]. Consequently, the prevailing approach is to first organize the data into spatially structured formats (*e.g.*, grid or graph formats), and then model them using hybrid deep learning models that combine Convolutional Neural Networks (CNNs) [55] or Graph Neural Networks (GNNs) [14, 40] with various sequence models, such as Transformer [45]. While effective, most existing models rely on extensive in-domain data collected from a specific spatial region to achieve decent performance, which fail to generalize to unfamiliar urban contexts with limited data availability [16, 17], *e.g.*, regions with newly deployed sensors, underdeveloped cities, or emerging urban services. This limitation poses a critical barrier to the real-world application of STP models.

To address this limitation, we are witnessing rapid progress of developing general-purpose STP models that can be transferred across diverse data-scarce scenarios. Recent studies either adapt pre-trained language models [23, 25, 28] or directly pre-train large Transformer-based foundation models [22, 57] to learn common patterns on numerous spatial-temporal datasets from multiple domains, demonstrating promising few-shot and zero-shot performance on unseen datasets. However, to accommodate diverse spatio-temporal

patterns, the parameter scale of these models has expanded from tens of millions [22, 57] to even billions [23, 25, 28], leading to significant deployment costs, particularly in resource-constrained industrial environments, *e.g.*, edge devices. Thus, it is indicative to minimize the number of parameters and develop compact yet capable pre-trained STP models.

Recently, several multi-layer perceptron (MLP) models [7, 29, 38, 61] have been proposed for STP tasks. These models replace parameter-intensive self-attention mechanisms used in Transformers with lightweight MLP layers, which substantially reduces the parameter scale while still achieving comparable or even superior performance to state-of-the-art STP models. This motivates our exploration of scalable pre-training for compact MLP models on large-scale spatio-temporal data. However, training a small MLP model on extensive and diverse spatio-temporal datasets presents three data-level challenges: (1) **Heterogeneous data distributions**: Spatio-temporal datasets vary considerably in their underlying distributions, both within the same domain, due to regional differences in city infrastructure, geography, and socio-economic factors [22, 35], and across domains, resulting in significant *spatial heterogeneity* and *domain heterogeneity*. Small models with limited capacity struggle to memorize the complex and varied data distributions from different regions and domains. (2) **Diverse spatio-temporal patterns at varying scales**: Different datasets often exhibit divergent patterns at vastly distinct spatio-temporal scales. For instance, minute-level traffic flow data captures short-term fluctuations and localized propagation among nearby intersections, while daily recorded air quality dataset reflects broader spatial dispersion patterns and long-term temporal trends of pollutants. The model must be capable of modeling the intricate dependencies in diverse spatio-temporal resolutions. (3) **Downstream knowledge misalignment**: Given inherent variations across urban contexts, downstream target dataset may only partially share the spatio-temporal knowledge acquired during pre-training. As a result, how to effectively preserve relevant pre-trained knowledge while maintaining flexible adaptability to the unique characteristics of specific target dataset is also a critical challenge.

To tackle the above challenges, we develop CompactST, a small yet capable pre-trained model tailored for generalizable STP in data-scarce scenarios. Specifically, we first propose a mixture-of-normalizers module, which is symmetrically structured to first remove and then restore specific domain and spatial effects of input data instances. This allows the model to learn domain-invariant and spatially generalizable features during pre-training, thereby addressing the first challenge. After that, we design a multi-scale spatio-temporal mixer, which includes a series of alternately stacked lightweight MLP layers and parameter-free graph-shift operators to capture spatio-temporal patterns at various scale ranges. Moreover, we propose an adaptive dataset-oriented tuning module, including (1) a knowledge selection mechanism that selectively adapts the pre-trained knowledge to target dataset, and (2) a private spatial mixing block that further grasps unique spatial knowledge in the target dataset to augment the pre-trained model.

Our contributions are summarized as follows: (1) We propose CompactST, a lightweight pre-trained model with specialized design (*i.e.*, only 300K parameters), which for the first time demonstrates the potential of tiny models for general-purpose STP. (2) We develop a mixture-of-normalizers module and a multi-scale spatio-temporal mixer, overcoming the bottleneck of pre-training extremely small MLP models on massive spatio-temporal data. (3) We introduce an adaptive dataset-oriented tuning module to alleviate negative knowledge transfer and further enhance the usability of pre-trained model on real-world target datasets. (4) We conduct extensive experiments on a large-scale spatio-temporal data corpora consisting of 300 million data points. The results demonstrate the effectiveness and efficiency of CompactST against state-of-the-art baselines.

2 PRELIMINARIES

In this section, we first present some basic concepts and then formally define our problem. We summarize the notations used throughout the paper in Table 1.

Table 1: Notations and explanations.

Notation	Explanation
$\mathbf{X}_{1:T} \in \mathbb{R}^{N \times T}$	Observation matrix from N nodes over T time steps
$\mathbf{x}_i$	Observation sequence at node i
$\mathbf{A}$	Spatial adjacency matrix
τ	Prediction horizon length
μ, σ^2	Mean and variance of the input instance
μ_s, σ_s^2	Mean and variance of node-wise sequence
γ, β	Learnable scaling and bias parameters
N_s	Number of spatial normalizers
$G(\cdot), E_n(\cdot)$	Routing function and n-th spatial normalizer
$\mathbf{x}'_i \in \mathbb{R}^{P \times M}$	M patches of node i with patch length P
$\mathbf{W}, \mathbf{E}$	Learnable parameter matrices
$\mathbf{x}_i^d, \mathbf{x}_i^p, \mathbf{x}_i^f, \mathbf{x}_i^s$	Node representations after spatio-temporal mixing
$\mathbf{x}_{i,m}^{s,(l)}$	Representation of patch m at layer l
$\mathrm{AGG}(\cdot)$	Patch aggregation function
$\mathbf{x}_i^{(l)}$	Output of node i at layer l
$\mathbf{X}^{(L)}$	Final output after L layers
$\hat{\mathbf{x}}_{T+1:T+\tau}$	Predicted values over future τ steps

Spatio-temporal data. Spatio-temporal data is represented as a two-dimensional observation matrix $\mathbf{X}_{1:T} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N) \in \mathbb{R}^{N \times T}$, where $\mathbf{x}_i$ denotes the sequence of T observations (*e.g.*, traffic flow, air quality) at node i (*i.e.*, a sensor or region in urban space) and N is the number of nodes. The spatial relations between nodes can be represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ is a set of edges, and A denotes the adjacency matrix built from geographical distance [20].

Typically, there are two formats of spatio-temporal data: network-based and raster-based. Network-based spatio-temporal data [14] are collected from a set of geo-distributed sensors deployed across a specific urban area. In contrast, raster-based spatio-temporal data [60] divides the urban area into regular grids, with each grid containing a specific time-varying attribute.

