MLPST: MLP is All You Need for Spatio-Temporal Prediction

Zijian Zhang*
Jilin University
City University of Hong Kong
zhangzj2114@mails.jlu.edu.cn

Ze Huang*
City University of Hong Kong
zehuang3-c@my.cityu.edu.hk

Zhiwei Hu*
City University of Hong Kong
zhiweihu4-c@my.cityu.edu.hk

Xiangyu Zhao†
City University of Hong Kong
xianzhao@cityu.edu.hk

Wanyu Wang
City University of Hong Kong
wanyuwang4-c@my.cityu.edu.hk

Zitao Liu
Guangdong Institute of Smart Education, Jinan University
liuzitao@jnu.edu.cn

Junbo Zhang
JD Intelligent Cities Research
JD iCity, JD Technology
msjunbozhang@outlook.com

S. Joe Qin
Lingnan University, Hong Kong
joeqin@LN.edu.hk

Hongwei Zhao
Jilin University
zhaohw@jlu.edu.cn

ABSTRACT

Traffic prediction is a typical spatio-temporal data mining task and has great significance to the public transportation system. Considering the demand for its grand application, we recognize key factors for an ideal spatio-temporal prediction method: efficient, lightweight, and effective. However, the current deep model-based spatio-temporal prediction solutions generally own intricate architectures with cumbersome optimization, which can hardly meet these expectations. To accomplish the above goals, we propose an intuitive and novel framework, MLPST, a pure multi-layer perceptron architecture for traffic prediction. Specifically, we first capture spatial relationships from both local and global receptive fields. Then, temporal dependencies in different intervals are comprehensively considered. Through compact and swift MLP processing, MLPST can well capture the spatial and temporal dependencies while requiring only linear computational complexity, as well as model parameters that are more than an order of magnitude lower than baselines. Extensive experiments validated the superior effectiveness and efficiency of MLPST against advanced baselines, and among models with optimal accuracy, MLPST achieves the best time and space efficiency.

CCS CONCEPTS

• **Information systems** → **Traffic analysis**; **Spatial-temporal systems**; • **Computing methodologies** → **Neural networks**.

*The three authors contributed equally to this research.
†Xiangyu Zhao is the corresponding author.

KEYWORDS

Spatio-Temporal Data Mining, Traffic Prediction, MLP-Mixer

ACM Reference Format:
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 (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614969

1 INTRODUCTION

In recent years, urbanization has experienced substantial growth, leading to an exponential increase in the number of vehicles on the roads. The availability of accurate traffic prediction information is crucial for traffic management as it serves as a foundation for informed decision-making and can help drivers to navigate roads with reduced congestion and accidents [14, 24, 26, 41, 43]. The advancements in sensor technologies, mobile devices, and Global Positioning Systems (GPS) have enabled the collection of a wide range of Spatio-Temporal (ST) data in urban areas, which in turn, has facilitated the application of Spatio-Temporal Data Mining (STDM) in urban life [40, 50–54]. STDM has widely fostered traffic prediction such as traffic flow analysis [3, 8, 56], travel time estimation [15, 37, 48], and traffic demand prediction [2, 11, 27].

Deep models have gained increasing popularity in recent years on STDM tasks. These models are known for their feature extraction and sequential modeling capacity, which enhances their effectiveness in extracting meaningful insights from ST data [1, 40]. Spatial maps are often represented and processed as image-like matrices, which has led to the widespread application of Convolutional Neural Networks (CNNs) [4, 18, 23, 33, 38] and Vision Transformers (ViT) [9, 13, 22, 30], which have been proven to be effective in capturing spatial features. On the other hand, for time series analysis, Recurrent Neural Networks (RNNs) [6, 7, 28] and self-attention mechanisms [10, 12, 17, 25] are commonly used to capture temporal correlations between different time slices.

Although prosperous urbanization has greatly facilitated our lives, various pressing problems emerge and need to be resolved.

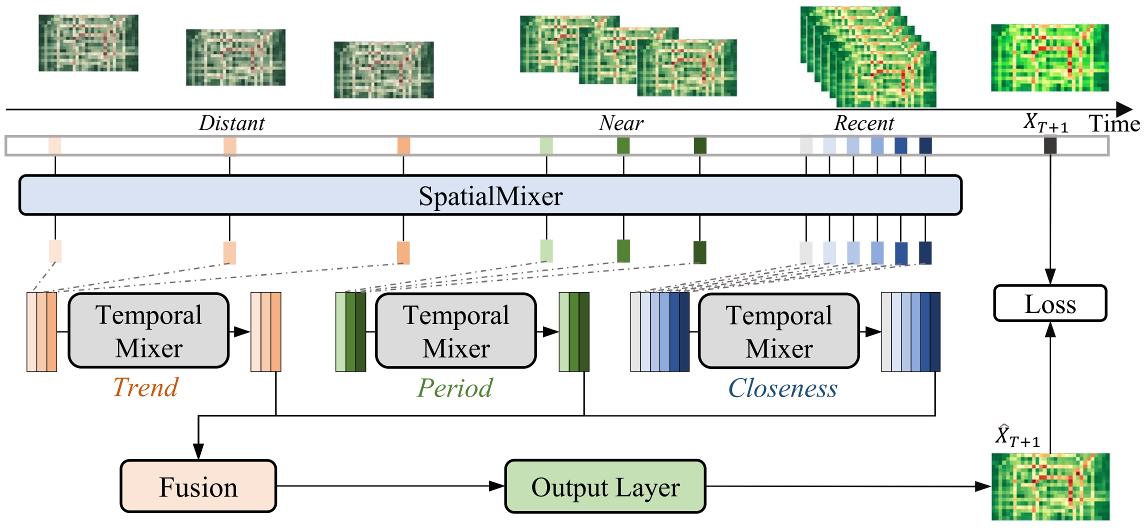

Figure 1: Framework overview for MLPST.

In consideration of the wide application of urban attribute prediction and the growing enormous quantity of data, we identify three crucial attributes for an effective spatio-temporal prediction method. (i) **Efficient.** Swift training on new attributes and data. (ii) **Lightweight.** Easy deployment and service with friendly storage requirements. (iii) **Effective.** Well-capture the spatial and temporal dependency and guarantee precision of prediction. However, current research can hardly meet the above ideal requirements. Though achieve promising performance on spatio-temporal prediction, they highly depend on the intricate techniques ensembling [19, 42], which takes overwhelming computational overhead and deployment cost [20, 29].

To achieve the above targets, we propose an intuitive and novel framework, MLPST, which is made up of pure Multi-Layer Perceptrons (MLP). By using interleaved MLPs across certain dimensions, we can mix information from different parts of the input data. In this framework, we model spatial dependencies by patch processing the spatial map and conduct SpatialMixer between patches to learn global correlation within this spatial map, thus achieving a global receptive field. Besides, we define temporal dependencies to be three parts: closeness, period, and trend, so the temporal dependencies from different time intervals can be well modeled by the TemporalMixer. Benefiting from the simple and efficient MLP, our MLPST enjoys a rather low computational complexity, *i.e.*, $O(N)$. To achieve a fair comparison, we verify the performance of MLPST on real-world traffic flow datasets through an open-source platform LibCity [39]. Extensive experiments prove the effectiveness of our framework. Results consistently show its superior performance compared to state-of-the-art methods. In a nutshell, the major contributions of our work can be summarized as follows:

- We propose an all-MLP framework for ST traffic prediction, MLPST. It consists of SpatialMixer and TemporalMixer, which are separately responsible for global spatial dependencies and temporal variations from different spans;
- Our MLPST owns linear computational complexity and remarkably reduced parameters compared to existing methods, which indicates its promising potential for practical application;
- We conduct extensive experiments to demonstrate the effectiveness of MLPST on two real-world datasets, which shows its superiority over existing baseline methods. We also present comprehensive analyzes to verify the validity of each component.

