Do We Really Need GCNs in Traffic Forecasting? A Graph-Less Pure-MLP Architecture

Yudong Zhang
School of Artificial Intelligence and Data Science, University of Science and Technology of China
Hefei, China
zyd2020@mail.ustc.edu.cn

Xu Wang*
School of Software Engineering, University of Science and Technology of China
Hefei, China
Suzhou Institute for Advanced Research, University of Science and Technology of China
Suzhou, China
wx309@ustc.edu.cn

Xuan Yu
School of Software Engineering, University of Science and Technology of China
Hefei, China
yx2024@mail.ustc.edu.cn

Kuo Yang
School of Artificial Intelligence and Data Science, University of Science and Technology of China
Hefei, China
yangkuo@mail.ustc.edu.cn

Zhengyang Zhou
School of Software Engineering, University of Science and Technology of China
Hefei, China
Suzhou Institute for Advanced Research, University of Science and Technology of China
Suzhou, China
zzy0929@ustc.edu.cn

Yang Wang*
School of Software Engineering, University of Science and Technology of China
Hefei, China
Suzhou Institute for Advanced Research, University of Science and Technology of China
Suzhou, China
angyan@ustc.edu.cn

Abstract

Traffic flow forecasting, as a fundamental task of intelligent transportation systems driven by the Web, remains challenging due to the complex spatial and temporal dependencies between different roads. Considering the inherent graph structure of road networks, recent works capture the spatial dependencies by utilizing graph convolutional networks (GCNs) and have devoted great efforts to enhancing GCNs. However, graph convolution, as the core of GCNs, has quadratic time and space complexity due to the internal matrix multiplication operation on the adjacency matrix, which prevents existing works from being deployed on huge road networks. In this paper, we doubt the necessity of graph convolution in capturing spatial dependencies and find that the aggregating operation in graph convolution can be replaced by Multi-Layer Perceptron (MLP) with layer normalization in traffic forecasting. Inspired by the emerging MLP models on other deep learning tasks, we further propose a pure-MLP traffic forecasting model without the dependence on graph convolution. Experiments on several real-world traffic datasets show that our pure-MLP model is more efficient and has competitive performances compared to state-of-the-art works.

CCS Concepts

• **Information systems** → **Spatial-temporal systems**; **Data mining**; • **Computing methodologies** → **Machine learning**.

Keywords

Traffic Forecasting; Multi-Layer Perceptron; Intelligent Transportation Systems

ACM Reference Format:
Yudong Zhang, Xu Wang, Xuan Yu, Kuo Yang, Zhengyang Zhou, and Yang Wang. 2025. Do We Really Need GCNs in Traffic Forecasting? A Graph-Less Pure-MLP Architecture. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3701716.3715464

1 Introduction

Traffic flow forecasting, as the premise and foundation of intelligent transportation systems driven by the Web, aims at predicting the future status of traffic systems, thereby assisting people in the management and scheduling of urban traffic [15]. Due to its huge practical value, traffic forecasting has attracted more and more attention from the data mining and artificial intelligence research community [2, 14]. The core challenge of traffic forecasting lies in capturing spatial and temporal dependencies between different nodes in road networks and different time steps. Recent works mainly utilize graph convolutional networks (GCNs) to model the

*Corresponding authors.

WWW Companion '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1331-6/25/04
https://doi.org/10.1145/3701716.3715464

spatial dependencies between nodes and have devoted great efforts to enhancing GCNs. Since GCNs calculate the products of adjacency matrices and features of nodes, the performance of existing works heavily depends on the strategies for constructing adjacency matrices.

Lots of novel strategies for constructing more effective adjacency matrices have been proposed for more accurate prediction. Early works [7, 13] intuitively define adjacency matrices according to the distance between nodes, and believe that closer nodes are higher correlated. However, distant nodes may share similar patterns and have certain correlations, *e.g.*, residential areas that are far away may have similar traffic flow. Following works [1, 10] make attempts to enhance the representation of the spatial model, and propose to learn node embeddings, which are utilized to infer the proximity among nodes by calculating their embedding similarities. Similar to early distance-based adjacency matrices, the learned adjacency matrices in those works are kept fixed after training and fall short of representing dynamic spatial dependencies in traffic data. To tackle this shortage, [4, 12] apply self-attention mechanism for modeling spatial dependencies, which enables those models to build attention matrices at each time step, and can be seen as building dynamic adjacency matrices. Although existing works achieve improved prediction performance, the strategies for calculating adjacency matrices have become increasingly complex, resulting in larger and more time-consuming models. ***Challenge:*** It is worth noting that even for vanilla graph convolution, *the time and space complexity scales quadratically with the number of nodes.* This unsatisfactory time and space complexity hinders the deployment of these models on large-scale road networks with a significant number of nodes.