Spatio-temporal prediction. For a specific dataset, given a sequence of past T observations $\mathbf{X}_{1:T}$ and the associated spatial graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, the goal of spatio-temporal prediction is to predict future states of all the nodes in the next τ time steps. This

can be expressed as follows:

$$\mathbf{X}_{T+1:T+\tau} = f_\theta(\mathbf{X}_{1:T}, \mathcal{G}) \tag{1}$$

where $\mathbf{X}_{T+1:T+\tau}$ is the predicted future values from time step $T+1$ to $T+\tau$, $f_\theta(\cdot)$ represents a mapping function parameterized by θ, *i.e.*, prediction model.

Multi-domain pre-training. Given a set of spatio-temporal datasets $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_C\}$, where each dataset $\mathcal{X}_i$ belongs to a specific source domain $\mathcal{D}_i$ (*e.g.*, taxi demand), our objective is to pre-train a STP model $f_\theta(\cdot)$ on $\mathcal{X}$, capable of encoding spatio-temporal knowledge that is generalizable to downstream dataset $\mathcal{X}_{target}$. Specifically, we focus on few-shot and zero-shot prediction scenarios, where the pre-trained model $f_\theta(\cdot)$ can adapt to $\mathcal{X}_{target}$ with only a limited number of training samples from $\mathcal{X}_{target}$ or even without additional training. Notably, the domain of target downstream dataset can be either seen or unseen during pre-training, *i.e.*, we may have $\mathcal{D}_{target} = \mathcal{D}_i$ for $1 \le i \le C$.

3 METHODOLOGY

Figure 1 illustrates an overview of CompactST. We will introduce each component of CompactST below.

3.1 Mixture-of-Normalizers

Existing studies [24, 39, 57] usually rely on statistics (*e.g.*, mean and standard deviation) of training data to normalize inputs. However, these statistics only capture distribution specific to a given dataset, lacking universal transferability across domains or spatial regions, especially in scenarios with limited or no historical data. To overcome this limitation, we draw inspiration from instance normalization [19, 44] and design the mixture-of-normalizers, which progressively eliminates domain- and location-specific effects from input instances, allowing the model to focus on learning domain-invariant and spatially generalizable features.

3.1.1 ***Domain normalizer.*** Given an input instance $\mathbf{X}_{1:T} \in \mathbb{R}^{N\times T}$, where N is the number of nodes and T denotes the length of look-back window, we omit the subscripts in $\mathbf{X}_{1:T}$ for simplicity, and then calculate the mean and variance of each input instance $\mathbf{X}$:

$$\mu = \frac{1}{NT}\sum_{i=1}^{N}\sum_{j=1}^{T}\mathbf{X}_{i,j}, \quad \sigma^2 = \frac{1}{NT}\sum_{i=1}^{N}\sum_{j=1}^{T}(\mathbf{X}_{i,j}-\mu)^2, \tag{2}$$

where $\mathbf{X}_{i,j}$ is the observation at node i and time step j, μ and σ^2 denote the mean and variance of input instance, respectively. Intuitively, the computed statistics of each instance inherently carry the properties (*e.g.*, numerical scales) of its associated domains. Therefore, we can remove domain-specific information by normalizing the original input instance $\mathbf{X}$ as follows:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}-\mu}{\sigma+\epsilon}, \tag{3}$$

where ϵ is a small constant added to ensure numerical stability. This normalization step produces a domain-agnostic instance, helping prevent the model from overfitting to the specific characteristics of particular domains.

3.1.2 ***Spatial normalizers.*** Although the domain normalizer can mitigate the domain shifts among instances, heterogeneity still exists within the same instance due to significant variations in data distribution across space. This spatial heterogeneity can introduce location-specific biases into the model, making it difficult to establish stable and universally applicable spatio-temporal correlations. Thus, we develop a series of spatial normalizers to further enhance the model's generalizability across different spatial regions.

Unlike domain normalizer, each spatial normalizer applies normalization operator on a per-node basis instead of the entire input instance, defined as

$$\mu_s = \frac{1}{T}\sum_{j=1}^{T}\mathbf{x}_{i,j}, \quad \sigma_s^2 = \frac{1}{T}\sum_{j=1}^{T}(\mathbf{x}_{i,j}-\mu_s)^2, \tag{4}$$

where $\mathbf{x}_i$ is the i-th row vector of the domain normalized input $\hat{\mathbf{X}}$. After that, we derive the spatially normalized input for each node i via the following equation:

$$\tilde{\mathbf{x}}_i = \gamma \cdot \frac{\mathbf{x}_i - \mu_s}{\sigma_s+\epsilon} + \beta, \tag{5}$$

where γ and β are learnable parameters. Specifically, γ and β allows for node-wise scaling and shifting of the normalized features, which can retain critical temporal dynamics.

However, simply normalizing the features may reduce the spatial distinguishability of each node, which has been proven very useful for model prediction in previous works [5, 38]. Hence, rather than using a unified normalizer for all nodes, we introduce the Mixture of Normalization Layer (MNL), which consists of multiple spatial normalizers. Concretely, an MNL layer employs learnable router to select a specific spatial normalizer from a shared normalizer pool for each node. Formally, an MNL layer is defined as

$$\tilde{\mathbf{x}}_i = \sum_{n=1}^{N_s} G_n(\mathbf{x}_i)\cdot E_n(\mathbf{x}_i), \tag{6}$$

where N_s denotes the number of normalizers, $G(\cdot)$ is a sparse router and $G_n(\cdot)$ is the n-th element of the output vector from $G(\cdot)$, indicating the importance of the n-th normalizer $E_n(\cdot)$. We instantiate $G(\cdot)$ with a simple MLP layer, denoted as $G(\mathbf{x}_i) = \text{Top-1}(\text{MLP}(\mathbf{x}_i))$, where Top-1$(\cdot)$ operator sparsely activate one normalizer with the largest entry in the output of MLP. By doing so, we can adaptively select the most suitable normalizer for each node, preserving important node-specific dynamics even after normalization. Finally, we denote the output of MNL as $\tilde{\mathbf{X}}$, where the i-th row vector of $\tilde{\mathbf{X}}$ is $\tilde{\mathbf{x}}_i$. The data instance will be denormalized to restore spatial and domain information at the output of prediction model, which is detailed in Section 3.2.2.

3.2 Multi-Scale Spatio-Temporal Mixer

In this section, we elaborate on how to capture spatio-temporal dynamics at varying scales via a multi-scale spatio-temporal mixer, which can significantly enhance model's adaptability to diverse pre-training datasets.