2 FRAMEWORK

In this section, we will present an all-MLP architecture, MLPST, which can solve STDM tasks in an effective way. We will start by introducing the overall architecture of our model and then elaborate on different modules in detail. Finally, we will provide the optimization process.

2.1 Preliminaries

In this subsection, we first present the task definition and define the essential notations.

Definition 1 (Traffic Flow). We split the city into $H \times W$ non-overlapping equal-sized grids based on longitude and latitude. Based on the traffic flow data in a certain period, we can calculate the traffic inflow and outflow of all grids. The traffic flow of all grids of a certain period i is called a traffic flow grid map, noted as a matrix $\mathbf{X}_i \in \mathbb{R}^{H\times W\times d}$, where $d = 2$ represents traffic inflow and outflow.

Definition 2 (Patch). A patch is defined as a $P \times P$ sized part of a grid map $\mathbf{X}_i$. Patches are non-overlapping parts of the original grid map. Thus, a grid map $\mathbf{X}_i$ is partitioned into $N_P = HW/P^2$ patches.

Definition 3 (Spatio-Temporal Traffic Prediction). Given the historical observations $\{\mathbf{X}_i | i = 1, 2, \dots, T\}$, the goal of spatio-temporal traffic prediction is to predict the future traffic state $\mathbf{X}_{T+1}$, and $f(\cdot)$ stands for the spatio-temporal prediction model:

$$\mathbf{X}_{T+1} = f([\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_T]) \tag{1}$$

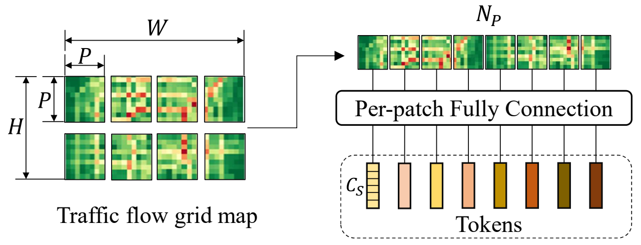

Figure 2: Patch processing.

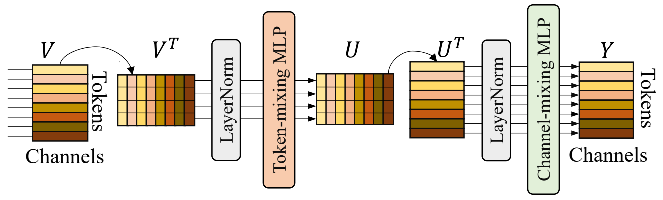

Figure 3: Visualization of all-MLP MixerLayer.

2.2 Framework Overview

In this subsection, we present an overview of our framework. MLPST is a novel and efficient framework based on an all-MLP architecture, which mainly consists of several modules. First, we propose two MLP-based modules, named SpatialMixer and TemporalMixer, to capture spatial and temporal dependency, respectively. Besides, we incorporate a fusion module to fuse the representation of different temporal dependencies. Finally, we employ an output layer to attain the prediction of the future traffic state.

We introduce the pipeline as visualized in Figure 1 from top to bottom. Firstly we recognize the historical data sequence with three different kinds of temporal dependencies: *trend*, *period*, and *closeness*. We address the spatial dependency by feeding SpatialMixer with the input feature of each time step, and obtain corresponding feature embeddings in a global view. Then, we incorporate three TemporalMixers to capture the temporal relationship inside the three temporal dependencies, respectively. Finally, we aggregate the output embeddings of different temporal dependencies through a fusion module. The output layer further maps the fusion result to a $H \times W$ sized grid map to accomplish the single-step prediction.

2.3 Detailed Modules

2.3.1 Input Layer. As aforementioned, the input layer is comprised of three fragments on the time axis denoting the observations from distant history, near history, and recent time. As illustrated in Figure 1, these fragments are used to model three types of temporal dependencies: *trend*, *period*, and *closeness*, which are brought up based on sequencing observations with three kinds of intervals. Intuitively, the interval for trend series is the largest, which shows the development of data in a relatively long time, like seasonal changes. Period stands for the periodic variation of data in a fixed time period, for instance, traffic flow data often changes in one day cycle. Closeness means ST data is affected by the situation of recent time intervals. For example, if there is traffic congestion at 8 a.m., then the volume of vehicles at 9 a.m. is most likely to be high because of the closeness dependency.

Concretely Speaking, let l_t, l_p and l_c stand for the intervals of trend, period and closeness dependent sequences with the length of t, p and c. Then the trend dependent sequence can be represented as a subset of all historical observations $[\mathbf{X}_{T-t\cdot l_t}, \ldots, \mathbf{X}_{T-l_t}]$. Likewise, the period and closeness dependent sequences can be defined as $[\mathbf{X}_{T-p\cdot l_p}, \ldots, \mathbf{X}_{T-l_p}]$, and $[\mathbf{X}_{T-c\cdot l_c}, \ldots, \mathbf{X}_{T-l_c}]$, where we assert $t + p + c = T$ so as to fix the input window. The usual situation is that $l_t > l_p > l_c$. In this paper, we assign the temporal dependencies as follows: set the trend span l_t to be one week which reveals the weekly trend; set l_p to be a one-day period that describes the daily periodicity; set l_c to be the unit interval to model the recent variation of the traffic flow.

2.3.2 SpatialMixer. Modeling the region dependency comprehensively is not trivial, and we explore to tackle from both global and local perspectives. We propose an all-MLP SpatialMixer to model the spatial dependency in a linear complexity w.r.t. N_P. Specifically, in the SpatialMixer module, the spatial information of the input grid data is extracted and mixed to a certain-sized embedding $\mathbf{E}_i$. The dimension of $\mathbf{E}_i$ is related to the patch number N_P and the channel number C_S, where *channel* is defined as the hidden dimension of SpatialMixer for feature representation. SpatialMixer is comprised of the following blocks: Patch processing and All-MLP MixerLayers.

Patch Processing. SpatialMixer starts with the patch processing step shown in Figure 2, including *patch division* and *per-patch fully connection*. Specifically, we first partition the $H \times W$ grid map into non-overlapping $P \times P$ sized patches. Each patch contains P^2 grids, so the flattened tensor belongs to $\mathbb{R}^{1\times P^2}$. Assume the number of feature channels here is C_S, then we map the sequence of patches from $\mathbb{R}^{1\times P^2}$ to $\mathbb{R}^{1\times C_S}$ through per-patch fully connection. Every patch is mapped to a token, and all tokens are concatenated to be a two-dimension input matrix with the size of $N_P \times C_S$.

All-MLP MixerLayer. We propose to capture the global correlations via an all-MLP MixerLayer [34]. We denote the output of patch processing as $\mathbf{V}$, which serves as the input matrix for MixerLayers. As demonstrated in Figure 3, MixerLayer performs LayerNorm, Token-mixing MLP, LayerNorm, and Channel-mixing MLP successively. The Token-mixing MLP is fomulated as:

$$\mathbf{U} = \mathbf{V}^T + \mathbf{W}_2\sigma\left(\mathbf{W}_1 \operatorname{LayerNorm}(\mathbf{V}^T) + \mathbf{b}_1\right) + \mathbf{b}_2 \tag{2}$$

where σ is a GELU activation, $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1$, and $\mathbf{b}_2$ are learnable weight matrices and bias.

