PDFormer: Propagation Delay-aware Dynamic Long-range Transformer for Traffic Flow Prediction

Jiawei Jiang,[1*] **Chengkai Han,**[1*] **Wayne Xin Zhao,**[4] **Jingyuan Wang**[1,2,3†]

[1]School of Computer Science and Engineering, Beihang University, Beijing, China
[2]Pengcheng Laboratory, Shenzhen, China
[3]School of Economics and Management, Beihang University, Beijing, China
[4]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
{jwjiang, ckhan, jywang}@buaa.edu.cn; batmanfly@gmail.com

Abstract

As a core technology of Intelligent Transportation System, traffic flow prediction has a wide range of applications. The fundamental challenge in traffic flow prediction is to effectively model the complex spatial-temporal dependencies in traffic data. Spatial-temporal Graph Neural Network (GNN) models have emerged as one of the most promising methods to solve this problem. However, GNN-based models have three major limitations for traffic prediction: i) Most methods model spatial dependencies in a static manner, which limits the ability to learn dynamic urban traffic patterns; ii) Most methods only consider short-range spatial information and are unable to capture long-range spatial dependencies; iii) These methods ignore the fact that the propagation of traffic conditions between locations has a time delay in traffic systems. To this end, we propose a novel Propagation Delay-aware dynamic long-range transFormer, namely PDFormer, for accurate traffic flow prediction. Specifically, we design a spatial self-attention module to capture the dynamic spatial dependencies. Then, two graph masking matrices are introduced to highlight spatial dependencies from short- and long-range views. Moreover, a traffic delay-aware feature transformation module is proposed to empower PDFormer with the capability of explicitly modeling the time delay of spatial information propagation. Extensive experimental results on six real-world public traffic datasets show that our method can not only achieve state-of-the-art performance but also exhibit competitive computational efficiency. Moreover, we visualize the learned spatial-temporal attention map to make our model highly interpretable.

Introduction

In recent years, rapid urbanization has posed great challenges to modern urban traffic management. As an indispensable part of modern smart cities, intelligent transportation systems (ITS) (Yin et al. 2015) have been developed to analyze, manage, and improve traffic conditions (*e.g.,* reducing traffic congestion). As a core technology of ITS, *traffic flow prediction* (Tedjopurnomo et al. 2022) has been widely studied, aiming to predict the future flow of traffic systems based on historical observations. It has been shown that accurate traffic flow prediction can be useful for various traffic-related applications (Wang et al. 2021), including route planning, vehicle dispatching, and congestion relief.

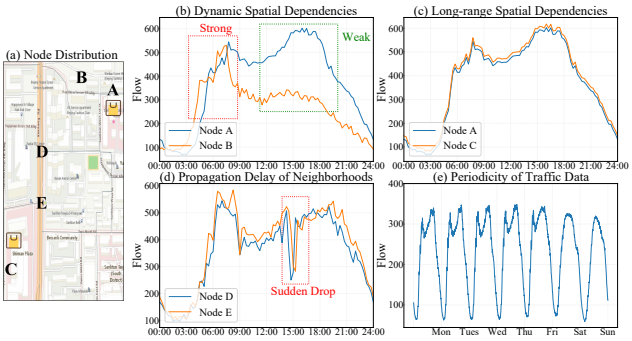

Figure 1: The Findings about Traffic Prediction.

For traffic flow prediction, the fundamental challenge is to effectively capture and model the complex and dynamic spatial-temporal dependencies of traffic data (Yin et al. 2022). Many attempts have been made in the literature to develop various deep learning models for this task. As early solutions, convolutional neural networks (CNNs) were applied to grid-based traffic data to capture spatial dependencies, and recurrent neural networks (RNNs) were used to learn temporal dynamics (Zhang, Zheng, and Qi 2017; Yao et al. 2018). Later, graph neural networks (GNNs) were shown to be more suited to model the underlying graph structure of traffic data (Li et al. 2018; Yu, Yin, and Zhu 2018), and thus GNN-based methods have been widely explored in traffic prediction (Wu et al. 2019; Song et al. 2020; Wu et al. 2020; Li and Zhu 2021; Fang et al. 2021; Choi et al. 2022).

Despite the effectiveness, GNN-based models still have three major limitations for traffic prediction. Firstly, the spatial dependencies between locations in a traffic system are highly *dynamic* instead of being static, which are time-varying as they are affected by travel patterns and unexpected events. For example, as shown in Fig. 1(b), the correlation between nodes A and B becomes stronger during the morning peak and weaker during other periods. While, existing methods model spatial dependencies mainly in a static

[*]These authors contributed equally.
[†]Corresponding author: jywang@buaa.edu.cn

manner (either predefined or self-learned), which limits the ability to learn dynamic urban traffic patterns. Secondly, due to the functional division of the city, two distant locations, such as nodes A and C in Fig. 1(c), may reflect similar traffic patterns, implying that the spatial dependencies between locations are *long-range*. Existing methods are often designed locally and unable to capture long-range dependencies. For example, GNN-based models suffer from over-smoothing, making it difficult to capture long-range spatial dependencies. Thirdly, the effect of *time delay* might occur in the spatial information propagation between locations in a traffic system. For example, when a traffic accident occurs in one location, it will take several minutes (a delay) to affect the traffic condition in neighboring locations, such as nodes D and E in Fig. 1(d). However, such a feature has been ignored in the immediate message passing mechanism of typical GNN-based models.

To address the above issues, in this paper, we propose a Propagation Delay-aware dynamic long-range transFormer model, namely PDFormer, for traffic flow prediction. As the core technical contribution, we design a novel spatial self-attention module to capture the dynamic spatial dependencies. This module incorporates local geographic neighborhood and global semantic neighborhood information into the self-attention interaction via different graph masking methods, which can simultaneously capture the short- and long-range spatial dependencies in traffic data. Based on this module, we further design a delay-aware feature transformation module to integrate historical traffic patterns into spatial self-attention and explicitly model the time delay of spatial information propagation. Finally, we adopt the temporal self-attention module to identify the dynamic temporal patterns in traffic data. In summary, the main contributions of this paper are summarized as follows:

- We propose the PDFormer model based on the spatial-temporal self-attention mechanism for accurate traffic flow prediction. Our approach fully addresses the issues caused by the complex characteristics from traffic data, namely dynamic, long-range, and time-delay.
- We design a spatial self-attention module that models both local geographic neighborhood and global semantic neighborhood via different graph masking methods and further design a traffic delay-aware feature transformation module that can explicitly model the time delay in spatial information propagation.
- We conduct both multi-step and single-step traffic flow prediction experiments on six real-world public datasets. The results show that our model significantly outperforms the state-of-the-art models and exhibits competitive computational efficiency. Moreover, the visualization experiments show that our approach is highly interpretable via the learned spatial-temporal attention.

PRELIMINARIES

In this section, we introduce some notations and formalize the traffic flow prediction problem.