3.2.1 ***Multi-scale spatio-temporal mixing.*** Given the normalized input $\tilde{\mathbf{x}}_i \in \mathbb{R}^T$ for node i, we first divide $\tilde{\mathbf{x}}_i$ into non-overlapping patches [34] along temporal dimension, denoted as $\mathbf{x}'_i \in \mathbb{R}^{P\times M}$, where P is patch length and M is the number of patches. By using

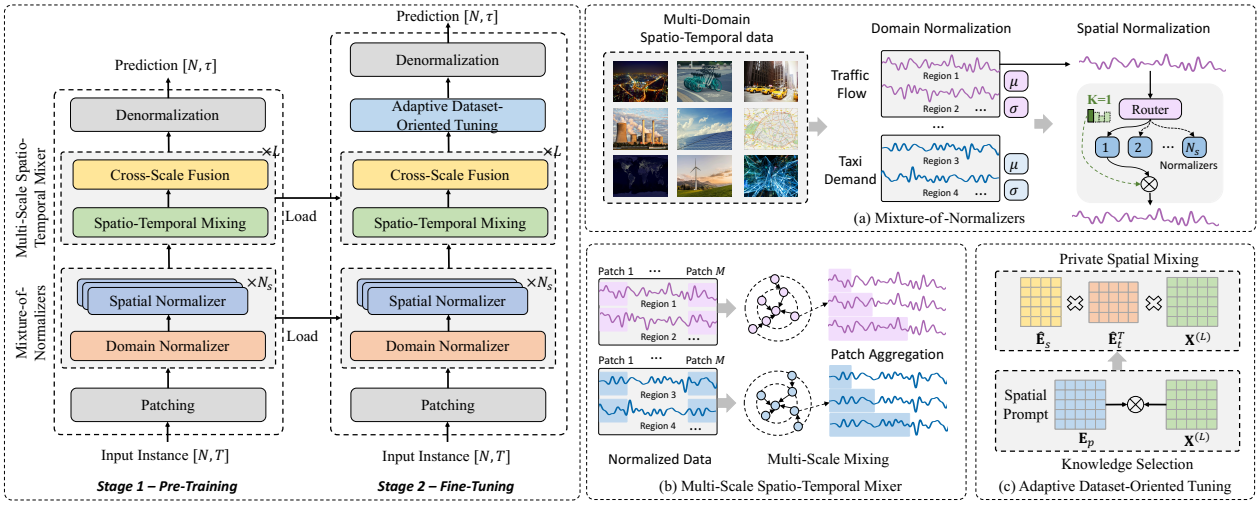


Figure 1: The framework of CompactST.

the patching strategy, we can significantly reduce the number of parameters and memory usage for subsequent processing. Next, we transform the patched input $\mathbf{x}'_i$ into an H-dimension latent space and inject positional embeddings to preserve the temporal order of each patch, defined as

$$\mathbf{x}_i^d = \mathbf{W}_d \mathbf{x}'_i + \mathbf{E}^{pos}, \tag{7}$$

where $\mathbf{W}_d \in \mathbb{R}^{H \times M}$ represents a learnable projection matrix, and $\mathbf{E}_{pos}$ is the sinusoidal positional embedding [45].

Previous studies suggests that replacing computationally intensive self-attention mechanism with lightweight MLP mixing layer can still yield strong performance in various tasks [7, 43]. Therefore, to reduce model complexity, we design the spatio-temporal mixing layer, which contains three types of operators: inter-patch mixing, intra-patch mixing, and spatial mixing. Specifically, inter-patch mixing processes each dimension separately by mixing temporal information across different patches:

$$\mathbf{x}_i^p = \mathbf{x}_i^d + \sigma(\mathbf{x}_i^d \mathbf{W}_{p1}) \mathbf{W}_{p2}, \tag{8}$$

where $\mathbf{W}_{p1} \in \mathbb{R}^{M \times M}$ and $\mathbf{W}_{p2} \in \mathbb{R}^{M \times M}$ are learnable inter-patch mixing matrices, $\sigma(\cdot)$ is a non-linear activation function, and a residual connection is added to facilitate information flow across layers. In contrast, intra-patch mixing operates on the output of the inter-patch mixing and mixes information along the feature dimension within each patch:

$$\mathbf{x}_i^f = \mathbf{x}_i^p + \mathbf{W}_{f2} \sigma(\mathbf{W}_{f1} \mathbf{x}_i^p), \tag{9}$$

where $\mathbf{W}_{f1} \in \mathbb{R}^{H \times H}$ and $\mathbf{W}_{f2} \in \mathbb{R}^{H \times H}$ are learnable intra-patch mixing matrices. In addition, we construct a spatial adjacency matrix $\mathbf{A}$ for each dataset, which encodes the prior spatial relationships among nodes [14]. Building upon the adjacency matrix $\mathbf{A}$, we design a new spatial mixing operator, which leverages a graph-shift operator that mixes information from different nodes at a particular time (*i.e.*, patch), defined as

$$\mathbf{x}_i^s = \mathbf{x}_i^f + \sigma(\sum_{j=1}^{N} \mathbf{A}_{i,j} \mathbf{x}_j^f), \tag{10}$$

Since $\mathbf{A}$ is a highly sparse matrix with fixed values, the spatial mixing operator is parameter-free and extremely fast in practice.

To learn spatio-temporal patterns at varied scales, we further introduce a layer-wise patch aggregator, which gradually reduces the number of patches by merging adjacent patches as the layer increases. Formally, suppose we stack L spatio-temporal mixing layers, the output of the l-th layer can be rewritten as a series of patches $\{\mathbf{x}_{i,1}^{s,(l)}, \mathbf{x}_{i,2}^{s,(l)}, ..., \mathbf{x}_{i,M^{(l)}}^{s,(l)}\}$, where $M^{(l)}$ is the number of patches in layer l. The patch aggregator for the subsequent layer $l+1$ is defined as follows:

$$\mathbf{x}_{i,m}^{s,(l+1)} = \text{AGG}([\mathbf{x}_{i,2m-1}^{s,(l)}, \mathbf{x}_{i,2m}^{s,(l)}]), \tag{11}$$

where $\text{AGG}(\cdot)$ is an aggregation function that combines two consecutive patches $\mathbf{x}_{i,2m-1}^{s,(l)}$ and $\mathbf{x}_{i,2m}^{s,(l)}$ from layer l to form a new patch $\mathbf{x}_{i,m}^{s,(l+1)}$ in layer $l+1$. Thus, the number of patches are reduced from $M^{(l)}$ to $M^{(l+1)} = M^{(l)}/2$. Here we implement $\text{AGG}(\cdot)$ with a trainable linear projection. By interleaving the spatial mixing operator with patch aggregator, we can progressively decreases the temporal granularity while simultaneously capturing dependencies across a wider spatial range.