Token-mixing MLP is an MLP block that takes columns of matrix $\mathbf{V}$ as input, *i.e.,* the rows of the transposed matrix $\mathbf{V}^T$ as Figure 4 illustrated. The MLP block maps $\mathbf{V}^T \in \mathbb{R}^{C_S\times N_P}$ to $\mathbf{U} \in \mathbb{R}^{C_S\times N_P}$.

The output of Token-mixing MLP $\mathbf{U}^T$ is fed to Channel-mixing MLP. As shown in Equation (3), Channel-mixing MLP maps $\mathbf{U}^T \in \mathbb{R}^{N_P\times C_S}$ to $\mathbf{Y} \in \mathbb{R}^{N_P\times C_S}$, which is the output of the MixerLayer.

$$\mathbf{Y} = \mathbf{U}^T + \mathbf{W}_4\sigma\left(\mathbf{W}_3 \operatorname{LayerNorm}(\mathbf{U}^T) + \mathbf{b}_3\right) + \mathbf{b}_4 \tag{3}$$

where $\mathbf{W}_3, \mathbf{W}_4, \mathbf{b}_3$, and $\mathbf{b}_4$ are the learnable MLP parameters.

The fully-connected operations within Token-mixing MLPs and Channel-mixing MLPs enable the interaction between different

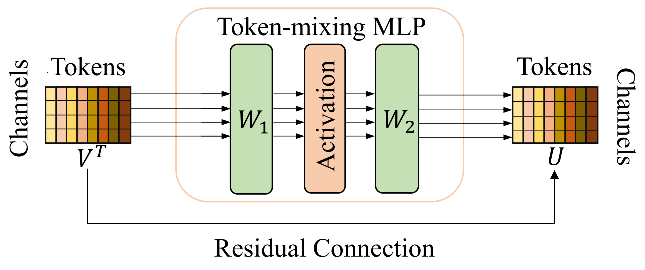

Figure 4: Illustration of Token-mixing MLP architecture.

spatial locations, *i.e.,* tokens, and channels. To conclude, MixerLayer maps $\mathbf{V}^T \in \mathbb{R}^{C_S \times N_P}$ to $\mathbf{Y} \in \mathbb{R}^{C_S \times N_P}$, which means the dimension of the input tensor does not change in MixerLayer. But at the same time, cross-location and per-location information can be captured with the composition of the Token-mixing and Channel-mixing. It is noteworthy that since tokens come from different locations on the original spatial map, then communication between tokens realizes the global mixing of spatial information, which effectively helps the feature extraction of spatial dependency. Besides, since C_S is selected independently of N_P, the complexity of this part should be $O(N_P)$, unlike $O(N_P^2)$ of self-attention mechanisms.

After the processing of N MixerLayers, for the convenience of following computation and interpretation, we formulate the output tensor to $\mathbf{E}_i \in \mathbb{R}^{1 \times N_P C_S}$ as the final output of SpatialMixer for time span i. Equation (4) represents the feature mapping process of SpatialMixer, where $\mathbf{E}_i$ denotes its output.

$$\mathbf{E}_i = \text{SpatialMixer}(\mathbf{X}_i), \forall i = 1, 2, ..., T \tag{4}$$

2.3.3 TemporalMixer. The time complexity is an inherent issue of temporal data mining, such as $O(T \cdot d^2)$ of RNN and $O(T^2 \cdot d)$ of self-attention, where T represents the number of input time steps and d is the embedding size. Therefore, we propose to model the temporal dependencies via TemporalMixer consisting of a series of MixerLayers, and possesses a reduced complexity of $O(T \cdot d)$.

As mentioned before, we adopted three kinds of representations to model different temporal dependencies from the global view, *i.e., trend, period, closeness.* The embeddings $\mathbf{E}_i$ obtained by SpatialMixer of the same temporal dependency are concatenated as the input of a TemporalMixer. Take the *trend* dependent sequence as example, number of tokens is the sequence length t, and the concatenated input matrix for the TemporalMixer is $\mathbf{E}_t \in \mathbb{R}^{t \times N_P C_S}$, as shown in Equation (5). Note that the internal architecture of TemporalMixer is identical with that of SpatialMixer, visualized in Figure 4. Assume that the number of channels for TemporalMixer is C_T, which is the hidden dimension for its All-MLP MixerLayer. Then through Token-mixing and Channel-mixing, information in different parts of $\mathbf{E}_t$ interact with each other, *i.e.,* the trend dependency of certain locations can be captured.

$$\mathbf{E}_t = \text{Concat}(\mathbf{E}_i), \forall i = 1, 2, \ldots, t \tag{5}$$

Similarly, dimension of input tensor does not change through TemporalMixer. In other word, an All-MLP MixerLayer is utilized to capture temporal dependencies, which maps $\mathbf{E}_t \in \mathbb{R}^{t \times N_P C_S}$ to $\widehat{\mathbf{E}}_t \in \mathbb{R}^{t \times N_P C_S}$ as expressed by Equation (6). The complexity here is linear with respect to the number of tokens, which is $O(t \cdot d_T)$, where $d_T = N_P C_S$ is the embedding size for each time step.

$$\widehat{\mathbf{E}}_t = \text{TemporalMixer}(\mathbf{E}_t) \tag{6}$$

Likewise, the same operations can be applied on the view of *period* and *closeness* dependencies, thus obtaining $\widehat{\mathbf{E}}_p$ and $\widehat{\mathbf{E}}_c$. The overall complexity of three parts is $O(T \cdot d_T)$ when $t + p + c = T$.

2.3.4 Fusion. In the Fusion module, we take the last time step of $\widehat{\mathbf{E}}_t$, $\widehat{\mathbf{E}}_p$ and $\widehat{\mathbf{E}}_c$ as mixing results of the three temporal dependencies. They are aggregated by assigning three corresponding learnable weights to different parts. Fusion module can be formulated by :

$$\widehat{\mathbf{E}} = \mathbf{W}_t \circ \widehat{\mathbf{E}}_t[\mathrm{t}] + \mathbf{W}_p \circ \widehat{\mathbf{E}}_p[\mathrm{p}] + \mathbf{W}_c \circ \widehat{\mathbf{E}}_c[\mathrm{c}] \tag{7}$$

where $\circ$ means dot product, $[\cdot]$ denotes the index of the time dimension, and $\mathbf{W}_t, \mathbf{W}_p, \mathbf{W}_c$ are the learnable parameters that adjust the impact of trend, period, closeness.

2.3.5 Output Layer. Output Layer is composed of a conventional MLP to map the embedding obtained by TemporalMixer to the size of the input grid, *i.e.,* Output Layer maps $\widehat{\mathbf{E}}$ to $\widehat{\mathbf{X}}_{T+1} \in \mathbb{R}^{H \times W}$, which achieves the single-step prediction and finishes the whole predictive learning pipeline.

$$\widehat{\mathbf{X}}_{T+1} = \mathbf{W}_o \circ \widehat{\mathbf{E}} + \mathbf{b}_o \tag{8}$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ are the weight and bias of output layer.

2.4 Optimization

In this section, we detail our loss function that is used for training. The loss function can be defined as:

$$L(\theta) = \Big(\sum_{i=1}^{M} |\widehat{\mathbf{X}}_{T+1} - \mathbf{X}_{T+1}|^q\Big)^{1/q} \tag{9}$$

where θ stands for the learnable parameters, and M is the number of data records. q indicates the regularization norm of loss function, such as $q = 1$ of Mean Average Error (MAE) and $q = 2$ of Root Mean Square Error (RMSE).