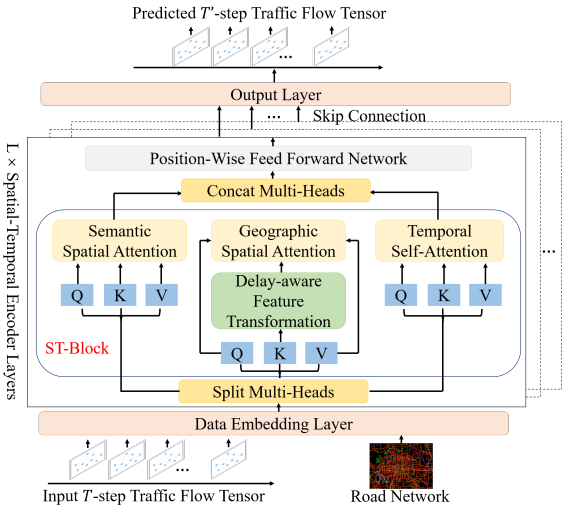

Figure 2: The Overall Framework of PDFormer.

Notations and Definitions

Definition 1 (Road Network)**.** *We represent the* Road Network *as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$, where $\mathcal{V} = \{v_1, \cdots, v_N\}$ is a set of N nodes ($|V| = N$), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, and $\boldsymbol{A}$ is the adjacency matrix of network $\mathcal{G}$. Here, N denotes the number of nodes in the graph.*

Definition 2 (Traffic Flow Tensor)**.** *We use $\boldsymbol{X}_t \in \mathbb{R}^{N \times C}$ to denote the traffic flow at time t of N nodes in the road network, where C is the dimension of the traffic flow. For example, $C = 2$ when the data includes inflow and outflow. We use $\boldsymbol{\mathcal{X}} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_T) \in \mathbb{R}^{T \times N \times C}$ to denote the traffic flow tensor of all nodes at total T time slices.*

Problem Formalization

Traffic flow prediction aims to predict the traffic flow of a traffic system in the future time given the historical observations. Formally, given the traffic flow tensor $\boldsymbol{\mathcal{X}}$ observed on a traffic system, our goal is to learn a mapping function f from the previous T steps' flow observation value to predict future T' steps' traffic flow,

$$[\boldsymbol{X}_{(t-T+1)}, \cdots, \boldsymbol{X}_t; \mathcal{G}] \xrightarrow{f} [\boldsymbol{X}_{(t+1)}, \cdots, \boldsymbol{X}_{(t+T')}]. \quad (1)$$

Methods

Fig. 2 shows the framework of PDFormer, which consists of a data embedding layer, stacked L spatial-temporal encoder layers, and an output layer. We describe each module below in detail.

Data Embedding Layer

The data embedding layer converts the input into a high-dimensional representation. First, the raw input $\boldsymbol{\mathcal{X}}$ is transformed into $\boldsymbol{\mathcal{X}}_{data} \in \mathbb{R}^{T \times N \times d}$ through a fully connected layer, d is the embedding dimension. Then, we further design a spatial-temporal embedding mechanism to incorporate the necessary knowledge into the model, including

the spatial graph Laplacian embedding to encode the road network structure and the temporal periodic embedding to model the periodicity of traffic flow.

To represent the structure of the road network, we use the graph Laplacian eigenvectors (Belkin and Niyogi 2003), which better describe the distance between nodes on the graph. First, we obtain the normalized Laplacian matrix by $\boldsymbol{\Delta} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, where $\boldsymbol{A}$ is the adjacency matrix, $\boldsymbol{D}$ is the degree matrix, and $\boldsymbol{I}$ is the identity matrix. Then, we perform the eigenvalue decomposition $\boldsymbol{\Delta} = \boldsymbol{U}^\top \boldsymbol{\Lambda} \boldsymbol{U}$ to obtain the eigenvalue matrix $\boldsymbol{\Lambda}$ and the eigenvector matrix $\boldsymbol{U}$. We use a linear projection on the k smallest non-trivial eigenvectors to generate the spatial graph Laplacian embedding $\boldsymbol{X}_{spe} \in \mathbb{R}^{N \times d}$. Laplacian eigenvectors embed the graph in Euclidean space and preserve the global graph structure information (Dwivedi et al. 2020).

In addition, urban traffic flow, influenced by people's travel patterns and lifestyle, has an obvious periodicity, such as morning and evening peak hours. Therefore, we introduce two embeddings to cover the weekly and daily periodicity, respectively, denoted as $\boldsymbol{t}_{w(t)}, \boldsymbol{t}_{d(t)} \in \mathbb{R}^d$. Here $w(t)$ and $d(t)$ are functions that transform time t into the week index (1 to 7) and minute index (1 to 1440), respectively. The temporal periodic embeddings $\boldsymbol{X}_w, \boldsymbol{X}_d \in \mathbb{R}^{T \times d}$ are obtained by concatenating the embeddings of all T time slices.

Following the original Transformer (Vaswani et al. 2017), we also employ a temporal position encoding $\boldsymbol{X}_{tpe} \in \mathbb{R}^{T \times d}$ to introduce position information of the input sequence.

Finally, we get the output of the data embedding layer by simply summing the above embedding vectors as:

$$\boldsymbol{\mathcal{X}}_{emb} = \boldsymbol{\mathcal{X}}_{data} + \boldsymbol{X}_{spe} + \boldsymbol{X}_w + \boldsymbol{X}_d + \boldsymbol{X}_{tpe}. \quad (2)$$

$\boldsymbol{\mathcal{X}}_{emb}$ will be fed into the following spatial-temporal encoders, and we use $\boldsymbol{\mathcal{X}}$ to replace $\boldsymbol{\mathcal{X}}_{emb}$ for convenience.

Spatial-Temporal Encoder Layer

We design a spatial-temporal encoder layer based on the self-attention mechanism to model the complex and dynamic spatial-temporal dependencies. The core of the encoder layer includes three components. The first is a spatial self-attention module consisting of a geographic spatial self-attention module and a semantic spatial self-attention module to capture the short-range and long-range dynamic spatial dependencies simultaneously. The second is a delay-aware feature transformation module that extends the geographic spatial self-attention module to explicitly model the time delay in spatial information propagation. Moreover, the third is a temporal self-attention module that captures the dynamic and long-range temporal patterns.

To formulate self-attention operations, we use the following slice notation. For a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{T \times N \times D}$, the t slice is the matrix $\boldsymbol{X}_{t::} \in \mathbb{R}^{N \times D}$ and the n slice is $\boldsymbol{X}_{:n:} \in \mathbb{R}^{T \times D}$.