3.2.2 ***Cross-scale fusion.*** In practice, we stack L spatio-temporal mixing layers, with each layer operating at a specific scale range. For the l-th layer, we apply mean pooling operator to aggregate its output $\mathbf{x}_i^{s,(l)} \in \mathbb{R}^{H \times M^{(l)}}$ along the patch dimension, resulting in $\mathbf{x}_i^{(l)} \in \mathbb{R}^H$. Then, we adopt a layer-wise prediction module to fuse the derived $\mathbf{x}_i^{(l)}$ at different layers, defined as

$$\hat{\mathbf{x}}_{T+1:T+\tau} = \sum_{l=1}^{L} \mathbf{x}_i^{(l)} \mathbf{W}_l, \tag{12}$$

where $\mathbf{W}_l \in \mathbb{R}^{H\times\tau}$ maps $\mathbf{x}_i^{(l)}$ to the future prediction. Intuitively, some datasets may benefit more from the fine-scale predictions of earlier layers, while others may rely on the broader patterns captured by deeper layers. This strategy enables the model to effectively integrate spatio-temporal patterns at multiple scales, thereby enhancing its capacity to generalize across different spatial-temporal contexts. Finally, we denormalize the output of the layer-wise prediction module using the statistics from Equation 3 and 5 to restore the original spatial and domain information:

$$\hat{\mathbf{x}}_{T+1:T+\tau} \leftarrow \sigma_s \cdot \frac{\hat{\mathbf{x}}_{T+1:T+\tau} - \beta}{\gamma} + \mu_s, \tag{13}$$

$$\hat{\mathbf{x}}_{T+1:T+\tau} \leftarrow \sigma \cdot \hat{\mathbf{x}}_{T+1:T+\tau} + \mu. \tag{14}$$

Our model is pre-trained by optimizing the following objective:

$$\mathcal{L} = \frac{1}{N\tau} \sum_{i=1}^{i=N} \sum_{j=T+1}^{j=T+\tau} |\mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}|. \tag{15}$$

3.3 Adaptive Dataset-Oriented Tuning

Finally, we present the adaptive dataset-oriented tuning module, which includes (1) a knowledge selection mechanism that adaptively selects beneficial knowledge from the pre-trained representations, and (2) a private spatial mixing block that further captures unique spatial dependencies in target dataset.

3.3.1 ***Knowledge selection.*** In multi-domain pre-training, the model acquires extensive spatio-temporal knowledge from heterogeneous datasets. However, the discrepancy between pre-training datasets and target dataset may lead to the negative transfer of irrelevant or even harmful knowledge. To address this issue, we introduce a learnable spatial prompt to adaptively select useful pre-trained knowledge for the target dataset:

$$\mathbf{X}^{(L)} \leftarrow \mathbf{X}^{(L)} \odot \mathbf{E}_p, \tag{16}$$

where the i-row vector of $\mathbf{X}^{(L)}$ is $\mathbf{x}_i^{(L)}$, $\mathbf{E}_p \in \mathbb{R}^{N\times H}$ is a learnable prompt, and $\odot$ denotes the element-wise product. The prompt $\mathbf{E}_p$ is dataset-specific and is learned during the fine-tuning process on the target dataset, ensuring that only useful knowledge is retained while negative information is automatically discarded.

3.3.2 ***Private spatial mixing.*** The target dataset might exhibit implicit spatial dependencies that are not encoded by the prior adjacency matrix. For example, in traffic prediction, regions that are spatially distant may share similar traffic patterns due to environmental or functional similarities [14, 40]. These dependencies are difficult to learn during pre-training, as they are highly context-dependent and often unique to the target dataset.

To capture such context-aware spatial dependencies, we introduce a private spatial mixing block, defined as follows:

$$\mathbf{X}^{(L)} \leftarrow \mathbf{W}_s \mathbf{X}^{(L)}, \tag{17}$$

where $\mathbf{W}_s \in \mathbb{R}^{N\times N}$ is a learnable matrix that mixes information across different nodes. One scalability concern is that the spatial mixing operator has quadratic complexity $O(N^2)$, making it computationally infeasible for large spatio-temporal systems with numerous nodes. In response, we develop an optional lightweight operator based on linear attention [18]. Concretely, we first define two trainable context embedding matrices $\mathbf{E}_s \in \mathbb{R}^{N\times H}$ and $\mathbf{E}_t \in \mathbb{R}^{N\times H}$, and the lightweight spatial mixing operator can be defined as follows:

$$\mathbf{X}^{(L)} \leftarrow \hat{\mathbf{E}_s}(\hat{\mathbf{E}_t}^\top \mathbf{X}^{(L)}), \tag{18}$$

where $\hat{\mathbf{E}_s} = \sigma(\mathbf{E}_s)/\|\sigma(\mathbf{E}_s)\|_2$ and $\hat{\mathbf{E}_t} = \sigma(\mathbf{E}_t)/\|\sigma(\mathbf{E}_t)\|_2$, with $\|\cdot\|_2$ and $\sigma(\cdot)$ denoting a row-wise Euclidean norm function and a non-linear activation function (*e.g.*, ReLU), respectively. The brackets indicate the order of matrix multiplication. Using this operator, we avoid explicitly materializing the quadratic spatial mixing matrix $\mathbf{W}_s$, thus reducing the computational and storage complexity to a linear scale. During fine-tuning process, the derived $\mathbf{X}^{(L)}$ can be utilized to generate final prediction by following equation 12.

3.4 Complexity Analysis

In this section, we analyze the complexity of each proposed component. For the mixture-of-normalizers module, the time and space complexities of domain normalizer are both $O(NT)$, while the complexities for spatial normalizers are $O(NT + ND)$ and $O(N + NT + N_s D)$, where D is the dimension of MLP used to determine the importance of each normalizer. In layer l of the spatio-temporal mixer, the inter-patch mixing operates on $M^{(l)}$ patches, resulting in a time complexity of $O({M^{(l)}}^2)$. Given $M^{(l)} = M/2^{l-1}$, where M is the number of patches in the first layer, the time complexity across L layers becomes $O(M^2 \sum_{l=1}^{L} 1/4^{l-1})$. Since $\sum_{l=1}^{L} 1/4^{l-1}$ converges, we simplify the complexity to $O(M^2)$. Similarly, the overall complexity of intra-patch mixing across L layers is $O(H^2 M)$. The spatial mixing requires a time complexity of $O(|\mathcal{E}|)$, where $\mathcal{E}$ denotes the number of sparsely connected edges in the spatial graph, which is more efficient than dense matrix multiplication. The patch aggregator reduces the number of patches by half with each layer, leading to a logarithmic complexity of $O(H \log M)$. Hence, the overall time complexity of spatio-temporal mixer is $O(M^2 + H^2 M + \mathcal{E} + H \log M)$, while the space complexity is $O(M^2 + H^2 + \mathcal{E})$. In adaptive dataset-oriented tuning, the time and space complexities of knowledge selection are both $O(NH)$. For private spatial mixing, the naive implementation has quadratic complexity in terms of N, while the lightweight version reduces the complexity from $O(N^2 H)$ to $O(NH^2)$. Overall, since the number of patches M is relatively small and can be regarded as a constant, the model complexity scales linearly with N, making it efficient for large-scale datasets.

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Dataset description. We conduct experiments on a large spatio-temporal data corpora encompassing 34 individual datasets, which can be classified into 10 domains based on measured variables. In particular, these datasets are collected from various regions in China and the United States, including cities such as Beijing, Shanghai, Shenzhen, Hangzhou, Chengdu, New York City, Chicago, and Washington, among others. For each dataset, we construct the corresponding graph adjacency matrix based on the distance between different spatial nodes [20]. Our pre-training utilizes a subset of 24 datasets, covering a total of 9,718 regions and 299,637,248 data points. We leave the remaining datasets for model evaluation. The statistics of each dataset are summarized in Table 2 and 3.