Insight: Considering the efficiency challenge posed by graph convolution in existing works, we can tackle this issue by replacing graph convolution with alternative modules for modeling spatial dependencies. ***Improvement:*** In this paper, we propose a **G**raph-**L**ess **M**LP **S**patio**T**emporal (**GLMST**) architecture for traffic forecasting, whose time and space complexity is linear to the number of nodes. The proposed GLMST consists of *Graph-Less Spatial MLP blocks (GLSM)* for capturing spatial dependencies and *Cross-Channel Temporal MLP blocks (CCTM)* for modeling temporal dependencies. In the spatial perspective, instead of utilizing graph convolution, we extract spatial dependencies by a simple fully-connected layer equipped with residual connection and layer normalization, where layer normalization connects the nodes in a graph-free way. In the temporal perspective, we apply two fully-connected layers in the time dimension and the feature dimension, respectively. We evaluate the proposed GLMST on several widely used real-world datasets for the multi-step traffic prediction task. Experimental results demonstrate that the proposed pure-MLP architecture outperforms existing methods in terms of both accuracy and efficiency.

2 Methodology

2.1 Problem Definition

Def 1 (Basic graph structure). A spatial graph $\mathbf{G}$ can be denoted as $\mathbf{G}(\mathbf{A}, \mathbf{V}, \mathbf{E})$, where $\mathbf{A}$, $\mathbf{V}$ and $\mathbf{E}$ denote the adjacency matrix, the set of vertices and edges respectively, $v_i \in \mathbf{V}(1 \le i \le |\mathbf{V}|)$ corresponds to the i-th vertex in $\mathbf{V}$. Since $\mathbf{A}$ contains the same information as $\mathbf{E}$, we may also denote a graph as $\mathbf{G}(\mathbf{A}, \mathbf{V})$ or $\mathbf{G}(\mathbf{V}, \mathbf{E})$.

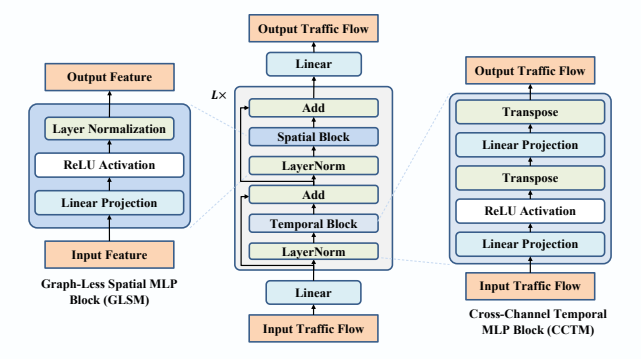

Figure 1: Overview of the proposed GLMST framework.

Def 2 (Historical spatiotemporal data). The feature matrix of all vertices in graph $\mathbf{G}$ in time interval t can be denoted as $\mathbf{V}^t \in \mathbb{R}^{|\mathbf{V}| \times D}$, where D is the dimensionality of the feature of each vertex. The historical spatiotemporal features of $\mathbf{G}$ in historical P time intervals can be formulated as $\left[\mathbf{V}^{(t-P+1)}, \mathbf{V}^{(t-P+2)}, \cdots, \mathbf{V}^t\right] \in \mathbb{R}^{P \times |\mathbf{V}| \times D}$.

Def 3 (Traffic forecasting). The goal of traffic forecasting is to learn a function f that predicts the spatiotemporal data of $\mathbf{G}$ for the next Q time steps based on the data of $\mathbf{G}$ from the past P steps, *i.e.*,

$$[\mathbf{V}^{t-P+1}, \cdots, \mathbf{V}^t,] \xrightarrow{f} [\mathbf{V}^{t+1}, \cdots, \mathbf{V}^{t+Q}]. \tag{1}$$

2.2 Overall Structure

Following the Transformer architecture [9], we design GLMST, as illustrated in Figure 1, which consists of three main components: (1) stacked L spatiotemporal blocks, each incorporating GLSM, CCTM, residual connections, and normalization modules; (2) an input linear projection without activation function for projecting the input $\mathbf{V}^t$ from d-dimension to d_{model}-dimension; and (3) an output linear projection without activation function for projecting the output of stacked spatiotemporal blocks from d_{model}-dimension back to d-dimension. During the process of spatiotemporal block, including GLSM and CCTM, the feature dimension is maintained as d_{model}.