Spatial Self-Attention (SSA**).** We design a *Spatial Self-Attention* module to capture dynamic spatial dependencies in traffic data. Formally, at time t, we first obtain the query, key, and value matrices of self-attention operations as:

$$\boldsymbol{Q}_t^{(S)} = \boldsymbol{X}_{t::}\boldsymbol{W}_Q^S, \boldsymbol{K}_t^{(S)} = \boldsymbol{X}_{t::}\boldsymbol{W}_K^S, \boldsymbol{V}_t^{(S)} = \boldsymbol{X}_{t::}\boldsymbol{W}_V^S, \quad (3)$$

where $\boldsymbol{W}_Q^S, \boldsymbol{W}_K^S, \boldsymbol{W}_V^S \in \mathbb{R}^{d \times d'}$ are learnable parameters and d' is the dimension of the query, key, and value matrix in this work. Then, we apply self-attention operations in the spatial dimension to model the interactions between nodes and obtain the spatial dependencies (attention scores) among all nodes at time t as:

$$\boldsymbol{A}_t^{(S)} = \frac{(\boldsymbol{Q}_t^{(S)})(\boldsymbol{K}_t^{(S)})^\top}{\sqrt{d'}}. \quad (4)$$

It can be seen that the spatial dependencies $\boldsymbol{A}_t^{(S)} \in \mathbb{R}^{N \times N}$ between nodes are different in different time slices, *i.e., dynamic*. Thus, the SSA module can be adapted to capture the dynamic spatial dependencies. Finally, we can obtain the output of the spatial self-attention module by multiplying the attention scores with the value matrix as:

$$\mathrm{SSA}(\boldsymbol{Q}_t^{(S)}, \boldsymbol{K}_t^{(S)}, \boldsymbol{V}_t^{(S)}) = \mathrm{softmax}(\boldsymbol{A}_t^{(S)})\boldsymbol{V}_t^{(S)}. \quad (5)$$

For the simple spatial self-attention in Eq. (5), each node interacts with all nodes, equivalent to treating the spatial graph as a fully connected graph. However, only the interaction between a few node pairs is essential, including nearby node pairs and node pairs that are far away but have similar functions. Therefore, we introduce two graph masking matrices $\boldsymbol{M}_{geo}$ and $\boldsymbol{M}_{sem}$ to simultaneously capture the *short-range* and *long-range* spatial dependencies in traffic data.

From the short-range view, we define the binary geographic masking matrix $\boldsymbol{M}_{geo}$, and only if the distance (*i.e.,* hops in the graph) between two nodes is less than a threshold λ, the weight is 1, and 0 otherwise. In this way, we can mask the attention of node pairs far away from each other. From the long-range view, we compute the similarity of the historical traffic flow between nodes using the Dynamic Time Warping (DTW) (Berndt and Clifford 1994) algorithm. We select the K nodes with the highest similarity for each node as its semantic neighbors. Then, we construct the binary semantic masking matrix $\boldsymbol{M}_{sem}$ by setting the weight between the current node and its semantic neighbors to 1 and 0 otherwise. In this way, we can find distant node pairs that exhibit similar traffic patterns due to similar urban functions.

Based on the two graph masking matrices, we further design two spatial self-attention modules, namely, *Geographic Spatial Self-Attention* (GeoSSA) and *Semantic Spatial Self-Attention* (SemSSA), which can be defined as:

$$\begin{aligned}\mathrm{GeoSSA}(\boldsymbol{Q}_t^{(S)}, \boldsymbol{K}_t^{(S)}, \boldsymbol{V}_t^{(S)}) &= \mathrm{softmax}(\boldsymbol{A}_t^{(S)} \odot \boldsymbol{M}_{geo})\boldsymbol{V}_t^{(S)}, \\ \mathrm{SemSSA}(\boldsymbol{Q}_t^{(S)}, \boldsymbol{K}_t^{(S)}, \boldsymbol{V}_t^{(S)}) &= \mathrm{softmax}(\boldsymbol{A}_t^{(S)} \odot \boldsymbol{M}_{sem})\boldsymbol{V}_t^{(S)},\end{aligned} \quad (6)$$

where $\odot$ indicates the Hadamard product. In this way, the spatial self-attention module simultaneously incorporates short-range geographic neighborhood and long-range semantic neighborhood information.

Delay-aware Feature Transformation (DFT**).** There exists a *propagation delay* in real-world traffic conditions. For example, when a traffic accident occurs in one region, it may take several minutes to affect traffic conditions in neighboring regions. Therefore, we propose a traffic delay-aware feature transformation module that captures the propagation delay from the short-term historical traffic flow of each node.

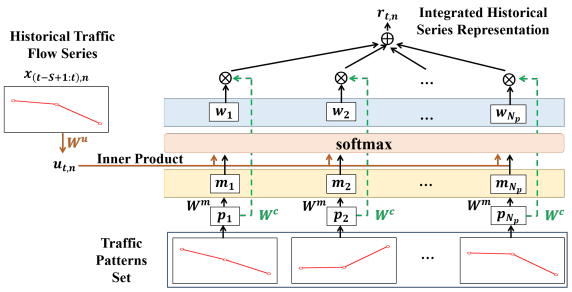

Figure 3: Delay-aware Feature Transformation.

Then, this module incorporates delay information into the key matrix of the geographic spatial self-attention module to explicitly model the time delay in spatial information propagation. Since traffic data can have multiple dimensions, such as inflow and outflow, here we only present the calculation process of this module using one dimension as an example.

First, we identify a group of representative short-term traffic patterns from historical traffic data. Specifically, we slice the historical traffic data with a sliding window of size S and obtain a set of traffic flow series. Then, we perform k-Shape clustering algorithm (Paparrizos and Gravano 2016) on these traffic flow series. The k-Shape algorithm is a time series clustering method that preserves the shape of the time series and is invariant to scaling and shifting. We use the centroid $\boldsymbol{p}_i$ of each cluster to represent that cluster, where $\boldsymbol{p}_i$ is also a time series of length S. Then, we use the set $\mathcal{P} = \{\boldsymbol{p}_i | i \in [1, \cdots, N_p]\}$ to represent the clustering results, where N_p is the total number of clusters. We can regard $\mathcal{P}$ as a set of short-term traffic patterns.

Similar traffic patterns may have similar effects on neighborhood traffic conditions, especially abnormal traffic patterns such as congestion. Therefore, we compare the historical traffic flow series for each node with the extracted traffic pattern set $\mathcal{P}$ to fuse the information of similar patterns into the historical flow series representation of each node as shown in Fig. 3. Specifically, given the S-step historical traffic flow series of node n from time slice $(t-S+1)$ to t, denoted as $\boldsymbol{x}_{(t-S+1:t),n}$, we first use the embedding matrix $\boldsymbol{W}^u$ to obtain a high-dimensional representation $\boldsymbol{u}_{t,n}$ as:

$$\boldsymbol{u}_{t,n} = \boldsymbol{x}_{(t-S+1:t),n}\boldsymbol{W}^u. \tag{7}$$

Then, we use another embedding matrix $\boldsymbol{W}^m$ to convert each traffic flow series in the traffic pattern set $\mathcal{P}$ into a memory vector as:

$$\boldsymbol{m}_i = \boldsymbol{p}_i\boldsymbol{W}^m. \tag{8}$$

We compare the historical traffic flow representation $\boldsymbol{u}_{t,n}$ of node n with the traffic pattern memory vector $\boldsymbol{m}_i$ and obtain the similarity vector as:

$$\boldsymbol{w}_i = \text{softmax}(\boldsymbol{u}_{t,n}^\top \boldsymbol{m}_i). \tag{9}$$