Table 2: Statistical information of the pre-training datasets, which are aggregated based on the measured variables. For sampling rate: T=minute, H=hour. We provide detailed description of each dataset in supplementary material.

Domain	Metro Flow	Bike-Sharing	Taxi Demand	Vehicle Speed	Vehicle Flow	Air Quality	Total
# of Regions	644	398	1,301	5,775	1,718	280	9,718
# of Records	7,141,216	3,853,440	74,561,556	126,515,596	85,112,640	2,452,800	299,637,248
Sampling Rate	10T, 15T, 30T	30T, H	15T, 30T, H	5T, 10T, 15T, 30T, H	5T, 10T, 15T, 30T, H	H	5T, 10T, 15T, 30T, H
Source	[27, 59]	[46]	[46, 59, 63]	[3, 20]	[11, 41]	[13]	-

Table 3: Statistical information of the downstream evaluation datasets. Notice that, the evaluation datasets differ from the pre-training datasets in terms of *spatial regions, sampling rate, or domains*. For example, the pre-training datasets of taxi demand are collected from Beijing and Shenzhen, China as well as Chicago, USA, whereas NYC-Taxi dataset used in evaluation is collected from New York City, USA.

Dataset	NYC-Bike	NYC-Taxi	DiDi-CD	DiDi-SZ	SD	Beijing-AQI	NYC-PVm	NYC-PVh	DC-Wind	DC-WP
Domain	Bike-Sharing	Taxi Demand	Traffic Index	Traffic Index	Vehicle Flow	Air Quality	Solar Energy	Solar Energy	Wind Speed	Wind Power
# of Regions	200	200	524	627	716	35	129	129	759	759
# of Records	1,152,000	1,152,000	9,054,720	10,834,560	25,088,640	306,600	2,260,080	1,130,040	13,334,112	13,334,112
Sampling Rate	30T	30T	10T	10T	15T	H	15T	H	30T	30T
Source	[46]	[46]	[33]	[33]	[30]	[13]	[1]	[1]	[6]	[6]

4.1.2 Baselines. For few-shot prediction, we compare the proposed model with state-of-the-art baseline methods, including (1) heuristic approach: Historical Average (HA), (2) time series forecasting models: Informer [67], PatchTST [34], MTSMixer [21], iTransformer [31], and TimeMixer [47], (3) Spatio-Temporal GNN models: STGCN [53] and GWNET [54], and (4) advanced non-GNN predictive models: ST-Norm [5], STID [38], STAEformer [26], and EasyST [42]. For zero-shot prediction, we compare our model with recently released large foundation models: TimesFM [4], Timer [32], and OpenCity [22]. We implement all deep learning baselines using a popular STP library BasicTS [37] or their official codes.

4.1.3 Implementation details. Our model and all deep learning baselines are implemented with Pytorch. Model pre-training is conducted on a Linux server with 2 NVIDIA A40 GPUs. Specifically, we set the look-back time window T and prediction horizons τ to 96 and 48, respectively. We set the number of spatial normalizers to 8. For multi-scale spatio-temporal mixer, we set the number of patches and the patch length in the first layer to 16 and 6. The hidden dimension H and the number of layers in spatio-temporal mixer is fixed to 64 and 5. The dropout rate is set to 0.1. The model is pre-trained with 30 epochs using the Adam optimizer, with a weight decay of 0.0003, a learning rate of 0.001, and a gradient clip of 5. The batch size of pre-training is set to 32, and we apply the learning rate decay at epochs 15 and 25 with a decay rate of 0.1. For downstream fine-tuning, we set the dimension of context embedding matrices D to 32. We split all evaluation datasets into training, validation, and test set by using the ratio of 7:1:2, and we only use 5% and 10% of the training data for few-shot prediction.

4.2 Overall Results

4.2.1 Few-shot prediction. In real-world applications, practitioners often have a limited amount of target data available for training a model. Thus, we evaluate few-shot performance on both seen (*i.e.*, NYC-Bike, NYC-Taxi, SD, and Beijing-AQI) and unseen (*i.e.*, DiDi-CD, DiDi-SZ, NYC-PVm, NYC-PVh) domains during pre-training. Table 4 report the few-shot results using 5% of training data. Overall, CompactST outperforms the best-performing baseline on seen domains by 2.8% ~32.9% and unseen domains by 1.3% ~3.6%, demonstrating the effectiveness of pre-trained weights in data-scarce scenarios. Moreover, we can make the following observations: (1) Informer performs even worse than HA on some datasets, likely due to its sensitivity to data scarcity, which leads to severe overfitting issues. (2) With a long look-back window of 96, PatchTST achieves strong results on most datasets due to its robust long-sequence modeling capability. However, its inability to capture spatial dependencies limits its performance on datasets rich in neighborhood information, such as DiDi-CD and DiDi-SZ, where GNN-based models like STGCN and GWNET perform better. (3) The performance of STID and STAEformer lags behind time series or GNN-based models, potentially due to difficulties in learning reliable embeddings with limited training data. Notably, STAEformer surpasses STID on most datasets by leveraging the Transformer architecture to capture richer spatial information, demonstrating better capability in modeling spatio-temporal dependencies.

4.2.2 Zero-shot prediction. To evaluate the zero-shot performance of CompactST, we compare it to three representative foundation models, TimesFM [4], Timer [32], and OpenCity [22], using five datasets that these models had not encountered during pre-training. The underlying distributions of these datasets deviate substantially from those in pre-training data, which is suitable for evaluating the model's robustness under spatio-temporal distribution shift without fine-tuning. As shown in Figure 2, CompactST demonstrates competitive and often superior performance compared to TimesFM, Timer, and the three versions of OpenCity (mini, base, and plus), despite having a smaller set of parameters. This further validates the effectiveness of the proposed mixture-of-normalizers and multi-scale modeling techniques, which enable the model to learn generalizable patterns from extensive and diverse spatio-temporal data.

Table 4: Few-shot prediction results with 5% of training data. The historical look-back window T and forecast length τ are set to 96 and 48, respectively. We report the average MAE and RMSE over all forecast lengths.