2.4.1 Framework Complexity. Since MLPST is an all-MLP architecture, the computational complexity of each individual module can be denoted as matrix multiplications.

SpatialMixer SpatialMixer starts feature extraction from patch processing with an output matrix $\mathbf{V} \in \mathbb{R}^{N_P \times C_S}$, and $\mathbf{V}$ serves as the input of all-MLP MixerLayers. Then the computational complexity of this MLP architecture can be denoted as several matrix multiplications, *i.e.,* $i \times h + h \times o$, where i is the number of input units, h is the number of hidden units, o is the number of output units. As mentioned in the Framework section, MixerLayer consists of Token-mixing MLPs and Channel-mixing MLPs. The hidden unit of these MLPs is an expansion ratio that is independent of input units and output units, and the input units number equals the output units. So for Token-mixing MLPs, the complexity is $O(N_P)$ with N_P being the number of input units. Likewise, the complexity for Channel-mixing is $O(C_S)$. Note that C_S is chosen independent of the number of input tokens N_P, therefore, the complexity for a MixerLayer is $O(N_P)$.

TemporalMixer Structure of MixerLayers within TemporalMixer is identical with that of SpatialMixer. Similarly, for TemporalMixers that process different temporal dependencies, we can obtain each

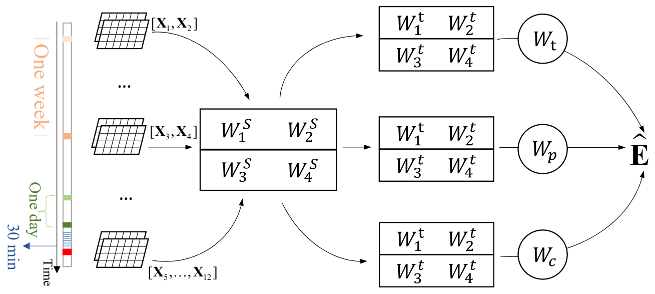
One week
One day
30 min
Time
$[\mathbf{X}_1, \mathbf{X}_2]$
$[\mathbf{X}_3, \mathbf{X}_4]$
$[\mathbf{X}_5, \ldots, \mathbf{X}_{12}]$
W_1^S W_2^S W_3^S W_4^S
W_1^t W_2^t W_3^t W_4^t
W_t
W_p
W_c
$\widehat{\mathbf{E}}$

Figure 5: Traffic flow prediction.

of their complexity. As for trend dependency, the input matrix is $\mathbf{E}_t \in \mathbb{R}^{t \times d_T}$. The number of input units for Token-mixing MLPs is the input sequence length t, so the complexity is $O(t)$ accordingly. Likewise, we have $O(d_T)$ for Channel-mixing MLPs. So the complexity of trend-dependent TemporalMixer is $O(t \cdot d_T)$. Similarly, we can get $O(p \cdot d_T)$ for period dependency and $O(c \cdot d_T)$ for closeness dependency. The add-up result of three parts should be $O(T \cdot d_T)$ when we assert $t + p + c = T$.

Based on the above deduction, the overall computational complexity for MLPST is $O(N_P \cdot T \cdot d_T)$.

2.5 Inference

We present the detailed inference procedure and a toy example for traffic flow prediction in an example. Specifically, we aim to make the single-step prediction with twelve input time steps of the traffic flow grid map $\{\mathbf{X}_i | i = 1, 2, \ldots, 12\}$, $\mathbf{X}_i \in \mathbb{R}^{H \times W \times 2}$, denoting the grid map size $H \times W$ and the feature dimension 2 which means inflow and outflow respectively. Furthermore, we divide $[\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{12}]$ into $[[\mathbf{X}_1, \mathbf{X}_2], [\mathbf{X}_3, \mathbf{X}_4], [\mathbf{X}_5, \ldots, \mathbf{X}_{12}]]$, which denotes three types of temporal dependent sequences $[[trend], [period], [closeness]]$. The learning process is performed by Token-mixing MLPs and Channel-mixing MLPs as pre-defined in Equation (10) and (11):

$$\mathbf{U} = \mathbf{V}^T + \mathbf{W}_2 \sigma \left(\mathbf{W}_1 \operatorname{LayerNorm}(\mathbf{V}^T) \right) \tag{10}$$

$$\mathbf{Y} = \mathbf{U}^T + \mathbf{W}_4 \sigma \left(\mathbf{W}_3 \operatorname{LayerNorm}(\mathbf{U}^T) \right) \tag{11}$$

where V_T is a input matrix for MixerLayers. For SpatialMixer, we assume that $\mathbf{W}_1^S, \mathbf{W}_2^S, \mathbf{W}_3^S, \mathbf{W}_4^S$ are the well learned weights. For TemporalMixer, we assume that $\mathbf{W}_1^t, \mathbf{W}_2^t, \mathbf{W}_3^t, \mathbf{W}_4^t$ are the learned weights for trend dependency, and similarly we have $\mathbf{W}_1^p, \mathbf{W}_2^p, \mathbf{W}_3^p, \mathbf{W}_4^p$ and $\mathbf{W}_1^c, \mathbf{W}_2^c, \mathbf{W}_3^c, \mathbf{W}_4^c$ for period dependency and closeness dependency respectively. At last, the traffic flow feature embedding of next time slot can be predicted as:

$$\widehat{\mathbf{E}} = \mathbf{W}_t \circ \widehat{\mathbf{E}}_t[\mathrm{t}] + \mathbf{W}_p \circ \widehat{\mathbf{E}}_p[\mathrm{p}] + \mathbf{W}_c \circ \widehat{\mathbf{E}}_c[\mathrm{c}] \tag{12}$$

where $\mathbf{W}_t, \mathbf{W}_p, \mathbf{W}_c$ are another three learned weights for adjusting different temporal dependencies. Figure 5 represents how we make predictions with the given learned weights using Equation (10) and Equation (11). Finally, the output layer maps $\widehat{\mathbf{E}}$ to the original input size to accomplish the single-step prediction.

3 EXPERIMENT

In this section, we conduct extensive experiments on two real-world datasets to evaluate the effectiveness of MLPST.

3.1 Experimental Setting

3.1.1 Datasets. We conduct experiments on two real-world datasets from the New York City OpenData[1], *i.e.,* NYCBike and NYCTaxi, which contain records of taxi and bike orders in New York City.

NYCBike: The NYCBike dataset contains bicycle trajectories collected from the NYC CitiBike system. The transaction records between July 1st, 2016, and August 31st, 2016 are selected. The following information is contained: bike pick-up station, bike drop-off station, bike pick-up time, bike drop-off time, and trip duration.

NYCTaxi: The NYCTaxi dataset contains tracks of different types of cabs collected by GPS for New York City from 2009 to 2020. We pick data from January 1st, 2015, to March 31st, 2015. It provides properties including start and arrival time, geological information of start and end points, and trip distance.

3.1.2 Evaluation Metrics. To evaluate the effectiveness of our framework on traffic flow prediction, we adopt some commonly used value-based metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Coefficient of Determination (R^2).

3.1.3 Implementation. We implement MLPST based on a public library, LibCity [39], which contains several of the most representative traffic flow prediction models.