Then, we perform a weighted sum of the traffic pattern set $\mathcal{P}$ according to the similarity vector $\boldsymbol{w}$ to obtain the integrated historical series representation $\boldsymbol{r}_{t,n}$ as:

$$\boldsymbol{r}_{t,n} = \sum_{i=1}^{N_p} \boldsymbol{w}_i(\boldsymbol{p}_i\boldsymbol{W}^c), \tag{10}$$

where $\boldsymbol{W}^c$ is a learnable parameter matrix. The integrated historical series representation $\boldsymbol{r}_{t,n}$ contains the historical traffic flow information from time slice $(t-S+1)$ to t of node n. Finally, we use the integrated representation of N nodes, denoted as $\boldsymbol{R}_t$, to update $\boldsymbol{K}_t^{(S)}$ in Eq. (4) as:

$$\tilde{\boldsymbol{K}}_t^{(S)} = \boldsymbol{K}_t^{(S)} + \boldsymbol{R}_t, \tag{11}$$

where $\boldsymbol{R}_t \in \mathbb{R}^{N\times d'}$ is obtained by concatenating all the integrated representation $\boldsymbol{r}_{t,n}$ of N nodes and d' is the dimension of the key matrix in this work.

In this way, the new key matrix $\tilde{\boldsymbol{K}}_t^{(S)}$ at time slice t integrates the historical traffic flow information of all nodes from time slice $(t-S+1)$ to t. When computing the product of the query matrix and the new key matrix to obtain the spatial dependencies $\boldsymbol{A}_t^{(S)}$ at time slice t in Eq. (4), the query matrix can take into account the historical traffic conditions of other nodes. This process explicitly models the *time delay* in spatial information propagation. We do not add this module to the semantic spatial self-attention module because the short-term traffic flow of a distant node has little impact on the current node.

Temporal Self-Attention (TSA)**.** There are dependencies (*e.g.,* periodic, trending) between traffic conditions in different time slices, and the dependencies vary in different situations. Thus, we employ a *Temporal Self-Attention* module to discover the dynamic temporal patterns. Formally, for node n, we first obtain the query, key, and value matrices as:

$$\boldsymbol{Q}_n^{(T)} = \boldsymbol{X}_{:n:}\boldsymbol{W}_Q^T, \boldsymbol{K}_n^{(T)} = \boldsymbol{X}_{:n:}\boldsymbol{W}_K^T, \boldsymbol{V}_n^{(T)} = \boldsymbol{X}_{:n:}\boldsymbol{W}_V^T, \tag{12}$$

where $\boldsymbol{W}_Q^T, \boldsymbol{W}_K^T, \boldsymbol{W}_V^T \in \mathbb{R}^{d\times d'}$ are learnable parameters. Then, we apply self-attention operations in the temporal dimension and obtain the temporal dependencies between all time slices for node n as:

$$\boldsymbol{A}_n^{(T)} = \frac{(\boldsymbol{Q}_n^{(T)})(\boldsymbol{K}_n^{(T)}))^\top}{\sqrt{d'}}. \tag{13}$$

It can be seen that the temporal self-attention can discover the dynamic temporal patterns in traffic data that are different for different nodes. Moreover, the temporal self-attention has a global receptive to model the long-range temporal dependencies among all time slices. Finally, we can obtain the output of the temporal self-attention module as:

$$\text{TSA}(\boldsymbol{Q}_n^{(T)}, \boldsymbol{K}_n^{(T)}, \boldsymbol{V}_n^{(T)}) = \text{softmax}(\boldsymbol{A}_n^{(T)})\boldsymbol{V}_n^{(T)}. \tag{14}$$

Heterogeneous Attention Fusion. After defining the three types of attention mechanisms, we fuse heterogeneous attention into a multi-head self-attention block to reduce the computational complexity of the model. Specifically, the attention heads include three types, *i.e.,* geographic (GeoSAH), semantic (SemSAH), and temporal (TAH) heads, corresponding to the three types of attention mechanisms, respectively. The results of these heads are concatenated and projected to obtain the outputs, allowing the model to integrate spatial and temporal information simultaneously. Formally, the spatial-temporal self-attention block is defined as:

$$\text{STAttn} = \oplus(\boldsymbol{Z}_{1\cdots h_{geo}}^{geo}, \boldsymbol{Z}_{1\cdots h_{sem}}^{sem}, \boldsymbol{Z}_{1\cdots h_t}^{t})\boldsymbol{W}^O, \tag{15}$$

Table 1: Data Description.

Datasets	#Nodes	#Edges	#Timesteps	#Time Interval	Time range
PeMS04	307	340	16992	5min	01/01/2018-02/28/2018
PeMS07	883	866	28224	5min	05/01/2017-08/31/2017
PeMS08	170	295	17856	5min	07/01/2016-08/31/2016
NYCTaxi	75 (15x5)	484	17520	30min	01/01/2014-12/31/2014
CHIBike	270 (15x18)	1966	4416	30min	07/01/2020-09/30/2020
T-Drive	1024 (32x32)	7812	3600	60min	02/01/2015-06/30/2015

where $\oplus$ represents concatenation, $\boldsymbol{Z}^{geo}$, $\boldsymbol{Z}^{sem}$, $\boldsymbol{Z}^{t}$ are output concatenations and h_{geo}, h_{sem}, h_t are the numbers of attention heads of GeoSSA, SemSSA and TSA, respectively, and $\boldsymbol{W}^{O} \in \mathbb{R}^{d \times d}$ is a learnable projection matrix. In this work, we set the dimension $d' = d/(h_{geo} + h_{sem} + h_t)$.

In addition, we employ a position-wise fully connected feed-forward network on the output of the multi-head self-attention block to get the outputs $\boldsymbol{\mathcal{X}}_o \in \mathbb{R}^{T \times N \times d}$. We also use layer normalization and residual connection here following the original Transformer (Vaswani et al. 2017).

Output Layer

We use a skip connection, consisting of 1×1 convolutions, after each spatial-temporal encoder layer to convert the outputs $\boldsymbol{\mathcal{X}}_o$ into a skip dimension $\boldsymbol{\mathcal{X}}_{sk} \in \mathbb{R}^{T \times N \times d_{sk}}$. Here d_{sk} is the skip dimension. Then, we obtain the final hidden state $\boldsymbol{\mathcal{X}}_{hid} \in \mathbb{R}^{T \times N \times d_{sk}}$ by summing the outputs of each skip connection layer. To make a multi-step prediction, we directly use the output layer to transform the final hidden state $\boldsymbol{\mathcal{X}}_{hid}$ to the desired dimension as:

$$\hat{\boldsymbol{\mathcal{X}}} = \mathrm{Conv}_2(\mathrm{Conv}_1(\boldsymbol{\mathcal{X}}_{hid})), \tag{16}$$

where $\hat{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{T' \times N \times C}$ is T' steps' prediction results, Conv_1 and Conv_2 are 1×1 convolutions. Here we choose the direct way instead of the recursive manner for multi-step prediction considering cumulative errors and model efficiency.