Dataset	NYC-Bike				NYC-Taxi				SD		Bejing-AQI		DiDi-CD		DiDi-SZ		NYC-PVm		NYC-PVh	
	Inflow		Outflow		Inflow		Outflow													
Metric	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
HA	3.51	10.54	3.54	10.64	20.33	60.42	21.20	58.69	118.07	148.35	41.81	57.15	6.38	8.20	4.85	6.89	4.77	7.86	4.71	7.69
Informer	3.71	11.56	3.73	11.48	21.89	61.88	22.33	59.85	41.27	72.17	43.76	62.19	3.28	4.91	2.92	4.55	2.88	6.85	2.68	6.09
iTransformer	2.41	7.96	2.49	8.03	15.09	47.31	15.65	45.66	32.51	56.01	38.99	55.42	3.24	4.84	2.82	4.66	2.68	5.84	2.41	5.67
PatchTST	2.47	8.06	2.36	7.82	12.33	39.28	12.59	38.92	32.79	60.51	35.26	55.19	3.44	4.96	2.84	4.41	2.56	5.78	2.44	5.73
STGCN	2.66	8.78	2.61	8.66	16.89	54.63	17.14	51.72	32.99	52.74	39.93	56.38	2.97	4.58	2.61	4.11	2.43	5.55	2.36	5.40
GWNET	2.38	6.75	2.28	8.58	14.55	46.27	15.31	47.12	34.67	53.42	38.91	54.19	3.14	4.66	2.73	4.46	3.41	8.98	3.39	8.67
ST-Norm	2.74	10.31	2.75	10.63	16.47	53.21	17.08	51.99	35.04	61.72	36.81	56.52	3.12	4.76	2.69	4.29	2.60	6.48	2.51	5.97
STID	2.49	8.34	2.42	8.16	13.22	41.76	12.81	38.64	35.45	67.83	34.53	52.82	3.23	4.92	2.81	4.35	3.07	6.37	2.84	6.50
STAEformer	2.41	8.18	2.35	8.42	14.57	43.38	15.03	44.41	30.69	47.45	37.67	53.18	3.11	4.62	2.81	4.43	2.54	6.38	2.55	5.74
MTSMixer	2.43	8.22	2.51	8.45	15.18	46.22	15.46	45.44	32.56	58.03	39.54	58.45	3.24	4.67	2.79	4.34	2.60	5.85	2.83	5.92
TimeMixer	2.24	7.82	2.20	7.89	16.51	50.59	15.65	44.97	28.95	56.12	38.43	54.32	3.15	4.60	2.80	4.36	2.45	5.63	2.35	5.39
EasyST	2.37	8.30	2.33	8.21	14.16	45.28	13.03	39.84	32.10	61.81	33.71	56.46	3.25	4.90	2.73	4.20	2.55	6.19	2.44	5.51
CompactST	**1.79**	**6.44**	**1.74**	**6.33**	**9.58**	**32.31**	**9.73**	**31.12**	**25.30**	**43.47**	**33.31**	**51.49**	**2.92**	**4.30**	**2.52**	**4.01**	**2.43**	**5.34**	**2.32**	**5.24**

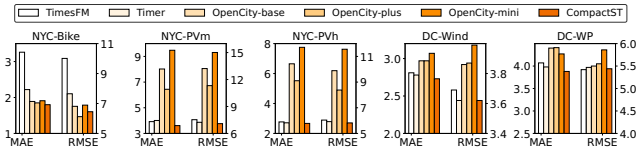


Figure 2: Zero-shot Prediction.

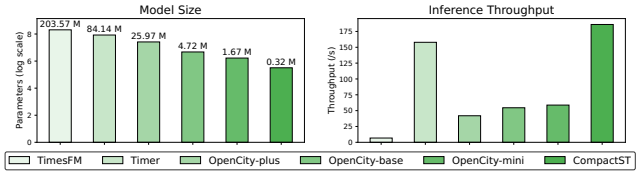
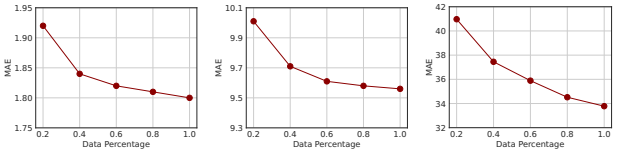


Figure 3: Model size and inference throughput.

4.3 Scalability Analysis

4.3.1 Model size and inference cost. Figure 3 shows the size and inference throughput of TimesFM, Timer, OpenCity, and CompactST, where throughput refers to the average number of batches the model can process per second. Compared to OpenCity-plus, our model achieves 4.4× higher inference acceleration, while reducing model size from 26M to 0.32M. In comparison to TimesFM, our model is 625× smaller and achieves 27.3× faster inference speed. These results confirm the potential of our model as a powerful, lightweight solution for real-world applications where unseen domains and data scarcity raise generalization concerns.

4.3.2 Scalability with data scale. In this section, we examine the scalability of CompactST with respect to data size. Specifically, we pre-train CompactST using varying amounts of data and evaluate its zero-shot generalization capability on the NYC-Bike, NYC-Taxi, and SD datasets. As shown in Figure 4, the zero-shot prediction error of CompactST consistently decreases as the pre-training data size increases. This steady improvement across datasets highlights the robustness of CompactST and its potential as an efficient solution capable of leveraging larger data scales to enhance zero-shot generalization performance. These observations indicate that even for a small model, its ability to learn from large-scale spatio-temporal data can be significantly enhanced through tailored design.

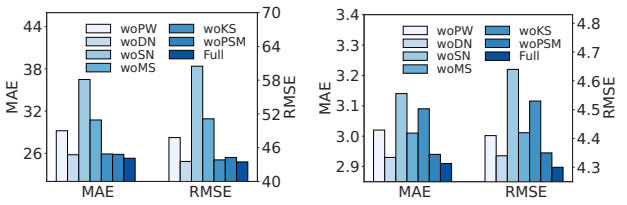


(a) NYC-Bike inflow. (b) NYC-Taxi inflow. (c) SD.

Figure 4: Scalability with data scale.

4.4 Ablation Study

In this section, we perform ablation studies on SD and DiDi-CD datasets to analyze the contribution of each component in the proposed model. Specifically, we evaluate five model variants: (1) *woPW*, which replaces the pre-trained model weights with randomly initialized weights; (2) *woDN*, which removes domain normalizer; (3) *woSN*, which excludes spatial normalizers; (4) *woMS*, which omits the patch aggregator and cross-scale fusion; (5) *woKS*, which eliminates knowledge selection: (6) *woPSM*, which removes private spatial mixing; (7) *Full*, which indicates the complete model. The results are presented in Figure 5. Firstly, we observe significant performance degradation when replacing pre-trained weights with randomly initialized weights, with the effect being more dramatic on SD. This is because the model was pre-trained on traffic flow datasets from other regions, enabling it to generalize well to SD. In contrast, DiDi-CD measures traffic index, a domain that the model did not encounter during pre-training, resulting in relatively smaller performance drop. Secondly, the domain normalizer, spatial normalizers, and multi-scale modeling module affect model performance to varying degrees. The removal of spatial normalizers has the most



(a) SD. (b) DiDi-CD.

Figure 5: Ablation study on SD and DiDi-CD datasets.

substantial impact, indicating that accounting for spatial heterogeneity is crucial in STP tasks. Thirdly, the knowledge selection mechanism plays an essential role in maintaining performance on DiDi-CD. The possible reason is that the traffic index domain is not involved during pre-training, resulting in limited shared knowledge between the pre-training data and the DiDi-CD dataset. Without knowledge selection, there is a higher risk of negative knowledge transfer when applied to this unseen domain.