2.3 Cross-Channel Temporal MLP Block

Existing works have designed lots of novel but complex modules for capturing temporal dependencies. Motivated by [8], we propose a Cross-Channel Temporal MLP (CCTM) block. As shown in the upper right of Figure 1, the proposed CCTM consists only of linear projections, ReLU activation, and transpose operations. When fed input as $\mathbf{V} \in \mathbb{R}^{N \times P \times d}$, CCTM first applies linear projection followed by a ReLU activation function in feature dimension d, and then applies linear projection in temporal dimension P,

$$\mathbf{V}' = \{[\mathrm{ReLU}(\mathbf{V}\mathbf{W}_{t_1} + \mathbf{b}_{t_1})]^\top \mathbf{W}_{t_2}\}^\top, \tag{2}$$

where $\mathbf{W}_{t_1} \in \mathbb{R}^{d \times d}$, $\mathbf{b}_{t_1} \in \mathbb{R}^d$ and $\mathbf{W}_{t_2} \in \mathbb{R}^{P \times Q}$ are learnable parameters, $\mathbf{V}' \in \mathbb{R}^{N \times Q \times d}$ is the output. Notably, the feature dimension is kept unchanged during the calculation of CCTM and there is no bias term in the linear projection of the temporal dimension. The proposed temporal MLP block provides a simple and effective way to model temporal dependencies.

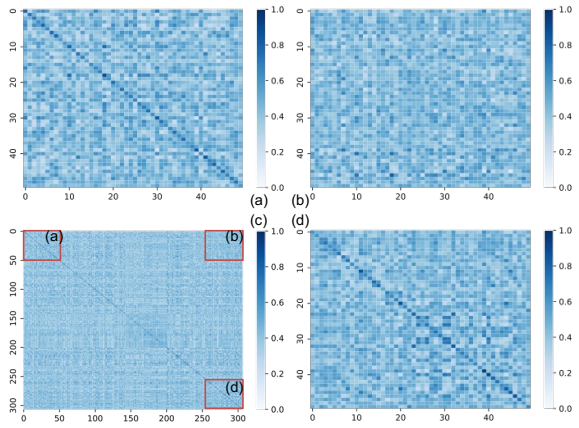

Figure 2: Learned adjacency matrix in [1] on PeMSD4 dataset. The bottom left subfigure shows the whole matrix, which has large values on the diagonal and small values elsewhere. The rest three subgraphs are taken from the upper left corner, lower right corner, and upper right corner of the whole matrix with a size of 50×50, respectively.

2.4 Graph-Less Spatial MLP Block

The Graph-Less Spatial MLP (GLSM) block is designed based on the observation that the learned adjacency matrices of several recent works share a simple pattern of having large values on the diagonal and small values elsewhere, as shown in Figure 2. The goal of the proposed GLSM is to approximate the graph convolution operation on adjacency matrices with such pattern, denoted as $\mathbf{A}_p$.

The detailed architecture of GLSM is shown on the left of Figure 1. GLSM consists only of a linear projection with a ReLU activation function and a layer normalization. Given input as $\mathbf{V} \in \mathbb{R}^{N \times Q \times d}$, GLSM first applies linear projection in feature dimension,

$$\mathbf{V}' = \mathrm{ReLU}(\mathbf{V}\mathbf{W}_{s_1} + \mathbf{b}_{s_1}), \tag{3}$$

where $\mathbf{W}_{s_1} \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_{s_1} \in \mathbb{R}^{d}$ are learnable parameters, $\mathbf{V}' \in \mathbb{R}^{N \times Q \times d}$ is the result of projection. Then a layer normalization is applied in the spatial and feature dimensions,

$$\hat{\mathbf{V}} = \frac{\mathbf{V}' - \mathrm{E}[\mathbf{V}']}{\sqrt{\mathrm{Var}[\mathbf{V}']}} \bigodot \mathbf{W}_{s_2} + \mathbf{B}_{s_2}, \tag{4}$$

where E and Var denotes the mean and standard deviation of all the elements in spatial and feature dimension of $\mathbf{V}'$, $\bigodot$ denotes element-wise product, $\mathbf{W}_{s_2} \in \mathbb{R}^{N \times 1 \times d}$ and $\mathbf{B}_{s_2} \in \mathbb{R}^{N \times 1 \times d}$ are learnable parameters, $\hat{\mathbf{V}}$ is the output of GLSM. As the calculation of mean, standard deviation, and element-wise multiplication all have linear complexity, the overall time and space complexity of the proposed GLSM is linear.