Experiments

Datasets

We verify the performance of PDFormer on six real-world public traffic datasets, including three graph-based highway traffic datasets, *i.e.,* PeMS04, PeMS07, PeMS08 (Song et al. 2020), and three grid-based citywide traffic datasets, *i.e.,* NYCTaxi (Liu et al. 2021a), CHIBike (Wang et al. 2021), T-Drive (Pan et al. 2019). The graph-based datasets contain only the traffic flow data, and the grid-based datasets contain inflow and outflow data. Details are given in Tab. 1.

Baselines

We compare PDFormer with the following 17 baselines belonging to four classes. (1) *Models For Grid-based Datasets*: We choose STResNet (Zhang, Zheng, and Qi 2017), DMVSTNet (Yao et al. 2018) and DSAN (Lin et al. 2020a), which are unsuitable for graph-based datasets. (2) *Time Series Prediction Models*: We choose VAR (Lu et al. 2016) and SVR (Drucker et al. 1996). (3) *Graph Neural Network-based Models*: We choose DCRNN (Li et al. 2018), STGCN (Yu, Yin, and Zhu 2018), GWNET (Wu et al. 2019), MTGNN (Wu et al. 2020), STSGCN (Song et al. 2020), STFGNN (Li and Zhu 2021), STGODE (Fang et al. 2021) and STGNCDE (Choi et al. 2022). (4) *Self-attention-based Models*: We choose STTN (Xu et al. 2020), GMAN (Zheng et al. 2020), TFormer (Yan and Ma 2021) and ASTGNN (Guo et al. 2021).

Table 2: Performance on Graph-based Datasets.

Model	PeMS04			PeMS07			PeMS08		
	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
VAR	23.750	18.090	36.660	101.200	39.690	155.140	22.320	14.470	33.830
SVR	28.660	19.150	44.590	32.970	15.430	50.150	23.250	14.710	36.150
DCRNN	22.737	14.751	36.575	23.634	12.281	36.514	18.185	11.235	28.176
STGCN	21.758	13.874	34.769	22.898	11.983	35.440	17.838	11.211	27.122
GWNET	19.358	13.301	31.719	21.221	9.075	34.117	15.063	9.514	24.855
MTGNN	19.076	12.961	31.564	20.824	9.032	34.087	15.396	10.170	24.934
STSGCN	21.185	13.882	33.649	24.264	10.204	39.034	17.133	10.961	26.785
STFGNN	19.830	13.021	31.870	22.072	9.212	35.805	16.636	10.547	26.206
STGODE	20.849	13.781	32.825	22.976	10.142	36.190	16.819	10.623	26.240
STGNCDE	19.211	12.772	31.088	20.620	8.864	34.036	15.455	9.921	24.813
STTN	19.478	13.631	31.910	21.344	9.932	34.588	15.482	10.341	24.965
GMAN	19.139	13.192	31.601	20.967	9.052	34.097	15.307	10.134	24.915
TFormer	18.916	12.711	31.349	20.754	8.972	34.062	15.192	9.925	24.883
ASTGNN	18.601	12.630	31.028	20.616	8.861	34.017	14.974	9.489	24.710
PDFormer	**18.321**	**12.103**	**29.965**	**19.832**	**8.529**	**32.870**	**13.583**	**9.046**	**23.505**

Experimental Settings

Dataset Processing. To be consistent with most modern methods, we split the three graph-based datasets into training, validation, and test sets in a 6:2:2 ratio. In addition, we use the past hour (12 steps) data to predict the traffic flow for the next hour (12 steps), *i.e.,* a multi-step prediction. For the grid-based datasets, the split ratio is 7:1:2, and we use the traffic inflow and outflow of the past six steps to predict the next single-step traffic inflow and outflow.

Model Settings. All experiments are conducted on a machine with the NVIDIA GeForce 3090 GPU and 128GB memory. We implement PDFormer [1] with Ubuntu 18.04, PyTorch 1.10.1, and Python 3.9.7. The hidden dimension d is searched over $\{16, 32, 64, 128\}$ and the depth of encoder layers L is searched over $\{2, 4, 6, 8\}$. The optimal model is determined based on the performance in the validation set. We train our model using AdamW optimizer (Loshchilov and Hutter 2017) with a learning rate of 0.001. The batch size is 16, and the training epoch is 200.

Evaluation Metrics. We use three metrics in the experiments: (1) Mean Absolute Error (MAE), (2) Mean Absolute Percentage Error (MAPE), and (3) Root Mean Squared Error (RMSE). Missing values are excluded when calculating these metrics. When we test the models on grid-based datasets, we filter the samples with flow values below 10, consistent with (Yao et al. 2018). Since the flow of CHIBike is lower than others, the filter threshold is 5. We repeated all experiments ten times and reported the average results.

Performance Comparison

The comparison results with baselines on graph-based and grid-based datasets are shown in Tab. 2 and Tab. 3, respectively. The bold results are the best, and the underlined results are the second best. Based on these two tables, we can make the following observations. (1) Spatial-temporal deep learning models perform better than traditional time

[1] https://github.com/BUAABIGSCity/PDFormer

Table 3: Performance on Grid-based Datasets.