5 RELATED WORKS

5.1 Spatio-Temporal Prediction

Spatio-Temporal Prediction (STP) has become a crucial component of various smart city services and has attracted significant research attention [40, 49–51]. Over the past decade, deep learning models have achieved remarkable success in STP tasks by simultaneously capturing intricate spatial and temporal dependencies. In early studies [55, 60], Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are frequently combined into hybrid neural architectures to process grid-based spatio-temporal data. Nevertheless, these methods fall short when learning from non-Euclidean data prevalent in urban systems, such as traffic flow in road networks [30]. To overcome this limitation, Spatio-Temporal Graph Neural Networks (STGNNs) [12, 14, 24, 39] have emerged as an effective approach, modeling complex spatio-temporal dependencies as a diffusion process over the system's underlying graph structure. For instance, GWNET [54] learns a graph adjacency matrix directly from data and combine Graph Neural Networks (GNNs) [52] with dilated CNNs for spatio-temporal correlation modeling. Another line of research, including STID [38] and STAEformer [26], utilizes identity embeddings to memorize stable spatio-temporal patterns, achieving state-of-the-art predictive performance. While effective, these approaches still require extensive dataset-specific training and a sufficient amount of in-domain data to build accurate predictive models, raising generalization concerns in ubiquitous real-world data-scarce scenarios. A few recent studies [22, 57] have started exploring the pre-training of a unified predictive model across different datasets. However, the effectiveness of pretraining is still limited by the inherent diversity of spatio-temporal data, which remains an open question in this burgeoning field.

5.2 Pre-trained Models for Time Series

In recent years, we have witnessed progressive breakthrough in time series learning models [36, 58, 62], such as PatchTST [34], TimeMixer [47], and MSD-Mixer [66]. Despite this progress, state-of-the-art models still experience substantial performance degradation when training data are scarce [32]. Therefore, it is crucial to improve the generalizability of time series learning models.

Pre-training has been proven highly effective in the fields of natural language processing [64]. However, extending this success to time series data remains challenging due to the limited availability and diversity of public time series datasets. Time series data differ semantically across domains (*e.g.*, healthcare, climate, and web traffic) and vary considerably in their fundamental characteristics (*e.g.*, frequencies, sequence length, and the number of variables) [8]. These unique properties complicate the development of pre-trained models that can generalize across different time series datasets. To address these issues, several approaches have been proposed recently, which fall into two classes: pre-trained language models and large time series models. The former directly reuse or fine-tune pre-trained language models for downstream time series analysis tasks [15, 68]. For example, Time-LLM [15] aligns the embedding space of large language models with time series data using a reprogramming strategy. More recently, significant progress has been made in building time series foundation models by pre-training on vast amounts of real-world and synthetic data, including Timer [32], Moirai [48], TimesFM [4], Moment [10], and TTM [8]. Among them, TTM is a lightweight pre-trained model with MLP architecture, demonstrating strong few-shot and zero-shot capabilities on unseen datasets. However, unlike the one-dimensional sequential structure of time series data, spatio-temporal data is inherently more complex, with entangled dependencies across both space and time. In this paper, we focus on capturing the spatio-temporal dependencies during the pre-training process.

6 CONCLUSION

In this paper, we introduce CompactST, an efficient, compact, and versatile pre-trained model designed to enhance few-shot and zero-shot STP in resource-constrained environments. By leveraging a combination of normalizers, a multi-scale spatio-temporal mixer, and an adaptive dataset-oriented tuning module, CompactST effectively captures generalizable spatio-temporal patterns across various spatial regions and domains while maintaining a restricted number of model parameters. Empirical evaluations on ten real-world datasets demonstrate that CompactST achieves notable gains in both performance and efficiency, particularly in data-scarce scenarios. However, while CompactST can handle datasets with varying numbers of spatial nodes, it is trained on specific look-back and prediction lengths, which limits its flexibility. In future work, we aim to extend CompactST with variable-length sequence modeling capabilities and deploy it into real-world STP systems.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (Grant No.2023YFF0725004), National Natural Science Foundation of China (Grant No.92370204), the Guangzhou Basic and Applied Basic Research Program under Grant No. 2024A04J3279, Education Bureau of Guangzhou Municipality, and CCF-DiDi GAIA Collaborative Research Funds.

REFERENCES

[1] Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. 2023. Scalable spatiotemporal graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 7218–7226.

[2] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. 2023. Graph deep learning for time series forecasting. *arXiv preprint arXiv:2310.15978* (2023).

[3] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. 2019. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems* 21, 11 (2019), 4883–4894.

[4] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. [n.d.]. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*.

[5] Jinliang Deng, Xiusi Chen, Renhe Jiang, Xuan Song, and Ivor W Tsang. 2021. St-norm: Spatial and temporal normalization for multi-variate time series forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 269–278.

[6] Caroline Draxl, Andrew Clifton, Bri-Mathias Hodge, and Jim McCaa. 2015. The wind integration national dataset (wind) toolkit. *Applied Energy* 151 (2015), 355–366.

[7] Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 459–469.

[8] Vijay Ekambaram, Arindam Jati, Nam H Nguyen, Pankaj Dayama, Chandra Reddy, Wesley M Gifford, and Jayant Kalagnanam. 2024. TTMs: Fast Multi-level Tiny Time Mixers for Improved Zero-shot and Few-shot Forecasting of Multivariate Time Series. *arXiv preprint arXiv:2401.03955* (2024).

[9] Ziquan Fang, Lu Pan, Lu Chen, Yuntao Du, and Yunjun Gao. 2021. MDTP: A multi-source deep traffic prediction framework over spatio-temporal trajectory data. *Proceedings of the VLDB Endowment* 14, 8 (2021), 1289–1297.

[10] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. 2024. MOMENT: A Family of Open Time-series Foundation Models. In *Forty-first International Conference on Machine Learning*.

[11] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.

[12] Jindong Han, Weijia Zhang, Hao Liu, Tao Tao, Naiqiang Tan, and Hui Xiong. 2024. Bigst: Linear complexity spatio-temporal graph neural network for traffic forecasting on large-scale road networks. *Proceedings of the VLDB Endowment* 17, 5 (2024), 1081–1090.

[13] Jindong Han, Weijia Zhang, Hao Liu, and Hui Xiong. 2023. Machine learning for urban air quality analytics: A survey. *arXiv preprint arXiv:2310.09620* (2023).

[14] Guangyin Jin, Yuxuan Liang, Yuchen Fang, Zezhi Shao, Jincai Huang, Junbo Zhang, and Yu Zheng. 2023. Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[15] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. 2024. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. In *The Twelfth International Conference on Learning Representations*.

[16] Yilun Jin, Kai Chen, and Qiang Yang. 2022. Selective cross-city transfer learning for traffic prediction via source city region re-weighting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 731–741.

[17] Yilun Jin, Kai Chen, and Qiang Yang. 2023. Transferable graph structure learning for graph-based traffic forecasting across cities. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 1032–1043.