We here present how layer normalization along the space/node dimension can approximate the graph convolution operation on $\mathbf{A}_p$. For simplicity, we set the feature dimension to 1 and omit the bias term in layer normalization. Given an input $\mathbf{x} \in \mathbb{R}^{N \times 1}$, a layer normalization with affine parameters $\mathbf{a} \in \mathbb{R}^{N \times 1}$ on $\mathbf{x}$ is equal to the following multiplication:

$$\mathrm{LN}(\mathbf{x}) = [\frac{diag(\mathbf{a})}{\sigma}(\mathbf{I} - \frac{\mathbf{1}}{N})]\mathbf{x} = \mathbf{A}\mathbf{x}, \tag{5}$$

where σ is standard deviation of $\mathbf{x}$, $\mathbf{I} \in \mathbb{R}^{N \times N}$ is the identity matrix and $\frac{\mathbf{1}}{N} \in \mathbb{R}^{N \times N}$ is a matrix of all $\frac{1}{N}$s. As can be found, layer normalization is exactly equivalent to graph convolution with adjacency matrix as $\mathbf{A} = [\frac{diag(\mathbf{a})}{\sigma}(\mathbf{I} - \frac{\mathbf{1}}{N})]$. Because of the term $\mathbf{I} - \frac{\mathbf{1}}{N}$, $\mathbf{A}$ has a similar pattern as $\mathbf{A}_p$, *i.e.*, having large values on the diagonal and small values elsewhere. Due to the affine parameters $\mathbf{a}$, $\mathbf{A}$ has different values at different rows rather than $\frac{\mathbf{1}}{N}$ everywhere, which allows $\mathbf{A}$ to capture spatial correlations among nodes more flexibly.

Table 1: Statistical information of datasets.

Datasets	#Nodes	interval	Time Range
PeMSD3	358	5 min	09/01/2018 - 11/30/2018
PeMSD4	307	5 min	01/01/2018 - 02/28/2018
PeMSD7	883	5 min	05/01/2017 - 08/31/2017
PeMSD8	170	5 min	07/01/2016 - 08/31/2016

3 Experiments

3.1 Experimental Settings

3.1.1 Datasets. The experiments are conducted on four widely used real-world traffic datasets about highway traffic flow in California, namely PeMSD3, PeMSD4, PeMSD7, and PeMSD8. The statistical information is summarized in Table 1.

Data Preprocess: Linear interpolation is utilized to fill the missing values in the datasets. Then, we apply min-max normalization to normalize the data into the range of $[-1, 1]$ to make the training process more stable. As we focus on multi-step traffic forecasting, we set P and Q in Eq. 1 to 12, which means that each sample contains 24 points consecutively in the dataset which corresponds to two-hour data points, then the historical one-hour data points are taken as input and the next one-hour data points are taken as target. All datasets are divided into training sets, validation sets, and testing sets according to the ratio of 6:2:2 in chronological order.

3.1.2 Baselines. We compare the proposed GLMST model with several representative and advanced baseline models, including VAR [16], LSTM [5], STGCN [13], DCRNN [7], GraphWaveNet [11], ASTGCN(r) [4], AGCRN [1], STFGNN [6], and STG-NCDE [3].

3.1.3 Implementation. Our GLMST model is implemented in Python with PyTorch 1.9, under the environment with one CPU Intel(R) Xeon(R) E5-2620 v4 @ 2.10GHz and one GPU Nvidia Tesla V100 16GB. Mean Absolute Error (MAE) is chosen as the loss function, and Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) are involved to evaluate the prediction accuracy of all the models. We utilize Adam for tuning the learnable parameters with an initial learning rate of 0.0008 on PeMSD3, 0.003 on PeMSD4, 0.005 on PeMSD7, and 0.005 on PeMSD8, respectively.

3.2 Experimental Results

3.2.1 Overall Comparison. Table 2 and Table 3 demonstrate the averaged MAE, RMSE, and MAPE over 12-step prediction of GLMST

Table 2: Performance comparison on PeMSD3 and PeMSD4 datasets. The best results are highlighted in bold, and * indicates the second-best results, hereinafter the same.