Datasets	NYCTaxi						T-Drive						CHIBike					
Metrics	inflow			outflow			inflow			outflow			inflow			outflow		
Models	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
STResNet	14.492	14.543	24.050	12.798	14.368	20.633	19.636	17.831	34.890	19.616	18.502	34.597	4.767	31.382	6.703	4.627	30.571	6.559
DMVSTNet	14.377	14.314	23.734	12.566	14.318	20.409	19.599	17.683	34.478	19.531	17.621	34.303	4.687	32.113	6.635	4.594	31.313	6.455
DSAN	14.287	14.208	23.585	12.462	14.272	20.294	19.384	17.465	34.314	19.290	17.379	34.267	4.612	31.621	6.695	4.495	31.256	6.367
DCRNN	14.421	14.353	23.876	12.828	14.344	20.067	22.121	17.750	38.654	21.755	17.382	38.168	4.236	31.264	5.992	4.211	30.822	5.824
STGCN	14.377	14.217	23.860	12.547	14.095	19.962	21.373	17.539	38.052	20.913	16.984	37.619	4.212	31.224	5.954	4.148	30.782	5.779
GWNET	14.310	14.198	23.799	12.282	13.685	19.616	19.556	17.187	36.159	19.550	15.933	36.198	4.151	31.153	5.917	4.101	30.690	5.694
MTGNN	14.194	13.984	23.663	12.272	13.652	19.563	18.982	17.056	35.386	18.929	15.762	35.992	4.112	31.148	5.807	4.086	30.561	5.669
STSGCN	15.604	15.203	26.191	13.233	14.698	21.653	23.825	18.547	41.188	24.287	19.041	42.255	4.256	32.991	5.941	4.265	32.612	5.879
STFGNN	15.336	14.869	26.112	13.178	14.584	21.627	22.144	18.094	40.071	22.876	18.987	41.037	4.234	32.222	5.933	4.264	32.321	5.875
STGODE	14.621	14.793	25.444	12.834	14.398	20.205	21.515	17.579	38.215	22.703	18.509	40.282	4.169	31.165	5.921	4.125	30.726	5.698
STGNCDE	14.281	14.171	23.742	12.276	13.681	19.608	19.347	17.134	36.093	19.230	15.873	36.143	4.123	31.151	5.913	4.094	30.595	5.678
STTN	14.359	14.206	23.841	12.373	13.762	19.827	20.583	17.327	37.220	20.443	15.992	37.067	4.160	31.208	5.932	4.118	30.704	5.723
GMAN	14.267	14.114	23.728	12.273	13.672	19.594	19.244	17.110	35.986	18.964	15.788	36.120	4.115	31.150	5.910	4.090	30.662	5.675
TFormer	13.995	13.912	23.487	12.211	13.611	19.522	18.823	16.910	34.470	18.883	15.674	35.219	4.071	31.141	5.878	4.037	30.647	5.638
ASTGNN	13.844	13.692	23.177	12.112	13.602	19.201	18.798	16.101	33.870	18.790	15.584	33.998	4.068	31.131	5.818	3.981	30.617	5.609
PDFormer	**13.152**	**12.743**	**21.957**	**11.575**	**12.820**	**18.394**	**17.832**	**14.711**	**31.606**	**17.743**	**14.649**	**31.501**	**3.950**	**30.214**	**5.559**	**3.837**	**29.914**	**5.402**

series prediction models such as VAR because the latter ignores the spatial dependencies in the traffic data. (2) Our PDFormer significantly outperforms all baselines in terms of all metrics over all datasets according to Student's t-test at level 0.01. Compared to the second best method, PDFormer achieves an average improvement of 4.58%, 5.00%, 4.79% for MAE/MAPE/RMSE. (3) Among the GNN-based models, MTGNN and STGNCDE lead to competitive performance. Compared to these GNN-based models, whose message passing is immediate, PDFormer achieves better performance because it considers the time delay in spatial information propagation. (4) As for the self-attention-based models, ASTGNN is the best baseline, which combines GCN and the self-attention module to aggregate neighbor information. Compared with ASTGNN, PDFormer simultaneously captures short- and long-range spatial dependencies via two masking matrices and achieves good performance. (5) Although STResNet, DMVSTNet, and DSAN are designed for grid-based data, they need to extract features from different time periods, which are excluded here, resulting in relatively poor performance.

Ablation Study

To further investigate the effectiveness of different parts in PDFormer, we compare PDFormer with the following variants. (1) *w/ GCN*: this variant replaces spatial self-attention (SSA) with Graph Convolutional Network (GCN) (Kipf and Welling 2016), which cannot capture the dynamic and long-range spatial dependencies. (2) *w/o Mask*: this variant removes two masking matrices $\boldsymbol{M}_{geo}$ and $\boldsymbol{M}_{sem}$, which means each node attends to all nodes. (3) *w/o GeoSAH*: this variant removes GeoSAH. (4) *w/o SemSAH*: this variant removes SemSAH. (5) *w/o Delay*: this variant removes the delay-aware feature transformation module, which accounts for the spatial information propagation delay.

Fig. 4 shows the comparison of these variants on the PeMS04 and NYCTaxi datasets. For the NYCTaxi dataset, only the results for inflow are reported since the results for outflow are similar. Based on the results, we can conclude the following: (1) The results show the superiority of SSA over GCN in capturing dynamic and long-range spatial dependencies. (2) PDFormer leads to a large performance improvement over *w/o Mask*, highlighting the value of using the mask matrices to identify the significant node pairs. In addition, *w/o SemSAH* and *w/o GeoSAH* perform worse than PDFormer, indicating that both local and global spatial dependencies are significant for traffic prediction. (3) *w/o Delay* performs worse than PDFormer because this variant ignores the spatial propagation delay between nodes but considers the spatial message passing as immediate.

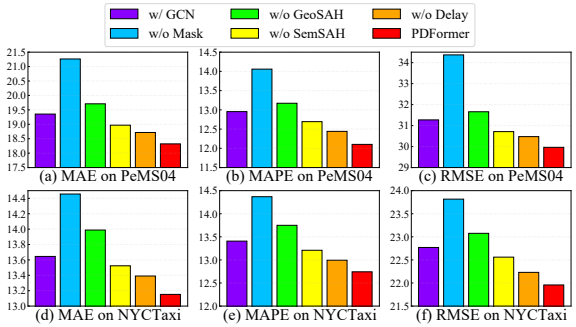

Figure 4: Ablation Study on PeMS04 and NYCTaxi inflow.

Case Study

In this section, we analyze the dynamic spatial-temporal attention weight map learned by the spatial-temporal encoder of PDFormer to improve its interpretability and demonstrate the effectiveness of focusing on short- and long-range spatial dependencies simultaneously.

We compare and visualize the attention map in two cases, *i.e.,* with or without the two spatial mask matrices $\boldsymbol{M}_{geo}$ and $\boldsymbol{M}_{sem}$. Here, for simplicity, we merge the attention map of GeoSAH and SemSAH. As shown in Fig. 5(a),(d), without the mask matrices, the model focuses on the major urban ring roads (or highways) with high traffic volume, or the attention distribution is diffuse, and almost the entire city shares the model's attention. However, low-traffic locations should focus on locations with similar patterns rather than hot locations. Moreover, locations that are too far away have little impact on the current location. The model performance will weaken if it focuses on all locations diffusely. Instead, when $\boldsymbol{M}_{geo}$ and $\boldsymbol{M}_{sem}$ are introduced, attention focuses on surrounding locations and distant similar-pattern locations as shown in Fig. 5(b),(e).

Let us take Region 592 in Fig. 5(b) as an example. High-

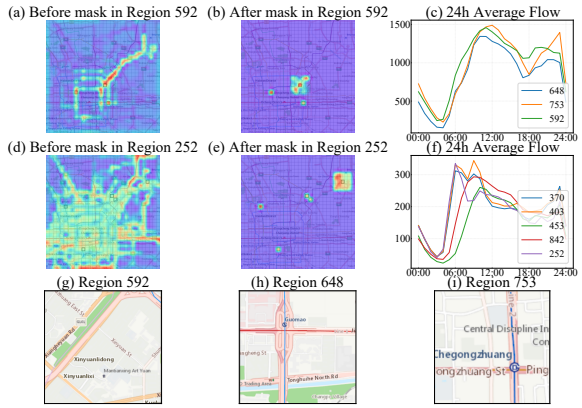

Figure 5: Case Study of Attention Map.

way S12 passes through this region, so the traffic volume is always high. In addition to the regions located upstream and downstream of the highway, region 592 also focuses on regions 648 and 753. From Fig. 5(c), we can see that these two regions have similar historical traffic volumes as 592. Besides, from Fig. 5(h)(i), these two regions are located near the Guomao Interchange and Beijing Second Ring Road, respectively, which are similar to 592 in their cities function as major traffic hubs. In another case, region 252 has low traffic volume, but we can observe similar patterns from regions 370, 403, 453, and 842 that region 252 focused on, *i.e.,* similar functional and historical traffic variations.