[18] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*. PMLR, 5156–5165.

[19] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

[20] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

[21] Zhe Li, Zhongwen Rao, Lujia Pan, and Zenglin Xu. 2023. Mts-mixers: Multivariate time series forecasting via factorized temporal and channel mixing. *arXiv preprint arXiv:2302.04501* (2023).

[22] Zhonghang Li, Long Xia, Lei Shi, Yong Xu, Dawei Yin, and Chao Huang. 2024. OpenCity: Open Spatio-Temporal Foundation Models for Traffic Prediction. *arXiv preprint arXiv:2408.10269* (2024).

[23] Zhonghang Li, Lianghao Xia, Jiabin Tang, Yong Xu, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024. Urbangpt: Spatio-temporal large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5351–5362.

[24] Zhonghang Li, Lianghao Xia, Yong Xu, and Chao Huang. 2024. GPT-ST: generative pre-training of spatio-temporal graph neural networks. *Advances in Neural Information Processing Systems* 36 (2024).

[25] Chenxi Liu, Sun Yang, Qianxiong Xu, Zhishuai Li, Cheng Long, Ziyue Li, and Rui Zhao. 2024. Spatial-temporal large language model for traffic prediction. *arXiv preprint arXiv:2401.10134* (2024).

[26] Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. 2023. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 4125–4129.

[27] Lingbo Liu, Jingwen Chen, Hefeng Wu, Jiajie Zhen, Guanbin Li, and Liang Lin. 2020. Physical-virtual collaboration modeling for intra-and inter-station metro ridership prediction. *IEEE Transactions on Intelligent Transportation Systems* 23, 4 (2020), 3377–3391.

[28] Lei Liu, Shuo Yu, Runze Wang, Zhenxun Ma, and Yanming Shen. 2024. How can large language models understand spatial-temporal data? *arXiv preprint arXiv:2401.14192* (2024).

[29] Xu Liu, Yuxuan Liang, Chao Huang, Hengchang Hu, Yushi Cao, Bryan Hooi, and Roger Zimmermann. 2024. Reinventing Node-centric Traffic Forecasting for Improved Accuracy and Efficiency. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 21–38.

[30] Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. 2024. Largest: A benchmark dataset for large-scale traffic forecasting. *Advances in Neural Information Processing Systems* 36 (2024).

[31] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. [n.d.]. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

[32] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. [n.d.]. Timer: Generative Pre-trained Transformers Are Large Time Series Models. In *Forty-first International Conference on Machine Learning*.

[33] Bin Lu, Xiaoying Gan, Weinan Zhang, Huaxiu Yao, Luoyi Fu, and Xinbing Wang. 2022. Spatio-temporal graph few-shot learning with cross-city knowledge transfer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1162–1172.

[34] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. [n.d.]. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.

[35] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1720–1730.

[36] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S Jensen, Zhenli Sheng, et al. 2024. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *arXiv preprint arXiv:2403.20150* (2024).

[37] Zezhi Shao, Fei Wang, Yongjun Xu, Wei Wei, Chengqing Yu, Zhao Zhang, Di Yao, Guangyin Jin, Xin Cao, Gao Cong, et al. 2023. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. *arXiv preprint arXiv:2310.06119* (2023).

[38] Zezhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu. 2022. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4454–4458.

[39] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 1567–1577.

[40] Zezhi Shao, Zhao Zhang, Wei Wei, Fei Wang, Yongjun Xu, Xin Cao, and Christian S Jensen. 2022. Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2733–2746.

[41] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. 2020. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 914–921.

[42] Jiabin Tang, Wei Wei, Lianghao Xia, and Chao Huang. 2024. EasyST: A Simple Framework for Spatio-Temporal Prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2220–2229.

[43] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems* 34 (2021), 24261–24272.

[44] Dmitry Ulyanov, Andrea Vedaldi, and Victor S Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. CoRR abs/1607.08022 (2016). *arXiv preprint arXiv:1607.08022* (2016).

[45] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[46] Jingyuan Wang, Jiawei Jiang, Wenjun Jiang, Chao Li, and Wayne Xin Zhao. 2021. Libcity: An open library for traffic prediction. In *Proceedings of the 29th international conference on advances in geographic information systems*. 145–148.

[47] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. [n.d.]. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

[48] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. Unified Training of Universal Time Series Forecasting Transformers. In *Forty-first International Conference on Machine Learning*.

[49] Xinle Wu, Xingjian Wu, Bin Yang, Lekui Zhou, Chenjuan Guo, Xiangfei Qiu, Jilin Hu, Zhenli Sheng, and Christian S Jensen. 2024. AutoCTS++: zero-shot joint neural architecture and hyperparameter search for correlated time series forecasting. *The VLDB Journal* (2024), 1–28.

[50] Xinle Wu, Dalin Zhang, Chenjuan Guo, Chaoyang He, Bin Yang, and Christian S Jensen. 2021. AutoCTS: Automated correlated time series forecasting. *Proceedings of the VLDB Endowment* 15, 4 (2021), 971–983.

[51] Xinle Wu, Dalin Zhang, Miao Zhang, Chenjuan Guo, Bin Yang, and Christian S Jensen. 2023. AutoCTS+: Joint neural architecture and hyperparameter search for correlated time series forecasting. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.

[52] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[53] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 753–763.

[54] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1907–1913.

[55] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5668–5675.

[56] Haitao Yuan, Gao Cong, and Guoliang Li. 2024. Nuhuo: An Effective Estimation Model for Traffic Speed Histogram Imputation on A Road Network. *Proceedings of the VLDB Endowment* 17, 7 (2024), 1605–1617.

[57] Yuan Yuan, Jingtao Ding, Jie Feng, Depeng Jin, and Yong Li. 2024. UniST: a prompt-empowered universal model for urban spatio-temporal prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4095–4106.

[58] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 11121–11128.

[59] Jinlei Zhang, Feng Chen, Zhiyong Cui, Yinan Guo, and Yadi Zhu. 2020. Deep learning architecture for short-term passenger flow forecasting in urban rail transit. *IEEE Transactions on Intelligent Transportation Systems* 22, 11 (2020), 7004–7014.

[60] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[61] Zijian Zhang, Ze Huang, Zhiwei Hu, Xiangyu Zhao, Wanyu Wang, Zitao Liu, Junbo Zhang, S Joe Qin, and Hongwei Zhao. 2023. MLPST: MLP is All You Need for Spatio-Temporal Prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3381–3390.

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

[63] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems* 21, 9 (2019), 3848–3858.

[64] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[65] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 1–55.

[66] Shuhan Zhong, Sizhe Song, Weipeng Zhuo, Guanyao Li, Yang Liu, and S-H Gary Chan. 2024. A Multi-Scale Decomposition MLP-Mixer for Time Series Analysis. *Proceedings of the VLDB Endowment* 17, 7 (2024), 1723–1736.

[67] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.

[68] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. 2023. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems* 36 (2023), 43322–43355.