For the SpatialMixer part, (a) *Per-patch Fully Connection*: first, we divide the city region into 10×20 disjoint grids of equal size. Then we set the patch size $P \times P$ to be 2×2. According to the input data size, the number of channels C_S is set to 20, which is the mapped dimension size of per-patch FC. (b) *MixerLayer*: we set the initial depth of MixerLayer to 8. As mentioned before, the number of patches N_P can be expressed as $HW/P^2 = 50$, and the number of channels C_S is known to be 20. The hidden parameter expansion is set to be 8, which is the number of hidden units in MLPs that are used in Token-Mixing and Channel-Mixing.

For the TemporalMixer part, (a) *Input layer setting*: it is made up of N MixerLayers, where the mixer depth N is 8, and the number of channels for TemporalMixer C_T is set to 20. We divide the input twelve time steps data into closeness, period, and trend. The initial values are 8, 2, and 2, respectively.

For other settings, (a) *Optimization*: MLPST is implemented with Pytorch. The training process uses the Adam optimizer with the learning rate of 1×10^{-3} and batch size of 64. We use cross-validation and early stop strategy. We utilize both MAE and RMSE error in our objective function, *i.e.,* $q = 1, 2$. (b) We use Ray tune to optimize hyperparameters for our model. Most of the experiments for the baseline are performed with the released code in LibCity. To make the baseline results comparable, we adjust the number of input steps and do parameter tuning for all baselines to maintain uniformity. Each baseline model is trained five times to obtain an average result.

3.1.4 Baseline. We compare the proposed MLPST with the following state-of-the-art baseline models:

[1]https://opendata.cityofnewyork.us/

Table 1: Overall performance comparison of all methods on two datasets.

Datasets / Metrics / Methods	NYC Bike MAE ↓	NYC Bike RMSE ↓	NYC Bike R2 ↑	NYC Taxi MAE ↓	NYC Taxi RMSE ↓	NYC Taxi R2 ↑	Parameters(K)
AutoEncoder	5.57	8.27	0.687	22.45	65.77	0.626	4.89×10^2
ResLSTM	3.44	10.72	0.472	19.74	59.86	0.715	1.74×10^4
Conv-GCN	5.19	7.71	0.728	11.80	20.58	0.966	3.48×10^3
Multi-STGCnet	1.41	4.95	0.885	9.31	42.13	0.844	1.81×10^4
ST-ResNet	<u>1.23</u>	3.37	0.946	8.60	63.80	0.631	1.80×10^3
FC-RNN	5.72	7.99	0.707	8.13	18.08	0.971	0.69×10^2
Seq2Seq	4.93	6.83	0.786	7.78	17.74	0.972	2.43×10^2
ACFM	1.26	<u>3.25</u>	<u>0.950</u>	<u>5.37</u>	**11.58**	<u>0.987</u>	3.39×10^2
MTGNN	1.72	5.28	0.876	7.25	19.95	0.965	4.40×10^2
MLPST	**1.20***	**3.17***	**0.954***	**4.76***	<u>11.60</u>	**0.988***	**0.60**$\times 10^2$

"*" indicates MLPST's statistically significant improvements (i.e., two-sided t-test with $p < 0.05$) over the best baseline.

- **AutoEncoder** [16]: it uses an encoder to learn the embedding vector from the data and then uses a decoder to predict future traffic conditions.
- **ResLSTM** [45]: it is a deep learning architecture merging the residual network, graph convolutional network, and long short-term memory to predict urban rail short-term passenger flow.
- **Conv-GCN** [46]: it combines a multi-graph convolutional network to address multiple spatio-temporal variations (*i.e.,* recent, daily, and weekly) separately. Then it uses a 3D convolutional neural network to integrate the inflow and outflow information deeply.
- **Multi-STGCnet** [44]: it contains three long short-term memory-based modules as temporal components and three spatial matrices as spatial components for extracting spatial associations of target sites.
- **ST-ResNet** [47]: it uses a residual neural network framework to model temporal closeness, period, and trend properties of crowd traffic, which is widely used in grid-based traffic prediction tasks.
- **FC-RNN** [39]: a swift deep learning model consisting of a fully connected layer and recurrent neural network for spatio-temporal prediction.
- **Seq2Seq** [31]: it is an encoder-decoder architecture based on gated cyclic units to predict the traffic state.
- **ACFM** [21]: Attention Crowd Flow Machine (ACFM) is capable of inferring the evolution of crowd flows by learning dynamic representations of data with temporal variation through an attention mechanism.
- **MTGNN** [42]: MTGNN solves multivariate time series prediction based on graph neural networks.

3.2 Overall Performance

Table 1 shows the overall performance of MLPST and the baseline models on the datasets NYCBike and NYCTaxi.

- As observed, the results of AutoEncoder on both datasets indicate that simply compressing and reconstructing the input data cannot extract useful features.
- In contrast, complex deep learning models such as ResLSTM, Conv-GCN, and Multi-STGCnet get better results. However, there are differences in their performance results on the two datasets. These three models employ GCN to model the spatial dependencies between target stations. It shows that capturing the spatial dependencies within the whole network is essential. Both ResLSTM and Multi-STGCnet use LSTM to learn temporal correlations. So extracting features in the temporal dimension allows for better short-term passenger flow prediction. From both spatial and temporal perspectives, we use modules composed of MLP-Mixer architectures in our framework MLPST, instead of complex deep learning structures like the three methods above. It is noteworthy that we obtain better results with our framework.
- ST-ResNet and Conv-GCN use different methods to model temporal closeness, period, and trend. ST-ResNet designs residual convolution units for each division and aggregates them dynamically. In our model, we similarly divide time steps into temporal closeness, period, and trend, but we apply MLP-mixer to each one and then fuse them by learnable weights. A reasonable division of input time steps improves traffic flow prediction accuracy.
- FC-RNN and Seq2seq process sequence data and learn temporal information. ACFM achieves spatial weight prediction using two ConvLSTMs units, where one LSTM learns an effective spatial-temporal feature representation.
- Our MLPST achieves consistently leading performance across all the metrics, which proves its advancing capability in modeling spatial and temporal dependencies. In addition to the superior prediction performance of MLPST, another advantage over the baseline is the relatively simple model structure and the small number of parameters. By sharing parameters between different layers, our MLPST owns the lowest amount of trainable parameters, *i.e.,* 0.60×10^2, which is lower by more than one order of magnitude than baseline methods.

To summarize, the above overall experimental performance demonstrates the effectiveness and efficiency of MLPST against representative baselines, which validates its superiority in spatio-temporal prediction tasks. MLPST uses a simple all-MLP structure for traffic flow prediction in both the spatial and temporal dimensions. Moreover, we divide time steps in the temporal dimension. MLPST accomplishes complex tasks with a simple structure and performs well beyond the baseline.

Table 2: Efficiency study of MLPST.

Dataset	**NYCTaxi**			**NYCBike**		
Methods	MAE	Train (s)	Infer (ms)	MAE	Train (s)	Infer (ms)
AutoEncoder	22.45	1,573	47	5.57	1,443	62
ResLSTM	19.74	1,483	1,992	3.44	1,167	1,932
Conv-GCN	11.80	297	160	5.19	157	150
Multi-STGCnet	9.31	943	588	1.41	591	380
ST-ResNet	8.60	4,208	1,613	1.23	6,845	1,781
FC-RNN	8.13	31	71	5.72	28	76
Seq2Seq	7.78	36	61	4.93	42	66
ACFM	5.37	4,775	2,319	1.26	5,843	2,891
MLPST	**4.76**	132	38	**1.20**	93	39

Infer time is averaged for one batch (batch size = 64)

3.3 Efficiency Analysis

In the context of swift urbanization, spatio-temporal model capacity, efficiency, and lightweight all play profound roles. This section surveys the efficiency of ST methods on the two datasets. To achieve a fair comparison, we conduct all experiments on one NVIDIA MX330 GPU. We follow the original settings for the baseline methods.