This case study shows that after introducing the spatial mask matrices, PDFormer not only considers the short-range spatial dependencies but also identifies the global functional area to capture the long-range spatial dependencies. The above ablation experiments also quantitatively show that the model performance drops sharply after removing the mask matrices, which supports the idea presented here.

Model Efficiency Study

Due to the better performance of the attention-based models, we compare the computational cost of PDFormer with other self-attention-based baselines on the PeMS04 and the NYCTaxi datasets. Tab. 4 reports the average training and inference time per epoch. We find that PDFormer achieves competitive computational efficiency in both short- and long-term traffic prediction. Compared to the best performing baseline ASTGNN on PeMS04, PDFormer reduces the training and inference time of over 35% and 80%, respectively. GMAN and ASTGNN retain a time-consuming encoder-decoder structure, which is replaced by a forward procedure in STTN and PDFormer. TFormer performs only spatial attention, resulting in less computation time.

Related Work

Deep Learning for Traffic Prediction

In recent years, more and more researchers have employed deep learning models to solve traffic prediction problems. Early on, convolutional neural networks (CNNs) were applied to grid-based traffic data to capture spatial dependencies in the data (Zhang, Zheng, and Qi 2017; Lin et al. 2020a). Later, thanks to the powerful ability to model graph data, graph neural networks (GNNs) were widely used for traffic prediction (Chen et al. 2020; Bai et al. 2020; Zhang et al. 2021; Oreshkin et al. 2021; Ye et al. 2021; Han et al. 2021; Ji et al. 2022; Liu et al. 2022). Recently, the attention mechanism has become increasingly popular due to its effectiveness in modeling the dynamic dependencies in traffic data (Guo et al. 2019; Wang et al. 2020; Lin et al. 2020b; Guo et al. 2021; Ye et al. 2022). Unlike these work, our proposed PDFormer not only considers the dynamic and long-range spatial dependencies through a self-attention mechanism but also incorporates the time delay in spatial propagation through a delay-aware feature transformation layer.

Table 4: Training and inference time per epoch comparison between self-attention-based models. (Unit: seconds)

Dataset	PeMS04		NYCTaxi	
Model	Training	Inference	Training	Inference
GMAN	501.578	38.844	130.672	4.256
ASTGNN	208.724	52.016	119.092	4.601
PDFormer	133.871	8.120	85.305	2.734
STTN	100.398	12.596	68.036	2.650
TFormer	71.099	7.156	76.169	2.575

Transformer

Transformer (Vaswani et al. 2017) is a network architecture based entirely on self-attention mechanisms. Transformer has been proven effective in multiple natural language processing (NLP) tasks. In addition, large-scale Transformer-based pre-trained models such as BERT (Devlin et al. 2018) have achieved great success in the NLP community. Recently, Vision Transformers have attracted the attention of researchers, and many variants have shown promising results on computer vision tasks (Dosovitskiy et al. 2021; Liu et al. 2021b). In addition, the Transformer architecture performs well in representation learning, which has been demonstrated in recent studies (Dwivedi and Bresson 2020; Ren et al. 2021; Jiang et al. 2023).

Conclusion

In this work, we proposed a novel PDFormer model with spatial-temporal self-attention for traffic flow prediction. Specifically, we developed a spatial self-attention module that captures the dynamic and long-range spatial dependencies and a temporal self-attention module that discovers the dynamic temporal patterns in the traffic data. We further designed a delay-aware feature transformation module to explicitly model the time delay in spatial information propagation. We conducted extensive experiments on six real-world datasets to demonstrate the superiority of our proposed model and visualized the learned attention map to make the model interpretable. As future work, we will apply PDFormer to other spatial-temporal prediction tasks, such as wind power forecasting (Jiang, Han, and Wang 2022). In addition, we will explore the pre-training techniques in traffic prediction to solve the problem of insufficient data.

Acknowledgment

This work was supported by the National Key R&D Program of China (2021ZD0111201). Prof. Wang's work was supported by the National Natural Science Foundation of China (No. 72171013, 82161148011, 72222022), the Fundamental Research Funds for the Central Universities (YWF-22-L-838) and the DiDi Gaia Collaborative Research Funds. Prof. Zhao's work was supported by the National Natural Science Foundation of China (No. 62222215).

References

Bai, L.; Yao, L.; Li, C.; Wang, X.; and Wang, C. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. In *NeurIPS*.

Belkin, M.; and Niyogi, P. 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comput.*, 15(6): 1373–1396.

Berndt, D. J.; and Clifford, J. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD Workshop*, 359–370. AAAI Press.

Chen, W.; Chen, L.; Xie, Y.; Cao, W.; Gao, Y.; and Feng, X. 2020. Multi-Range Attentive Bicomponent Graph Convolutional Network for Traffic Forecasting. In *AAAI*, 3529–3536. AAAI Press.

Choi, J.; Choi, H.; Hwang, J.; and Park, N. 2022. Graph Neural Controlled Differential Equations for Traffic Forecasting. In *AAAI*, 6367–6374. AAAI Press.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*. OpenReview.net.

Drucker, H.; Burges, C. J. C.; Kaufman, L.; Smola, A. J.; and Vapnik, V. 1996. Support Vector Regression Machines. In *NIPS*, 155–161. MIT Press.

Dwivedi, V. P.; and Bresson, X. 2020. A Generalization of Transformer Networks to Graphs. *CoRR*, abs/2012.09699.

Dwivedi, V. P.; Joshi, C. K.; Laurent, T.; Bengio, Y.; and Bresson, X. 2020. Benchmarking Graph Neural Networks. *CoRR*, abs/2003.00982.

Fang, Z.; Long, Q.; Song, G.; and Xie, K. 2021. Spatial-Temporal Graph ODE Networks for Traffic Flow Forecasting. In *KDD*, 364–373. ACM.

Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. In *AAAI*, 922–929. AAAI Press.

Guo, S.; Lin, Y.; Wan, H.; Li, X.; and Cong, G. 2021. Learning Dynamics and Heterogeneity of Spatial-Temporal Graph Data for Traffic Forecasting. *IEEE Trans. Knowl. Data Eng.*, 1–1.