Model	PEMSD3			PEMSD4		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
VAR	23.65	38.26	24.51%	24.54	38.61	17.24%
FC-LSTM	21.33	35.11	23.33%	26.77	40.65	18.23%
STGCN	17.55	30.42	17.34%	21.16	34.89	13.83%
DCRNN	17.99	30.31	18.34%	21.22	33.44	14.17%
GraphWaveNet	19.12	32.77	18.89%	24.89	39.66	17.29%
ASTGCN(r)	17.34	29.56	17.21%	22.93	35.22	16.56%
AGCRN	15.98	28.25	15.23%	19.83	32.26	12.97%
STFGNN	16.77	28.34	16.30%	20.48	32.51	16.77%
STG-NCDE	15.57*	27.09*	15.06%*	19.21*	**31.09**	12.76%*
GLMST	**15.21**	**26.77**	**14.74%**	**19.00**	31.20*	**12.46%**

Table 3: Performance comparison on PeMSD7 and PeMSD8.

Model	PeMSD7			PeMSD8		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
VAR	50.22	75.63	32.22%	19.19	29.81	13.10%
FC-LSTM	29.98	45.94	13.20%	23.09	35.17	14.99%
STGCN	25.33	39.34	11.21%	17.5	27.09	11.29%
DCRNN	25.22	38.61	11.82%	16.82	26.36	10.92%
GraphWaveNet	26.39	41.5	11.97%	18.28	30.05	12.15%
ASTGCN(r)	24.01	37.87	10.73%	18.25	28.06	11.64%
AGCRN	22.37	36.55	9.12%	15.95	25.22	10.09%
STFGNN	23.46	36.6	9.21%	16.94	26.25	10.60%
STG-NCDE	**20.53**	**33.84**	8.80%*	**15.45**	24.81*	9.92%*
GLMST	20.73*	34.60*	**8.76%**	15.48*	**24.69**	**9.78%**

and alternative baselines, which reveal that *our GLMST achieves competitive performance across all datasets.*

3.2.2 Exploratory Analysis of GLMST. To further analyze GLMST, we select three representative baselines that represent different calculating strategies of adjacency matrix, including (1) **STGCN** constructs distance-based adjacency matrix, (2) **AGCRN** learns embeddings for nodes and calculates the adjacency matrix according to the dot products of node embeddings, and (3) **ASTGCN** utilizes self-attention mechanism for calculating attention weights at each time step, and dot-products the attention weights with distance-based adjacency matrix which results in different adjacency matrices at each time step. Additionally, we involve STG-NCDE in this comparison, since STG-NCDE performs best in all baselines.

❶ **Time Consuming.** We compare the training and testing time consuming on PeMSD4 of our model and the selected four baselines. As shown in Table 4, the proposed GLMST has the least testing time cost among the five models and is 7× faster than STG-NCDE, which indicates *the improvement in time-consuming of our model is much greater than the performance improvement.* Due to the learnable parameters involved in layer normalization, the training speed of GLMST on PeMSD4 is slower than STGCN, AGCRN, and ASTGCN, but is still 5× faster than STG-NCDE.

❷ **Deploying GLSM on other models.** We deploy the proposed GLSM on STGCN, AGCRN, ASTGCN, and STG-NCDE to compare both the efficiency and effectiveness of GLSM and graph convolution on PeMSD4. Graph convolution in the original models is replaced with GLSM while keeping all other settings unchanged. As shown in Table 5, replacing graph convolution with GLSM leads to less time consumption both in training and testing procedures, which validates that GLSM has lower time complexity than graph convolution. Notably, when replacing the graph convolution with GLSM in the existing models, the prediction accuracy is nearly the same as that of the origin models or even better. Since the performance loss from such replacement of graph convolution is little, the proposed GLSM can be deployed as a plug-in spatial module to replace the graph convolution. *The experimental results validate the effectiveness of the proposed GLSM in modeling spatial dependencies.*

Table 4: Time consumption (s/epoch) of different models on PeMSD4. GLMST demonstrates the lowest testing time cost.

Model	MAE	Testing time	Training time
STGCN	21.16	2.88	**17.94**
AGCRN	19.83	2.44	21.72
ASTGCN	22.93	3.98	21.69
STG-NCDE	19.21	16.54	167.58
GLMST	**19.00**	**2.17**	33.21

Table 5: Effect of replacing graph convolution with GLSM. The replacement is nearly harmless to performance, and GLSM leads to faster training and testing on all models.