Table 2 shows the comparison results. For results on NYCTaxi, we observe that AutoEncoder, FC-RNN, and Seq2Seq obtain poor results for all three models despite their short training time and inference time due to their simple structures. The best-performing baseline ST-ResNet requires a significant amount of training time to improve its accuracy. Meanwhile, our MLPST achieves better results than ST-ResNet with only about 3% of the training time. On NYCBike, we can observe that FC-RNN and Seq2Seq have very short training time and inference time because of their relatively simple structures, but these two models obtain inferior results to our MLPST. The best-performing baseline ACFM takes a lot of time on training and inference to improve its accuracy. We achieve superior results than ACFM, with only about 1% of the training time and the inference time.

The advanced performance, as well as training and inference efficiency of MLPST, can be ascribed to its simple structure and the incorporation of MLP-Mixer in both spatial and temporal dimensions. Therefore, MLPST obtains a better balance between time efficiency and performance.

3.4 Ablation Study

In this subsection, we will examine the effectiveness of each component of the model’s architecture. MLPST mainly consists of two key components, *i.e.,* the SpatialMixer and the TemporalMixer. We remove our model’s SpatialMixer and TemporalMixer architectures, respectively. Then we obtain two variants of the original framework, *i.e.,* MLP-AT and MLP-SA. By comparing the performance of adjusted models above with MLPST, we can acquire insight into the effectiveness of these two modules. In the following experiments, we implement these two variants of MLPST on the NYCtaxi dataset:

- **MLP-AT**: We remove the SpatialMixer, and keep the TemporalMixer unchanged for temporal feature learning.
- **MLP-SA**: We remove the TemporalMixer, and keep the SpatialMixer unchanged for spatial feature learning.

As shown in Figure 6, the MAE obtained by MLP-AT is 7.17. Model performance decreases by about 50% compared to MLPST, indicating that using the MLP-Mixer architecture in the spatial dimension effectively captures the spatial correlation between different grids. MLP-SA worsens the model performance with the MAE increased by 3.43 because of removing TemporalMixer. We observe that the model’s prediction accuracy decreases sharply when the TemporalMixer is replaced. This result indicates that the MLP-Mixer architecture used by MLPST in the temporal dimension effectively captures valuable regions at each time step. Based on the experiments, it is obvious that each part of MLPST contributes as expected to the model.

3.5 Temporal Dependency Study

In this section, we analyze the time step division in detail for the temporal dependency study to learn the influence of our MLPST on temporal dependency capture.

On the input side, we take a set of twelve time steps as input and output the predicted value of one time step. We select the critical time steps for modeling by dividing the twelve time steps into three groups: closeness, period, and trend. We fix the other parameters and make different adjustments to the division of the input data. For ease of representation, we use (closeness, period, trend) as a triplet. As shown in Figure 7, there are six groups of input data from a to f, which are (12, 0, 0), (10, 2, 0), (10, 0, 2), (8, 2, 2), (8, 4, 0), and (8, 0, 4). It does not make sense if the period is one or the trend is one because we need to do Token-mixing and Channel-mixing between embeddings in our model.

From Figure 7, We can see that the input (10, 2, 0) gives slightly lower results compared to the input (12, 0, 0), and obtains the best result, which proves that the introduction of period benefits the model with the temporal dependency capture. However, because the results for input (8, 4, 0) dropped, it also indicates that introducing too long a period may not be useful or may be difficult to capture the information in it. The results for input (10, 0, 2) are worse than those for input (12, 0, 0) and (10, 2, 0). Finally, the results of input (8, 2, 2) and (8, 0, 4) do not work well, suggesting that the key closeness is still more helpful, which is in line with our perception of prediction in the spatial-temporal domain.

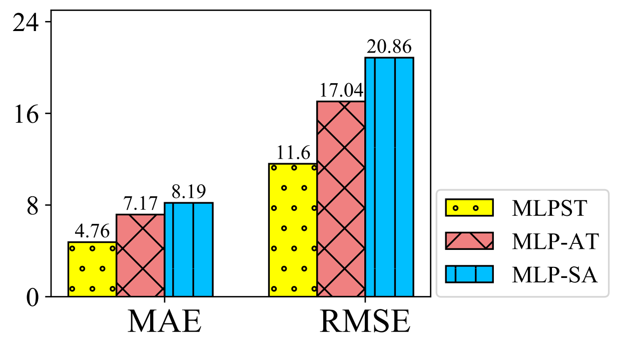

Figure 6: Results of ablation study on NYCTaxi dataset.

3.6 Hyper-parameters Analysis

Hyper-parameters in MLPST essentially influence the performance. This subsection will test the impact of the number of MixerLayer N, the number of channel in SpatialMixer C_S and the number of channel in TemporalMixer C_T.

As we can see from Figure 8 (a), the number of MixerLayer N affects the results and time efficiency of the whole experiment. We find that the model performs the worst when $N = 3$. When $N < 6$, our model is relatively effective in terms of training time and exhibits outstanding performance. It proves MLPST's superior efficiency and efficacy in capturing spatio-temporal dependencies. When $N > 3$, We find the results fluctuate but get better as the N continues to increase. The possible reason is that the architecture with deep MixerLayer often has an advantage in capturing global spatial-temporal dependencies. Overall, as the MixerLayer N increases, the model performance gets worse and then better.

For the number of channels in SpatialMixer C_S and TemporalMixer C_T, we set them with equal values, *i.e.,* $C_S = C_T = C$. From Figure 8 (b), the model performs best when $C = 32$.

In Figure 8 (c), we illustrate the performance of our MLPST with different patch sizes. As aforementioned, we patch process the input spatio-temporal feature matrix and then process it with SpatialMixer and TemporalMixer. From the results, we can observe that the best performance emerges when the patch size is 2. Given the input grid matrix with shape of 10×20, it means 50 patches to be handled in parallel. MLPST addresses the correlations among the 50 patches. As the patch size rises to 5 and 10, it means the input matrix is divided into 4 and 2 patches, respectively. Due to the sparse common information among the few patches, MLPST gets worse results. What's more, we also test MLPST when the patch size is 1, *i.e.,* each grid serves as a patch and the input matrix is represented by 10×20 patches. It is not inconceivable that MLPST gets lower performance with patch size of 2, because the patch is utilized to describe local spatial dependency, and taking each grid as a unit eliminates this locality.

4 RELATED WORK

MLP-Based Model. Since the emergence of CNN [18] and ViT [36], they have been the most popular methods for computation vision. Later on, MLP-Mixer [34] is proposed as an all-MLP framework for vision. ZHAO *et al.* [55] argue that MLP-Mixer can achieve strong performance with small-sized models, but it can exhibit serious overfitting when the size of the model increases. MLP-Mixer has been followed by various modified frameworks for better performance or lower complexity. ResMLP [35] is another similar structure by Facebook consisting of Residual MLPs without normalization-based statistics. CycleMLP [5] proposed the idea of Cycle Fully-Connected Layers to enlarge the receptive fields for dense prediction tasks. gMLP [34] proposed to use MLPs with gating and sMLP [32] made adjustments on Token-mixing with sparse interaction, weight sharing and depth-wise convolution.