Han, L.; Du, B.; Sun, L.; Fu, Y.; Lv, Y.; and Xiong, H. 2021. Dynamic and Multi-faceted Spatio-temporal Deep Learning for Traffic Speed Forecasting. In *KDD*, 547–555. ACM.

Ji, J.; Wang, J.; Jiang, Z.; Jiang, J.; and Zhang, H. 2022. STDEN: Towards Physics-Guided Neural Networks for Traffic Flow Prediction. In *AAAI*, 4048–4056. AAAI Press.

Jiang, J.; Han, C.; and Wang, J. 2022. BUAA_BIGSCity: Spatial-Temporal Graph Neural Network for Wind Power Forecasting in Baidu KDD CUP 2022.

Jiang, J.; Pan, D.; Ren, H.; Jiang, X.; Li, C.; and Wang, J. 2023. Self-supervised Trajectory Representation Learning with Temporal Regularities and Travel Semantics. In *2023 IEEE 39th international conference on data engineering (ICDE)*. IEEE.

Kipf, T. N.; and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs/1609.02907.

Li, M.; and Zhu, Z. 2021. Spatial-Temporal Fusion Graph Neural Networks for Traffic Flow Forecasting. In *AAAI*, 4189–4196. AAAI Press.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR (Poster)*. OpenReview.net.

Lin, H.; Bai, R.; Jia, W.; Yang, X.; and You, Y. 2020a. Preserving Dynamic Attention for Long-Term Spatial-Temporal Prediction. In *KDD*, 36–46. ACM.

Lin, Z.; Li, M.; Zheng, Z.; Cheng, Y.; and Yuan, C. 2020b. Self-Attention ConvLSTM for Spatiotemporal Prediction. In *AAAI*, 11531–11538. AAAI Press.

Liu, D.; Wang, J.; Shang, S.; and Han, P. 2022. MSDR: Multi-Step Dependency Relation Networks for Spatial Temporal Forecasting. In *KDD*, 1042–1050. ACM.

Liu, L.; Zhen, J.; Li, G.; Zhan, G.; He, Z.; Du, B.; and Lin, L. 2021a. Dynamic Spatial-Temporal Representation Learning for Traffic Flow Prediction. *IEEE Trans. Intell. Transp. Syst.*, 22(11): 7169–7183.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021b. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *CoRR*, abs/2103.14030.

Loshchilov, I.; and Hutter, F. 2017. Fixing Weight Decay Regularization in Adam. *CoRR*, abs/1711.05101.

Lu, Z.; Zhou, C.; Wu, J.; Jiang, H.; and Cui, S. 2016. Integrating Granger Causality and Vector Auto-Regression for Traffic Prediction of Large-Scale WLANs. *KSII Trans. Internet Inf. Syst.*, 10(1): 136–151.

Oreshkin, B. N.; Amini, A.; Coyle, L.; and Coates, M. 2021. FC-GAGA: Fully Connected Gated Graph Architecture for Spatio-Temporal Traffic Forecasting. In *AAAI*, 9233–9241. AAAI Press.

Pan, Z.; Liang, Y.; Wang, W.; Yu, Y.; Zheng, Y.; and Zhang, J. 2019. Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning. In *KDD*, 1720–1730. ACM.

Paparrizos, J.; and Gravano, L. 2016. k-Shape: Efficient and Accurate Clustering of Time Series. *SIGMOD Rec.*, 45(1): 69–76.

Ren, H.; Wang, J.; Zhao, W. X.; and Wu, N. 2021. RAPT: Pre-training of Time-Aware Transformer for Learning Robust Healthcare Representation. In *KDD*, 3503–3511. ACM.

Song, C.; Lin, Y.; Guo, S.; and Wan, H. 2020. Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting. In *AAAI*, 914–921. AAAI Press.

Sukhbaatar, S.; Szlam, A.; Weston, J.; and Fergus, R. 2015. End-To-End Memory Networks. In *NIPS*, 2440–2448.

Tedjopurnomo, D. A.; Bao, Z.; Zheng, B.; Choudhury, F. M.; and Qin, A. K. 2022. A Survey on Modern Deep Neural Network for Traffic Prediction: Trends, Methods and Challenges. *IEEE Trans. Knowl. Data Eng.*, 34(4): 1544–1561.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NIPS*, 5998–6008.

Wang, J.; Jiang, J.; Jiang, W.; Li, C.; and Zhao, W. X. 2021. LibCity: An Open Library for Traffic Prediction. In *SIGSPATIAL/GIS*, 145–148. ACM.

Wang, X.; Ma, Y.; Wang, Y.; Jin, W.; Wang, X.; Tang, J.; Jia, C.; and Yu, J. 2020. Traffic Flow Prediction via Spatial Temporal Graph Neural Network. In *WWW*, 1082–1092. ACM / IW3C2.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *KDD*, 753–763. ACM.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI*, 1907–1913. ijcai.org.

Xu, M.; Dai, W.; Liu, C.; Gao, X.; Lin, W.; Qi, G.; and Xiong, H. 2020. Spatial-Temporal Transformer Networks for Traffic Flow Forecasting. *CoRR*, abs/2001.02908.

Yan, H.; and Ma, X. 2021. Learning dynamic and hierarchical traffic spatiotemporal features with Transformer. *CoRR*, abs/2104.05163.

Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; and Li, Z. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. In *AAAI*, 2588–2595. AAAI Press.

Ye, J.; Sun, L.; Du, B.; Fu, Y.; and Xiong, H. 2021. Coupled Layer-wise Graph Convolution for Transportation Demand Prediction. In *AAAI*, 4617–4625. AAAI Press.

Ye, X.; Fang, S.; Sun, F.; Zhang, C.; and Xiang, S. 2022. Meta Graph Transformer: A Novel Framework for Spatial-Temporal Traffic Prediction. *Neurocomputing*, 491: 544–563.

Yin, C.; Xiong, Z.; Chen, H.; Wang, J.; Cooper, D.; and David, B. 2015. A literature survey on smart cities. *Sci. China Inf. Sci.*, 58(10): 1–18.

Yin, X.; Wu, G.; Wei, J.; Shen, Y.; Qi, H.; and Yin, B. 2022. Deep Learning on Traffic Prediction: Methods, Analysis, and Future Directions. *IEEE Trans. Intell. Transp. Syst.*, 23(6): 4927–4943.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *IJCAI*, 3634–3640. ijcai.org.

Zhang, J.; Zheng, Y.; and Qi, D. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. In *AAAI*, 1655–1661. AAAI Press.

Zhang, X.; Huang, C.; Xu, Y.; Xia, L.; Dai, P.; Bo, L.; Zhang, J.; and Zheng, Y. 2021. Traffic Flow Forecasting with Spatial-Temporal Graph Diffusion Network. In *AAAI*, 15008–15015. AAAI Press.

Zheng, C.; Fan, X.; Wang, C.; and Qi, J. 2020. GMAN: A Graph Multi-Attention Network for Traffic Prediction. In *AAAI*, 1234–1241. AAAI Press.