Model		MAE	RMSE	MAPE(%)	Testing time	Training time
STGCN	Origin	22.3	36.13	14.528	2.79	18.94
	With GLSM	21.51	35.25	14.13	2.74	18.67
ASTGCN	Origin	22.71	33.82	15.51	3.98	21.69
	With GLSM	20.51	32.84	14.00	0.87	7.56
AGCRN	Origin	19.74	32.16	13.14	2.44	21.72
	With GLSM	19.69	32.02	12.87	1.18	15.30
STG-NCDE	Origin	19.21	31.09	12.76	16.54	167.58
	With GLSM	19.32	31.19	12.78	13.90	142.69

❸ **Variants of GLSM.** As in Eq.5, multiplication with $\frac{1}{N}$ is equivalent to the mean operation on node dimension, thus, we present the following variants to further investigate GLSM. (1) **Mean** replaces layer normalization of GLSM with mean function, which is equal to directly use $(\mathbf{X} + mean(\mathbf{X}))\mathbf{W}$. (2) **MeanP** replaces layer normalization with mean function but keeps the affine parameters of layer normalization, which is designed to compare mean function and layer normalization. (3) **NoLNP** removes layer normalization in GLSM but keeps the affine parameters of layer normalization in Eq 4. (4) **NoRes** removes the residual connection of GLSM. (5) **LNNoP** removes the affine parameters of layer normalization. Results on PeMSD4 (Fig. 3) show that MeanP performs best among the variants but falls short of GLSM, highlighting the superiority of normalization. Affine parameters significantly enhance performance with both mean and layer normalization. The gap between NoRes and Origin GLSM underscores the importance of residual connections. The substantial performance drop in NoLNP demonstrates that, despite minimal spatial operations, layer normalization is crucial and has a greater impact than affine parameters.

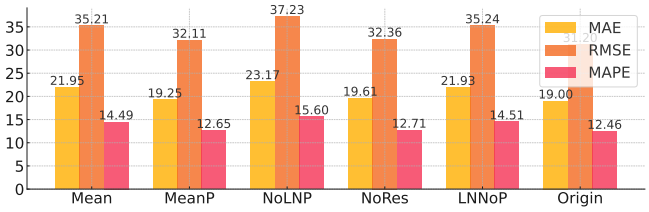

Figure 3: The performance of different variants of GLSM.

4 Conclusion

In this work, we propose a novel graph-less pure-MLP architecture (GLMST) for traffic forecasting, which consists of a Graph-Less Spatial MLP block (GLSM) and a Cross-Channel Temporal MLP block (CCTM). Experimental results on four real-world traffic flow datasets show the effectiveness of GLSM in capturing spatial dependencies and the superiority of our GLMST in traffic forecasting.

Acknowledgments

This paper is partially supported by the National Natural Science Foundation of China (No.12227901), the Project of Stable Support for Youth Team in Basic Research Field from the Chinese Academy of Sciences (No.YSBR-005), the Natural Science Foundation of Jiangsu Province (No.BK20240460), and the open fund of the State Key Laboratory of Resources and Environmental Information System.

References

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems* 33 (2020), 17804–17815.

[2] Jie Bao, Pan Liu, and Satish V Ukkusuri. 2019. A spatiotemporal deep learning approach for citywide short-term crash risk prediction with multi-source data. *Accident Analysis & Prevention* 122 (2019), 239–254.

[3] Jeongwhan Choi, Hwangyong Choi, Jeehyun Hwang, and Noseong Park. 2022. Graph Neural Controlled Differential Equations for Traffic Forecasting. (2022).

[4] 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.

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[6] Mengzhang Li and Zhanxing Zhu. 2021. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4189–4196.

[7] 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*.

[8] 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).

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[10] 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.

[11] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121* (2019).

[12] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. 2020. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908* (2020).

[13] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*.

[14] Yudong Zhang, Pengkun Wang, Binwu Wang, Xu Wang, Zhe Zhao, Zhengyang Zhou, Lei Bai, and Yang Wang. 2024. Adaptive and Interactive Multi-Level Spatio-Temporal Network for Traffic Forecasting. *IEEE Transactions on Intelligent Transportation Systems* 25, 10 (2024), 14070–14086.

[15] Yudong Zhang, Xu Wang, Pengkun Wang, Binwu Wang, Zhengyang Zhou, and Yang Wang. 2024. Modeling Spatio-Temporal Mobility Across Data Silos via Personalized Federated Learning. *IEEE Transactions on Mobile Computing* 23, 12 (2024), 15289–15306. doi:10.1109/TMC.2024.3453657

[16] Eric Zivot and Jiahui Wang. 2006. Vector autoregressive models for multivariate time series. *Modeling financial time series with S-PLUS®* (2006), 385–429.