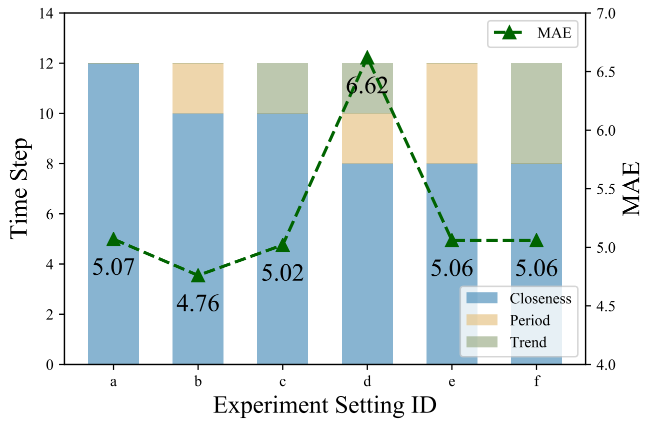

Figure 7: Impact of time step on NYCTaxi dataset.

Unlike the MLP-Mixer methods in the related area, spatio-temporal data mining demands capturing both spatial and temporal relations. In this paper, we propose a novel and effective MLP-based architecture to comprehensively model the global spatial information and the temporal patterns with multiple intervals. To the best of the authors' knowledgement, MLPST is the first effort to address spatio-temporal prediction with an all-MLP framework.

Spatio-temporal Data Mining. Most existing models for STDM tasks often adopt methods like encoders, CNN, graph convolutional networks (GCN), long short-term memory (LSTM), attention mechanisms, *etc.* Seq2Seq [31] and AutoEncoder [16] are designed based on an encoder-decoder architecture for spatio-temporal prediction. STResNet [47] is a CNN-based system designed for citywide ST prediction, and he proposed to make full use of temporal properties like closeness, period, and trend. ResLSTM [45] and Multi-STGCnet [44] both adopt GCN for spatial feature extraction and LSTM for temporal dependency modeling. Conv-GCN [46] also uses GCN to learn spatial correlation, but a 3D CNN is applied in temporal feature mining. T-GCN [49] is a deep neural network method for traffic prediction, named temporal graph convolutional network, which combines GCN and GRU for spatial-temporal feature capturing. ACFM [21] stands for the attention crowd flow machine that can infer crowd flow and learn the dynamic representation of data through the attention mechanism.

Spatio-temporal data mining methods aim to address the complex spatio-temporal dependency, which leads to the highly-complicated architecture and computational complexity. In this paper, we prove

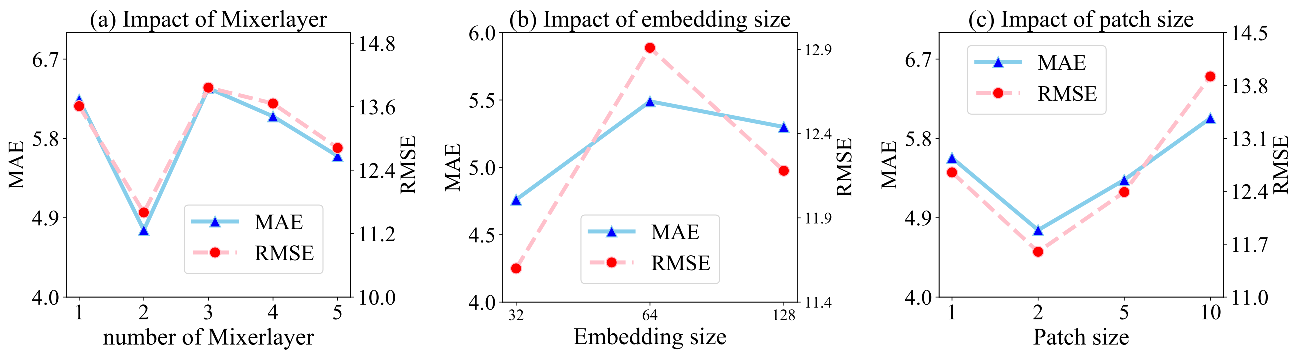

Figure 8: Impact of Hyper-parameters on NYCTaxi dataset.

that MLP-based architecture is enough for spatio-temporal characteristic capture without mechanisms with high complexity. Our effective SpatialMixer and TemporalMixer can effectively model spatial and temporal dependency. It achieves state-of-the-art performance while attaining remarkable time and space efficiency.

5 CONCLUSION

In this paper, we propose a novel framework for solving STDM tasks, MLPST, which is an all-MLP architecture that is simple yet effective. To be specific, MLPST uses SpatialMixer and TemporalMixer targeting spatial and temporal view respectively for feature extraction. On the one hand, SpatialMixer integrates interleaved MLPs on different spatial locations to model spatial dependency with global receptive. On the other hand, TemporalMixer captures the temporal correlations from multiple time intervals, which comprehensively considers temporal dependencies of different spans. We demonstrate the effectiveness of our framework by testing its performances on two real-world traffic flow datasets. The results show the superiority of our proposed framework against baselines, where MLPST attains optimal accuracy with the best efficiency. Furthermore, MLPST is promising to be used in multiple types of STDM tasks in daily urban life, such as air quality prediction, urban energy consumption forecasting, *etc.*

ACKNOWLEDGMENTS

This research was partially supported by APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for New Faculty of City University of Hong Kong), CityU - HKIDS Early Career Research Grant (No.9360163), Hong Kong ITC Innovation and Technology Fund Midstream Research Programme for Universities Project (No.ITS/034/22MS), SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074), SRG-Fd - CityU Strategic Research Grant (No.7005894), Tencent (CCF-Tencent Open Fund, Tencent Rhino-Bird Focused Research Fund), Huawei (Huawei Innovation Research Program), Ant Group (CCF-Ant Research Fund, Ant Group Research Fund) and Kuaishou. Zitao Liu is supported by National Key R&D Program of China, under Grant No. 2022YFC3303600, and Key Laboratory of Smart Education of Guangdong Higher Education Institutes, Jinan University (2022LSYS003). Junbo Zhang is funded by the National Natural Science Foundation of China (62172034), the Beijing Natural Science Foundation (4212021), and the Beijing Nova Program (Z201100006820053). Partial financial support for this work from a Collaborative Research Fund by RGC of Hong Kong (Project No. C1143-20G), a grant from the Natural Science Foundation of China (U20A20189), and a Shenzhen-Hong Kong-Macau Science and Technology Project Category C (9240086). Hongwei Zhao is funded by the Provincial Science and Technology Innovation Special Fund Project of Jilin Province, grant number 20190302026GX, Natural Science Foundation of Jilin Province, grant number 20200201037JC, and the Fundamental Research Funds for the Central Universities, JLU.

REFERENCES

[1] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. 2018. Spatio-temporal data mining: A survey of problems and methods. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–41.
[2] Lei Bai, Lina Yao, Salil Kanhere, Xianzhi Wang, Quan Sheng, et al. 2019. Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. *arXiv preprint arXiv:1905.10069* (2019).
[3] Di Chai, Leye Wang, and Qiang Yang. 2018. Bike flow prediction with multi-graph convolutional networks. In *Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems*. 397–400.
[4] Meng Chen, Xiaohui Yu, and Yang Liu. 2018. PCNN: Deep convolutional networks for short-term traffic congestion prediction. *IEEE Transactions on Intelligent Transportation Systems* 19, 11 (2018), 3550–3559.
[5] Shoufa Chen, Enze Xie, Chongjian Ge, Ding Liang, and Ping Luo. 2021. Cyclemlp: A mlp-like architecture for dense prediction. *arXiv preprint arXiv:2107.10224* (2021).
[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[8] Zulong Diao, Dafang Zhang, Xin Wang, Kun Xie, Shaoyao He, Xin Lu, and Yanbiao Li. 2018. A hybrid model for short-term traffic volume prediction in massive transportation systems. *IEEE Transactions on Intelligent Transportation Systems* 20, 3 (2018), 935–946.
[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. https://doi.org/10.48550/ARXIV.2010.11929
[10] Vivien Sainte Fare Garnot, Loic Landrieu, Sebastien Giordano, and Nesrine Chehata. 2020. Satellite image time series classification with pixel-set encoders and temporal self-attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12325–12334.
[11] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. 2019. Spatiotemporal multi-graph convolution network for ride-hailing

demand forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3656–3663.

[12] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 922–929. https://doi.org/10.1609/aaai.v33i01.3301922

[13] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. 2021. Transformer in transformer. *Advances in Neural Information Processing Systems* 34 (2021), 15908–15919.

[14] Xiao Han, Xiangyu Zhao, Liang Zhang, and Wanyu Wang. 2023. Mitigating Action Hysteresis in Traffic Signal Control with Traffic Predictive Reinforcement Learning. 673–684.

[15] Peilan He, Guiyuan Jiang, Siew-Kei Lam, and Dehua Tang. 2018. Travel-time prediction of bus journey with multiple bus trips. *IEEE Transactions on Intelligent Transportation Systems* 20, 11 (2018), 4192–4205.

[16] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.

[17] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. 2019. Dsanet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 2129–2132.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[19] Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. 2022. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *International Conference on Machine Learning*. PMLR, 11906–11917.

[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*. https://openreview.net/forum?id=SJiHXGWAZ

[21] Lingbo Liu, Ruimao Zhang, Jiefeng Peng, Guanbin Li, Bowen Du, and Liang Lin. 2018. Attentive crowd flow machines. In *Proceedings of the 26th ACM international conference on Multimedia*. 1553–1561.

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10012–10022.

[23] Regina J Meszlényi, Krisztian Buza, and Zoltán Vidnyánszky. 2017. Resting state fMRI functional connectivity-based classification using a convolutional neural network architecture. *Frontiers in neuroinformatics* 11 (2017), 61.

[24] Wanli Min and Laura Wynter. 2011. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies* 19, 4 (2011), 606–616.

[25] R Mohammadi Farsani and Ehsan Pazouki. 2020. A transformer self-attention model for time series forecasting. *Journal of Electrical and Computer Engineering Innovations (JECEI)* 9, 1 (2020), 1–10.

[26] Attila M Nagy and Vilmos Simon. 2018. Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing* 50 (2018), 148–163.

[27] Bei Pan, Ugur Demiryurek, and Cyrus Shahabi. 2012. Utilizing real-world transportation data for accurate traffic prediction. In *2012 ieee 12th international conference on data mining*. IEEE, 595–604.

[28] Honglei Ren, You Song, Jingwen Wang, Yucheng Hu, and Jinzhi Lei. 2018. A deep learning approach to the citywide traffic accident risk prediction. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 3346–3351.

[29] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete Graph Structure Learning for Forecasting Multiple Time Series. In *International Conference on Learning Representations*. https://openreview.net/forum?id=WEHSlH5mOk

[30] Xiaoming Shi, Heng Qi, Yanming Shen, Genze Wu, and Baocai Yin. 2020. A spatial–temporal attention approach for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems* 22, 8 (2020), 4909–4918.

[31] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems* 27 (2014).

[32] Chuanxin Tang, Yucheng Zhao, Guangting Wang, Chong Luo, Wenxuan Xie, and Wenjun Zeng. 2022. Sparse MLP for Image Recognition: Is Self-Attention Really Necessary?. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 2344–2351.

[33] Yumeng Tao, Xiaogang Gao, Alexander Ihler, Kuolin Hsu, and Soroosh Sorooshian. 2016. Deep neural networks for precipitation estimation from remotely sensed information. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1349–1355.

[34] 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.

[35] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. 2021. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404* (2021).

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[37] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? Estimating travel time based on deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[38] Hao Wang, GaoJun Liu, Jianyong Duan, and Lei Zhang. 2017. Detecting transportation modes using deep neural network. *IEICE TRANSACTIONS on Information and Systems* 100, 5 (2017), 1132–1135.

[39] 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.

[40] Senzhang Wang, Jiannong Cao, and Philip Yu. 2020. Deep learning for spatio-temporal data mining: A survey. *IEEE transactions on knowledge and data engineering* (2020).

[41] Senzhang Wang, Jiaqiang Zhang, Jiyue Li, Hao Miao, and Jiannong Cao. 2021. Traffic Accident Risk Prediction via Multi-View Multi-Task Spatio-Temporal Networks. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[42] 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.

[43] Tong Xu, Hengshu Zhu, Xiangyu Zhao, Qi Liu, Hao Zhong, Enhong Chen, and Hui Xiong. 2016. Taxi driving behavior analysis in latent vehicle-to-vehicle networks: A social influence perspective. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1285–1294.

[44] Jiexia Ye, Juanjuan Zhao, Kejiang Ye, and Chengzhong Xu. 2020. Multi-stgcnet: A graph convolution based spatial-temporal framework for subway passenger flow forecasting. In *2020 International joint conference on neural networks (IJCNN)*. IEEE, 1–8.

[45] Jinlei Zhang, Feng Chen, Zhiyong Cui, Yinan Guo, and Yadi Zhu. 2019. Deep-learning architecture for short-term passenger flow forecasting in urban rail transit. *arXiv preprint arXiv:1912.12563* (2019).

[46] Jinlei Zhang, Feng Chen, Yinan Guo, and Xiaohong Li. 2020. Multi-graph convolutional network for short-term passenger flow forecasting in urban rail transit. *IET Intelligent Transport Systems* 14, 10 (2020), 1210–1217.

[47] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-first AAAI conference on artificial intelligence*.

[48] Zijian Zhang, Xiangyu Zhao, Hao Miao, Chunxu Zhang, Hongwei Zhao, and Junbo Zhang. 2023. AutoSTL: Automated Spatio-Temporal Multi-Task Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[49] 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.

[50] Xiangyu Zhao, Wenqi Fan, Hui Liu, and Jiliang Tang. 2022. Multi-Type Urban Crime Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 4388–4396.

[51] Xiangyu Zhao and Jiliang Tang. 2017. Exploring Transfer Learning for Crime Prediction. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 1158–1159.

[52] Xiangyu Zhao and Jiliang Tang. 2017. Modeling Temporal-Spatial Correlations for Crime Prediction. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 497–506.

[53] Xiangyu Zhao and Jiliang Tang. 2018. Crime in Urban Areas:: A Data Mining Perspective. *ACM SIGKDD Explorations Newsletter* 20, 1 (2018), 1–12.

[54] Xiangyu Zhao, Tong Xu, Yanjie Fu, Enhong Chen, and Hao Guo. 2017. Incorporating Spatio-Temporal Smoothness for Air Quality Inference. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1177–1182.

[55] Yucheng Zhao, Guangting Wang, Chuanxin Tang, Chong Luo, Wenjun Zeng, and Zheng-Jun Zha. 2021. A battle of network structures: An empirical study of cnn, transformer, and mlp. *arXiv preprint arXiv:2108.13002* (2021).

[56] Zibin Zheng, Yatao Yang, Jiahao Liu, Hong-Ning Dai, and Yan Zhang. 2019. Deep and embedded learning approach for traffic flow prediction in urban informatics. *IEEE Transactions on Intelligent Transportation Systems* 20, 10 (2019), 3927–3939.